MedTAKMI-CDI: Interactive knowledge discovery for clinical decision intelligence

A. Inokuchi
K. Takeda
N. Inaoka
F. Wakao

This paper describes MedTAKMI-CDI, an online analytical processing system that enables the interactive discovery of knowledge for clinical decision intelligence (CDI). CDI supports decision making by providing in-depth analysis of clinical data from multiple sources. We discuss the fundamental challenges we faced and explain how we met those challenges and developed a prototype experimental CDI system that currently handles clinical information for about 7,000 patients at the National Cancer Center in Japan. We elaborate on a three-layer model (attribute-value pairs, ordered sequences of events, and time-stamped sequences of events) for clinical information, which can represent three different levels of abstraction. This flexibility supports a broad range of analysis, from simple demographic analysis to a mission-critical clinical-path pattern analysis. Rather than a collection of rigid relational schema for clinical information, our relational database system employs a metaschema with patient identifier, time stamp, attribute name, and attribute values. This allows us to modify the representation of clinical information without having to reload the data and rewrite the analytic components. We also describe the analytic functions that are used to understand clinical care practice at the hospital, to obtain an overview of the clinical information, to navigate the clinical information by using the layers of abstraction and the ontologies, and to extract the patterns and rules for clinical paths.

INTRODUCTION

All Japanese citizens are covered by health insurance that is managed by public organizations. Patients can freely choose any clinic or hospital for consultation and treatment. However, there are some problems, such as wide variations in hospital length of stay and in hospital and physician fees. Health-care costs in Japan are increasing rapidly. The Japanese government is introducing a reimbursement system, called diagnosis procedure combination (DPC), that is based on specified fees for specified services. Without reducing the quality of clinical treatment, the DPC payment system is

intended to promote better administrative performance by hospitals. To manage clinical quality and to improve their administrative performance, hospital administrators must obtain evidence and knowledge from the existing medical data stored in the hospitals.

In the early 1980s, hospitals in Japan began using computerized physician order entry systems, and they are now widely used in hospitals. The installation of electronic medical record (EMR) systems, including medical imaging reference functions, is increasing, and personal digital assistants and notebook PCs with wireless local area networks and bar-code readers are widely used in patient wards. A great deal of longitudinal patient clinical data and administrative data is stored digitally in EMR repositories. Data items include medical services (such as prescriptions, injections, laboratory test results, radiological examinations, endoscope data, surgical procedures, and interventions), patient status (such as laboratory test results and pathological diagnoses), outcomes, billing information and costs, hospital income data, and more.

Given this warehouse of data, hospitals and medical institutions need to know which patient groups (for example, based on diagnoses, laboratory test results, or ages before treatment) received what kinds of medical services and in which order (for example, radiation therapy, chemotherapy, or surgical operations), and whether the outcomes were good or bad (for example, in diagnosis stage categorization). The analytics of such kinds of pattern extraction and rule finding from the actual data of clinical and administrative processes would be helpful to support treatment selection decisions by medical staff members and patients. The extracted patterns and rules are also useful for developing clinical pathways and guidelines. A *clinical pathway* is the sequence of a plan of care, predictable multidisciplinary interventions, and expected patient outcomes, drafted in advance for patient groups.

Multidimensional database technology is one of the key tools for interactive analysis of large amounts of data for decision-support purposes. In the traditional multidimensional data models intended for online analytic processing (OLAP), data is viewed as specifying points in multidimensional space. For example, the sale of a particular item in a particular store of a retail chain can be viewed as a point in a space whose dimensions are the product, location, and time, and this point is associated with one or more measures, such as price or profit. Pedersen and Jensen described nine requirements and proposed a multidimensional data model to analyze more complex data, such as clinical records, using a real-world medical case study.[1] The proposed model used a history of each patient as a fact and aggregated the number of patients grouped by their diagnoses. The relationship between a fact and each dimension for the clinical data is not always a many-to-one mapping. For example, some patients have several diagnoses, although the relationships in the classical model are many-to-one mappings. In accord with some of the requirements for their conceptual model, this paper further enhances the OLAP for clinical records to respond to complex queries on high volumes of data.

In building a decision-support solution, we identified some fundamental challenges in modeling clinical information and ontologies. The first challenge was designing a database and data warehouse system for clinical information management. The second was how to implement interconnected analytic functions for knowledge discovery and rule generation. Based on our experiences at the National Cancer Center in Japan, we developed responses to these challenges and prototyped an experimental system for clinical decision intelligence (CDI). The system now runs with clinical information for about 7,000 patients and has been tested for analyzing correlations among cancers, tumor markers, and clinical treatments. In this paper, we describe the technical aspects of these challenges and our approach to building the CDI solution. In particular, we elaborate on a three-layer model of clinical information (using attribute-value pairs, ordered sequences of events, and time-stamped sequences of events), which represents three different levels of abstraction. This flexibility is important to support a broad range of analyses, from simple demographic analyses to a mission-critical clinical-path pattern analysis. Rather than a collection of rigid relational schema for clinical information, our relational database system employs a metaschema—a schema about the schema—with time stamps, patient identifiers, attribute names, and attribute values. This allows us to modify the representation of clinical information without the time-consuming work of reloading data and rewriting analytic components. We also describe our collection of

analytic functions for such uses as understanding clinical care practice at the hospital, constructing overviews of the clinical information, navigating the clinical information by using ontologies (dimensional hierarchies), and extracting the patterns and rules for clinical paths.

The remainder of this paper is organized as follows. We describe a traditional multidimensional database and OLAP and discuss the use of OLAP for clinical records. We propose a data model and its implementation to solve the issues that were introduced and to support rapid computation. We then introduce MedTAKMI-CDI, the prototyped system, and its functions. We provide some scenarios using real-world clinical data from the National Cancer Center in Japan, we put our work in the context of related work, and then draw our conclusions.

ISSUES FOR OLAP

This section explains traditional databases and OLAP and lists some issues of OLAP for medical records.

Traditional multidimensional databases and OLAP

Multidimensional database technology is a key factor in the interactive analysis of large amounts of data for decision-making purposes.[2] Multidimensional models categorize data either as facts associated with numerical measures or as textual dimensions that characterize the facts (*Figure 1A*). For a retail business, a purchase would be a fact, and the purchase amount and price would be measures; the type of product purchased and the time and location of the purchase would be dimensions. OLAP queries aggregate measures over a range of dimensional values to provide results, for example, total sales per month of a given product, which then lead to identifying trends.

An important feature of multidimensional modeling is to use hierarchical dimensions to provide as much context as possible for the facts. Dimensions are used for selecting and aggregating data at the desired level of detail. Most traditional multidimensional data models assume that dimension hierarchies are balanced and nonragged trees, as shown in *Figure 1B*. (For an explanation of types of hierarchies, see Reference 3.)

A multidimensional database lends itself to certain types of interactive queries:

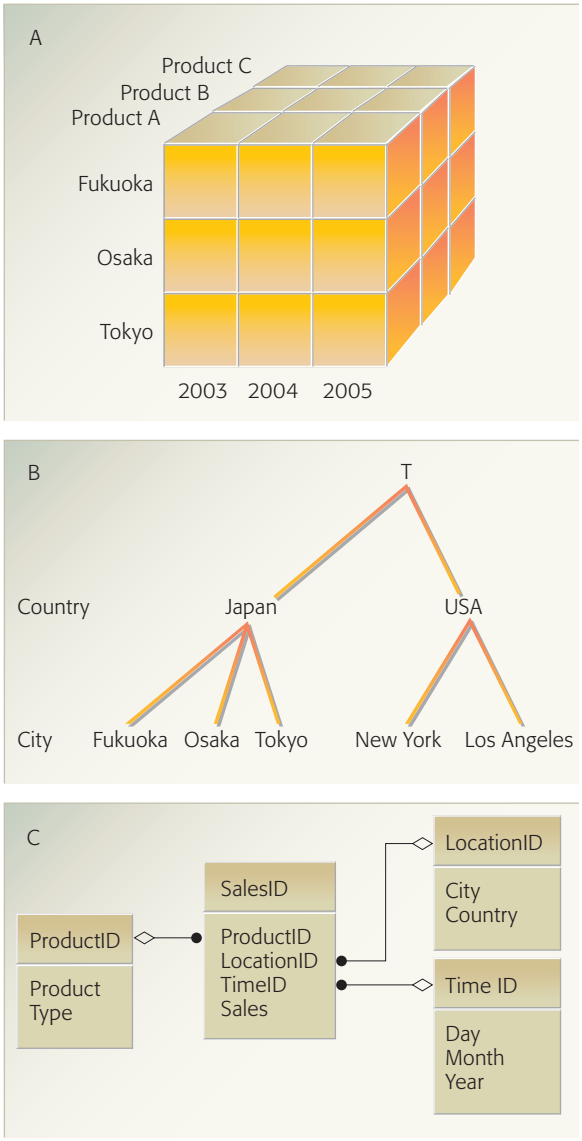

Figure 1
Traditional multidimensional database: (A) cube view of a multidimensional model; (B) dimension hierarchy of location; (C) star schema

- So called "slice-and-dice" queries make selections for dimensional reduction by focusing on certain data. Selecting a single dimension value reduces the dimensionality of the cube. For example, we can slice the cube by considering only those cells that relate to a specific dimensional value, and then further reduce this slice by considering only the cells for another dimensional value in a different dimension.
- "Drill-down and roll-up" queries are inverse operations that use dimension hierarchies and

Table 1 Table schemas in an EMR system

Profile	profile (patientID, gender, birthDate, dateOfFirstVisit, liverDysfunction, renalDysfunction, ...)
History	careHistory (patientID, dateOfAdmission, department, ...)
Examination	laboratoryTest (patientID, date, material, testName, result, ...) pathologicalDiagnosis (patientID, date, cytoscreeningOrTissueDiagnosis, substance, diagnosis, ...) physiologicalExamination(patientID, date, type, ...) endoscopicalExamination(patientID, date, type, ...) radiologicalDiagnosis(patientID, date, type, ...)
Therapy	surgery (patientID, date, careGroup, operativeProcedure1, operativeSite1, ..., operativeProcedure 10, operativeSite10, ...) radiologicalTherapy(patientID, startDate, endDate, date, equipment, site, ...) endoscopicTherapy (patientID, date, type, ...) chemotherapy (patientID, date, type, ...) bloodInfusion (patientID, date, type) injection (patientID, date, number, medicine, ...) prescription (patientID, date, number, medicine, ...)
Diagnosis	admission (patientID, dateOfAdmission, dateOfDischarge, diseaseNameOnAdmission, diseaseNameOndischarge, outcome, ...) dischargeSummary(patientID, dateOfDischarge, number, diseaseName, icd10, stageOfCancer)

measures to perform aggregations. Rolling up to a top value corresponds with omitting the dimension from dimension values at a finer granularity to those at a coarser granularity. For example, in Figure 1B, rolling from *City* to *Country* aggregates the values for Los Angeles and New York into a single value, *USA*.

- Rotating a cube allows users to see the data grouped by other dimensions.
- Ranking, or "top *n*" queries, return only those cells that appear at the top of the specified order.

Relational OLAP, which is one of the implementations of multidimensional databases, typically uses star or snowflake schemas, both of which store data in fact tables and dimension tables. As shown in *Figure 1C*, a fact table holds one row for each fact in the cube, and it has a column for each measure that contains the measured value for the particular fact and a column for each dimension that contains a foreign key referring to a dimension table for the particular dimension.

OLAP for medical records

Table 1 shows the schemas derived from an EMR system created by IBM Japan. The schemas are categorized into five groups. Tables in the first group store patient profiles, which contain data such as gender, birth date, medical history of liver dysfunction, renal dysfunction, and so forth. Tables in the second group contain medical histories. For example, the table "admission" contains the dates of admission to and discharge from a hospital and the number of days in each hospital stay. The table "careHistory" contains the dates when and medical departments in which patients received medical treatments. The third group of tables contains data for examinations, such as laboratory tests, pathological diagnoses, physiological examinations, endoscopic examinations, and radiological diagnosis. Values in the underlined columns, such as "material" and "testName", are stored as foreign keys referring to master tables. The fourth group is tables containing the treatment events. Although the operative procedures performed at various operative sites in a surgical operation event are stored as one instance in the "surgery" table, drugs dispensed at the same time are stored in the "dispenseDrug" table. The fifth group is the diagnosis leading to hospitalization. Data in the column "outcome" of the table "admission" is subjectively assigned to each patient by a physician. The "icd10" value is determined by the standard classification of diseases.[4] Except for the birth date, all date values in all of the tables are stored as time stamps.

Table 2 shows an example of an analysis for the medical histories of patients admitted to the hospital. Axes that can be selected besides the operative procedures and radiological examinations include the types of chemotherapies, radiation therapies, endoscopic therapies, laboratory tests,

Table 2 The number of patients who had the radiological examination and operative procedure

Operative Procedure	Radiological Examination									
	Chest	Abdominal	Chest (portable)	Chest CT†	Liver-pelvis CT	Chest (portable), upper abdominal (portable)	Chest CT, liver-pelvis CT	Brain MR††	Liver-pelvis CT, chest CT	Chest (portable), lower abdominal (portable)
Mastectomy	00	01	02	03	04	05	06	07	08	09
Distal gastrectomy	10	11	12	13	14	15	16	17	18	19
Colectomy	20	21	22	23	24	25	26	27	28	29
Total gastrectomy	30	31	32	33	34	35	36	37	38	39
Transurethral resection of the bladder tumor	40	41	42	43	44	45	46	47	48	49
Thoracoscope	50	51	52	53	54	55	56	57	58	59
Radical prostatectomy	60	61	62	63	64	65	66	67	68	69

† CT = computerized tomography
†† MR = magnetic resonance

and physiological examinations and patient profiles containing gender, birth date, and so forth. The dimension selected as an axis may be a hierarchical dimension. For example, ***Figure 2*** shows the instances of a therapy dimension for medical records. Of the therapy dimension's five levels, the detailed operative procedure is the lowest level. The operative-procedure-level values are grouped into medical-care group level values. For example, gastrectomy and laparoscopic surgeries are grouped into the practice group for the stomach.

When we analyze such clinical records with the traditional commercial multidimensional databases,

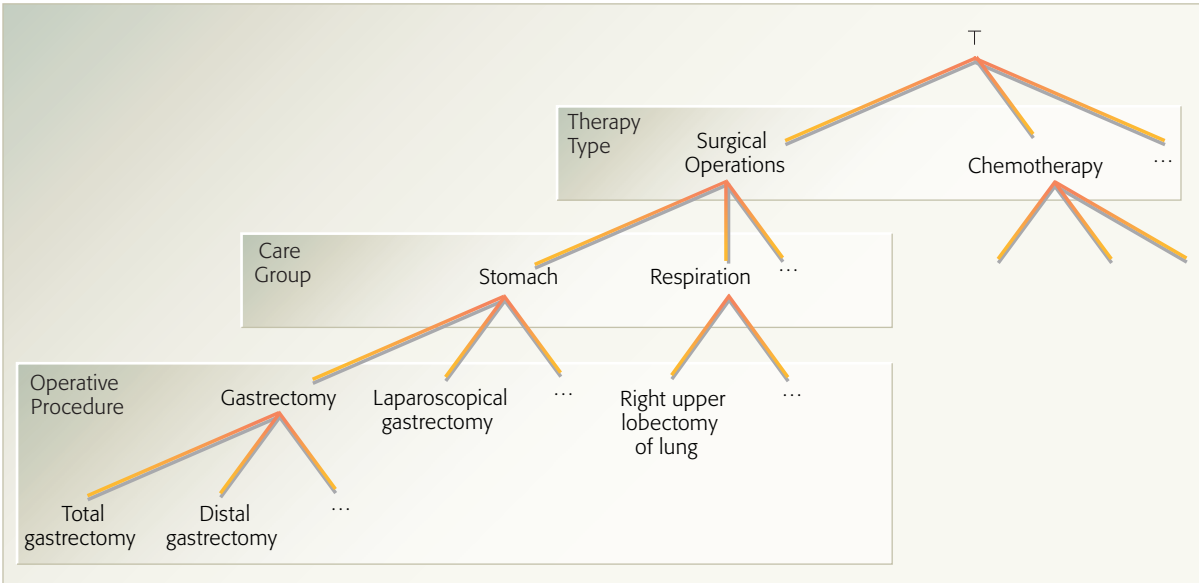

Figure 2
Hierarchy of therapy dimension

we encounter the difficulties explained in the following subsections.

Complex hierarchy and multiple dimension values

For a retail business, a purchase corresponds to a fact, and each fact can be mapped to a point in a three-dimensional space, where the dimensions are the location of the store, the product sold, and the purchase date. In other words, each fact has exactly one dimension value in each dimension. In addition, most traditional multidimensional data models assume that dimension hierarchies are balanced and nonragged trees. However, for medical records, each patient may have many medical treatments, examinations, and records of patient statuses. In addition, medical treatments are segmented into surgical operation, chemotherapy, and radiation therapy, and the patient may have many different types of treatments and different types of examinations. The dimension hierarchies that we intend to use are not balanced trees, as shown in Figure 2. For example, in the data used in Table 2, the average number of laboratory tests that a patient had during one hospital stay was more than 200. The patient also had physiological examinations, radiological examinations, and may have had endoscopic examinations. Therefore, it is impossible to store medical data in a star schema or snowflake schema, which are often used for OLAP.

Specification of arbitrary intervals

In traditional multidimensional databases, slicing by a single dimension value reduces the cube's dimensionality, which corresponds to narrowing down all of the facts into a subset. For medical records, the aggregates returned by a ranking query for a laboratory test, after slicing by admission to and discharge from the hospital, contain the tests performed during the hospital stay and the outpatient tests performed preadmission. Doctors, however, would like to aggregate only those tests performed during the hospital stay. Although one solution is to assign an identifier to each hospitalization, this does not allow for specifying arbitrary intervals, such as from patient's first visit until admission into the hospital or from the date of a surgical operation until 10 days after discharge from the hospital.

Measure

For a retail business, the purchase amount and price would be measures. Measures can be combined along any dimension, which allows for precomputation. One of the measures for medical records is the number of patients, as shown in Table 2. Because each patient has many values for each dimension, it is impossible to simply combine lower-level values along any dimension for rolling up. In addition, depending on the needs of the analysis, the numbers of arbitrary intervals and events would be measures. For example, the data should be viewed by separately counting each interval, such as a hospital stay. In addition, administrators, managers, and medical staff personnel would like to aggregate the number of events, such as surgical operations and laboratory tests, for use in determining how to reduce costs.

Temporal order among dimension values

For medical records, a value in each dimension corresponds to an event with a time stamp, and there is a temporal order among the dimensional values. For example, there are cases in which patients with larynx cancer have the surgical operation after reducing the size of the tumor with chemotherapy or radiation therapy, and where patients have chemical or radiation therapy to prevent recurrence of cancer after the surgical operation on their larynxes. The OLAP system for medical records needs to have a function to aggregate the number of patients distinctly for these various cases.

Performance for interactive analysis

A key strategy to speed up cube-view processing, as shown in Figure 1A, is to use precomputed cube views. The precomputation makes it possible for query response time involving potentially huge amounts of data to be fast enough to allow interactive data analysis in the traditional approaches. However, OLAP for medical records cannot precompute or preaggregate in advance of receiving queries, because the number of all combinations of values can be prohibitively large.

To overcome the preceding difficulties, we designed a prototype system, MedTAKMI-CDI. The predecessor of this system, IBM Text Analysis and Knowledge Mining (TAKMI), is a text-mining system used to mine customer-support call logs for customer relationship management[5] and to mine biomedical documents for the life sciences.[6] In the next sections, we give a detailed description of how to model OLAP for medical records and how to support fast response times.

DATA MODEL

In this section, we give formal definitions of a hierarchy, an ontology, and our data model according to Bonatti et al.,[7] Pedersen and Jensen,[1] and Inokuchi and Takeda.[8]

If S is a nonempty set, and $<\subseteq S \times S$, then $(S, <)$ is an ordering. Although a definition of the ordering is generally represented as $\leq\subseteq S \times S$, this paper uses $<$ to represent a direct relation between two elements in the set S, $\ll$ to represent its transitive closure, and $\leq$ to represent its transitive closure or represent that the elements are equal. If $x < x$ for $x \in S$, then S is reflexive. If $x < y$ and $y < z \rightarrow x < z$ for $x, y, z \in S$, then S is transitive. If $x < y$ and $y < x \rightarrow x = y$ for $x, y \in S$, then S is antisymmetric. $(S, <)$ is a partial ordering if S is a reflexive, transitive, and antisymmetric binary relation on S.

Definition 1 (better): Let $(S, <_1)$ and $(S, <_2)$ be two orderings. We say $(S, <_1)$ is better than $(S, <_2)$ iff $\forall x, y \in S\ (x <_1 y \rightarrow x <_2 y)$. In addition, we say that $(S, <_1)$ is strictly better than $(S, <_2)$ iff $(S, <_1)$ is better than $(S, <_2)$ and $(S, <_2)$ is not better than $(S, <_1)$.

Definition 2 (hierarchy): Let $(S, <)$ be a partial ordering. A hierarchy of S is an ordering $(S, \prec)$ such that $(S, \prec)$ is better than $(S, <)$, $(S, <)$ is the reflexive, transitive closure of $(S, \prec)$, and there is no other ordering $(S, <_1)$ satisfying the preceding two conditions such that $(S, <_1)$ is strictly better than $(S, <)$.

Definition 3 (ontology): Suppose Σ is some finite set of strings and S is some set. An ontology with respect to Σ is a partial mapping θ from Σ to hierarchies for S.

For example, when S is given as {tire, car, hubcap}, where tire is a part of car, hubcap is a part of car, and hubcap is a part of tire. In addition, everything is a part of itself. For the set S, a partial order is defined as {(tire, tire), (car, car), (hubcap, hubcap), (tire, car), (hubcap, car), (hubcap, tire)}, and only one hierarchy is defined as {(tire, car), (hubcap, tire)}.

Given a hierarchy (or an ontology) $(S, <)$, a fact schema is defined as $S' = (F', T')$, where F' is a fact type and T' is a hierarchy type, $T' = (C', <_{T'}, top_{T'})$, which is strictly better than $(S, <)$, and the relations in $(S, <)$ required for analyzing the documents are remaining in T'. The hierarchy type is a three-tuple $(C', <_{T'}, top_{T'})$, where $C'=\{C'_j, j=1, ..., n\}$ is a set of category types of T', and $<_{T'}$ is a partial order on the C's, with $top_{T'} \in C'$ being the top element of the ordering. The intuition is that the top element of the ordering logically contains all other elements; that is, $\forall C'_j \in C'$, $C'_j \leq top_{T'}$. *A* hierarchy instance T of type T' is a two-tuple $T = (C, <)$, where C is a set of categories c_j such that $Type(c_j) = C'_j$, and $<$ is a partial order on C. Each category $c \in C$ has an associated set $dom(c)$ called its domain. The members of $dom(c)$ are called values of the category c. An element in $dom(c)$ is represented as $c : v$.

Let $F = \{f_i, i = 1, ..., m\}$ be a set of facts. Each fact corresponds to a patient. A fact-hierarchy relationship between F and T is a set $R = \{(f, t, c : v)\}$, where $f \in F$, $c \in C$, and $v \in dom(c)$. $(f, t, c : v)$ represents that an event which is described by a term v of category c occurs at time t for a patient f. Thus, R links facts to hierarchical values. Our data object is a four-tuple $D = \{S', F, T, R\}$, where $S' = (F', T')$ is the fact schema, F is a set of facts where $Type(f) = F'$, $T = (C, <)$ is a hierarchy instance where $Type(c_j) = C_j$ for $c_j \in C$ and $C'_j \in C'$, and R is a set of fact-hierarchy relations such that $(f, t, c : v) \in R \rightarrow f \in F \wedge \exists c \in C(v \in dom(c))$.

Conceptually, R corresponds to a relation $P \subseteq 2^{dom(c_1)} \times \ldots\ 2^{dom(c_n)}$, which is not a normalized relation. P corresponds to a fact table for a star schema, and each row and column in P corresponds to a patient and a category (dimension value in the star schema), respectively, A naive method cannot store the data in a relational database and cannot efficiently aggregate the data along the hierarchy. The first reason that it cannot do so is that the relation has many missing values and a set of values for each attribute c_j. The second reason is that the number of attributes in the relation becomes very large. For example, the number of categories c_j in our prototype is about 250,000. The third reason is that a complex relationship among the attributes (columns) exists.

IMPLEMENTATION

As explained earlier in the section "Issues for OLAP," medical record data cannot be precomputed and preaggregated in advance of receiving queries. We must design table schema or data structures to achieve query response times that are as fast as

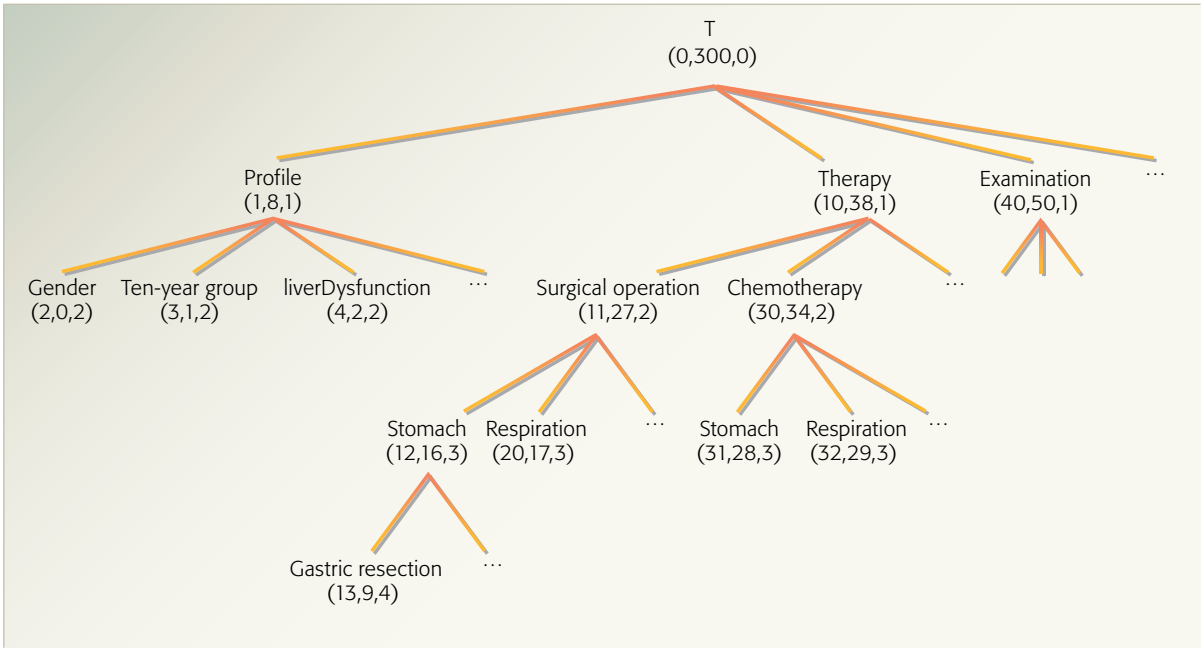

Figure 3
Category tree for medical records

possible. Dimension hierarchies for our medical OLAP constitute a general tree rather than a set of balanced trees, and in our schema, each path from the root node to a leaf node corresponds to a record in a dimension table of a star schema. For medical records, the hierarchy is modeled as a tree rather than a forest to allow for multiple hierarchies. We call the hierarchy a *category tree*. The category tree is stored in the following CATEGORY table, in which each row contains the information pertaining to a particular node. The table is defined as

```
CATEGORY    (PATH          CHARACTER,
             DESCRIPTION   CHARACTER,
             PREORDER      INTEGER,
             POSTORDER     INTEGER,
             DEPTH         INTEGER,
             PARENT        INTEGER).
```

In the table, PATH represents a path from the root node to the node corresponding to its record, and DESCRIPTION is its node's name. PREORDER, POSTORDER, and DEPTH are a preorder, postorder, and depth assigned to the category node for calculation efficiency, respectively, and PARENT is a preorder of its parent node. For example, ***Figure 3*** shows an example of a part of a category tree. All leaves in the dimension hierarchy of Figure 2 are stored as values in a table EVENT. A label for each node, such as "Surgical operation" or "Chemotherapy", is stored as the node name. Numbers below the node name are the preorder, postorder, and depth that are assigned to the node, respectively. The 10-year age group is calculated from each patient's birth date.

In addition to the CATEGORY table, a table EVENT whose records correspond to the lowest-level values in the fact table of a star schema is defined as

```
EVENT       (PATIENTID    INTEGER,
             DATE         DATE,
             PREORDER     INTEGER,
             VALUE1       CHARACTER,
             VALUE2       DOUBLE,
             EVENTID      CHARACTER).
```

In this table, PATIENTID is an identifier for a patient, DATE is the date when an event occurs. PREORDER is a preorder of the category node to which the event refers. It is not necessarily the case that the referred-to node is a leaf node in the category tree. VALUE1 and VALUE2 are detailed values that the event describes. For example, ***Table 3A*** shows a history containing three surgical events. In preprocessing, information in the table is converted into instances in the EVENT schema, as shown in ***Table 3B***, where values in the column EVENTID represent the IDs to identify events that occur at the same time. As

Table 3 Preprocess of MedTAKMI-CDI: (A) example of a table "Surgery"; (B) example of a table "EVENT"

A					
PatientID	**Date**	**Care group**	**Surgery1**	**Surgery2**	**Surgery3**
1	2006/04/05 12:51	stomach	Total gastrectomy	laparoscopical operation	NULL
1	2006/05/12 08:12	respiration	Ablation of right upper lobe of lung	NULL	NULL
2	2006/04/05 13:22	stomach	Total gastrectomy	NULL	NULL
B					
PATIENTID	**DATE**	**PREORDER**	**VALUE1**	**VALUE2**	**EVENTID**
1	2006/04/05 12:51	13	Total gastrectomy	NULL	1
1	2006/04/05 12:51	12	Laparoscopical operation	NULL	1
1	2006/05/12 08:12	20	Ablation of right upper lobe of lung	NULL	2
2	2006/04/05 13:22	13	Laparoscopical operation	NULL	3

shown in Table 1, the table "dischargeSummary" contains information such as diagnosis, which is not an event but a statement recorded on the date of discharge. In this case, `DATE` and `VALUE1` in a record for the diagnosis in the `EVENT` table become the observed date and the diagnosed disease name, respectively. In addition, records for laboratory tests contain information in their numerical results. In this case, `VALUE1` and `VALUE2` in the corresponding record in the `EVENT` table become `NULL` and the numerical value, respectively.

The preceding data schema allows for fast aggregation at the desired level of detail rather than for leaf-level values in the category tree. For example, this makes it possible to roll up from detailed operative procedures to care groups. The `PREORDER`, `POSTORDER`, and `DEPTH` in the tables `CATEGORY` and `EVENT` are used to handle ancestor-descendant containment in a tree.[9] The method for checking the containment is by assigning a preorder and a postorder to each node in the tree, as shown in Figure 3, and then comparing the numbers assigned to the two nodes. If node A is an ancestor of node B, the preorder of A must be less than the preorder of B, and the postorder of A must be greater than the postorder of B. Because the ancestor-descendant containment can be represented as the relationship of preorder and postorder without using any functions to process the strings, the method can quickly aggregate the various distributions.[8,10]

Concretely, the higher-level values are derived using the following SQL query:

```
SELECT PATIENTID, DATE, pre1, DESCRIPTION AS
    VALUE1, NULL, EVENTID
FROM EVENT, CATEGORY
WHERE
  PARENT = pre1 AND EVENT.PREORDER > pre1 AND
  EVENT.PREORDER <= (post1 + dep1) AND
  EVENT.PREORDER >= CATEGORY.PREORDER AND
  EVENT.PREORDER <= POSTORDER + DEPTH,          (1)
```

where `pre1`, `post1`, and `dep1` are a preorder, postorder, and depth assigned to the parent of the desired level node in the category tree. Because the derived result is the same structure as the original data, we can continue the query process by using the results of the previous query operations, which allows physicians to analyze medical records in an interactive manner.

Here are three examples for ranking, rolling up, and slicing queries, respectively. The first is a Structured Query Language (SQL) query to aggregate the number of patients who had surgical operations in a stomach care group for each operative procedure and to rank the aggregated numbers. This query is represented as

```
SELECT VALUE1, COUNT(DISTINCT PATIENTID) AS
    COUNT
FROM EVENT
```

```
WHERE PREORDER >= pre2 AND PREORDER <= (post2 +
    dep2)
GROUP BY VALUE1
ORDER BY COUNT DESC,                              (2)
```

where pre2, post2, and dep2 are the preorder, postorder, and depth of a care group for stomachs.

Another SQL query to aggregate the number of surgical events for each care group is represented as

```
SELECT VALUE1, COUNT(DISTINCT EVENTID) AS COUNT
FROM
  (
    SELECT PATIENTID, DESCRIPTION AS VALUE1,
         EVENTID
    FROM EVENT, CATEGORY
    WHERE
      PARENT = pre3 AND EVENT.PREORDER > pre3 AND
      EVENT.PREORDER <= (post3+dep3) AND
      EVENT.PREORDER >= CATEGORY.PREORDER AND
      EVENT.PREORDER <= POSTORDER+DEPTH) EVENT
  )
GROUP BY VALUE1 ORDER BY COUNT DESC,               (3)
```

where pre3, post3, and dep3 are the preorder, postorder, and depth assigned to the category node for surgical operation in Figure 3.

When slicing by considering only patients who have a certain chemotherapy, val4, the SQL (2) query is modified as

```
SELECT VALUE1, COUNT(DISTINCT PATIENTID)
FROM EVENT
WHERE
  PREORDER >= pre2 AND PREORDER <= (post2+dep2)
      AND
  PATIENTID IN (
    SELECT PATIENTID
    FROM EVENT
    WHERE
      PREORDER >= pre4 AND
      PREORDER <= (post4+dep4) AND
      VALUE1=val4
  )
GROUP BY VALUE1,                                  (4)
```

where pre4, post4, and dep4 are the preorder, postorder, and depth for the category node for that chemotherapy.

MedTAKMI-CDI always maintains intervals as views when slicing by considering the patients who satisfy certain conditions. For example, an SQL query to specify intervals from admission to discharge from the hospital is represented as

```
SELECT PATIENTID, BEGIN, BEGIN+MIN(DATE-BEGIN)
    AS END
FROM
  (
    SELECT A.PATIENTID, DATE, B.BEGIN, B.END
    FROM
      (SELECT PATIENTID, DATE FROM EVENTM
          WHERE PREORDER = pre6 ) A,
      (
        SELECT PATIENTID, DATE AS BEGIN,
            CURRENT DATE AS END
        FROM EVENT WHERE PREORDER = pre5
      ) B
    WHERE A.PATIENTID=B.PATIENTID AND
        DATE>=BEGIN
  ) A
GROUP BY PATIENTID, BEGIN, END,                   (5)
```

where pre5 and pre6 are the preorders assigned to the nodes corresponding to admission to and discharge from the hospital, respectively, when the nodes have no children nodes. Because patients may be admitted to a hospital many times, the SQL specifies the first discharge after each admission by using the function MIN.

MEDTAKMI-CDI

In the earlier sections, we presented the data models and the basic implementation. This section describes representative functions of MedTAKMI-CDI.

Target selection

The target selection is a function to slice by considering only patients who meet certain conditions. The upper right frame in ***Figure 4*** shows the results returned by SQL code segment (5). The figure represents specifying intervals "before" the "first" discharge from the hospital "after" "all" admissions for each patient. The first column can be selected from among "normal", "before (<)", "before (<=)", "after (>)", "after (>=)" and "on the same day". The "before (<)" means that the selected interval does not include the day of the event, such as the day of discharge. The "on the same day" allows a user to make the start date and the end date of an interval the same. The second column can select

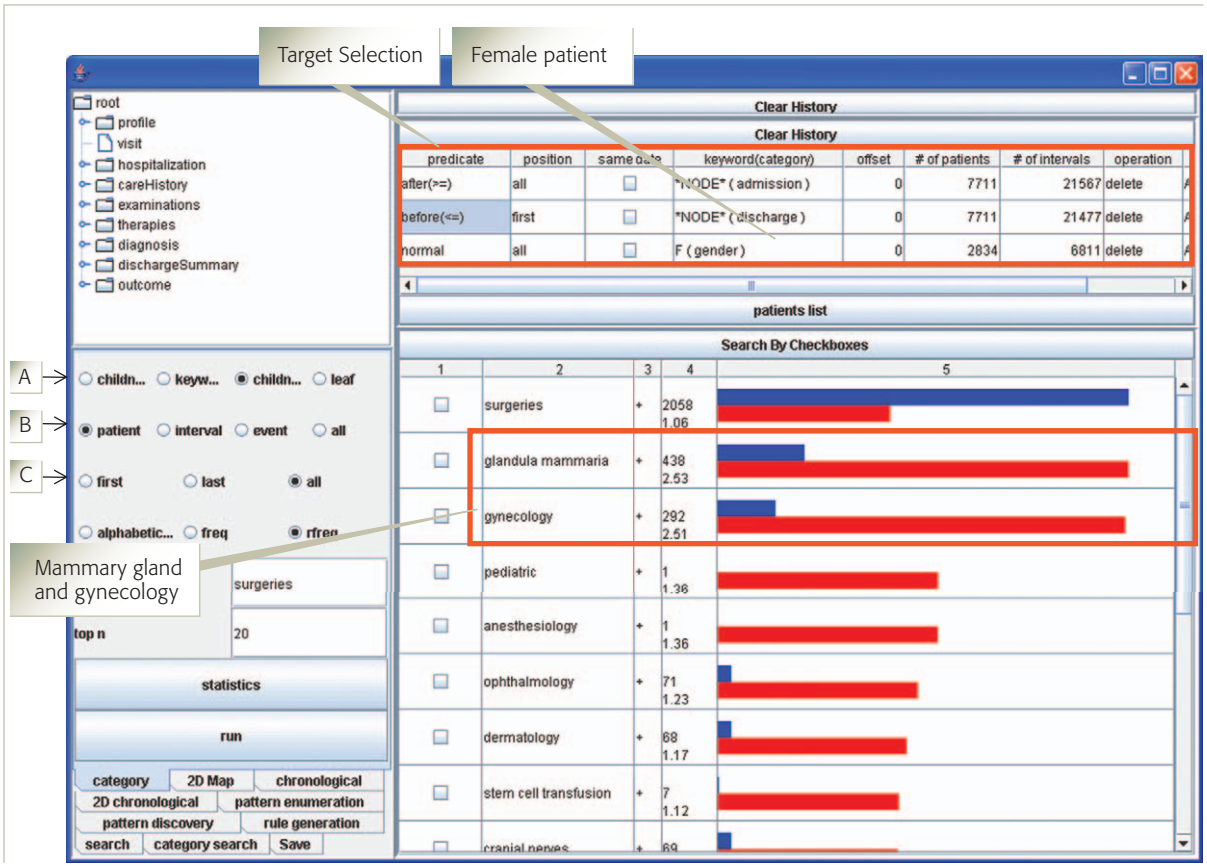

Figure 4
Category hierarchy viewer

from "all", "first", and "last". By checking a box in the third column, a physician can specify intervals "before" the "first" discharge from the hospital "after" the date of "all" admissions in which the patient had surgical operations. By using "offset" in the fifth column, physicians can select intervals from the date of admission to 10 days after discharge from the hospital. By clicking "delete" in the eighth column, a condition that narrowed down into a subset of the patients is deleted.

Hierarchical category viewer

The hierarchical category viewer returns the number of patients, events, or intervals for each child node of a category node selected by a physician. This function corresponds to a ranking query in a traditional multidimensional database. For example, Figure 4 shows the distribution of patients for each care group after narrowing down into female patients by target selection.

The blue bars in the lower right frame of the viewer show the numbers of patients. In Figure 4, there are 438 patients who had surgical operations in the care group for mammary glands. Red bars indicate relative frequencies, comparing each subset of patients to the initial set of patients. For defining this, let S be the initial set of all patients. The target selection due to some conditions returns S_1, which is the subset of S that satisfies the conditions. The relative frequency for a category c in the subset S_i is calculated using the following formula:

$$\text{relfreq}(c, S_i) = \frac{\text{freq}(c, S_i)}{|S_i| \dfrac{\text{freq}(c, S)}{|S|}}$$

where $|S_i|$ is the number of patients in the set S_i and $\text{freq}(c, S_i)$ is the number of patients who have an event represented by the category c in the collection S_i. As female patients are already narrowed down, the relative frequencies for the groups of mammary

gland and gynecology in Figure 4 are higher than the others.

By selecting one of the radio buttons pointed at by 'A' in Figure 4, a physician can aggregate the numbers for each child node of a selected category node, for each value with a preorder that is referring to a selected category node, or for each value with a preorder that is referring to a descendant of a selected category node. By selecting one of the radio buttons pointed at by 'B' in Figure 4, the user can aggregate the number of patients, events, or intervals for the selected category nodes. By selecting one of the radio buttons pointed at by 'C' in the figure, the first event, the last event, or all of the events for each specified interval in the target selection can be aggregated.

Clicking one of the results of the category view and the other functions leads to narrowing down the patients into a subset of those who satisfy the condition corresponding to the selected result, and the condition is added into the set of conditions in the target selection. Because this system is interactive, the physicians are better able to discover hidden knowledge by using a combination of mining functions and trial-and-error approaches.

Chronological viewer

This viewer allows a physician to discover trends by viewing the chronological distribution of a set of patients. This function supports yearly, quarterly, monthly, and daily distributions. Using this viewer, one can investigate how the frequency of some occurrence changes with time.

Two-dimensional map viewer

The two-dimensional viewer allows a physician to visualize the strength of an association between events. ***Figure 5*** shows associations between events of surgical operations and radiological examinations for hospital patients. The values in each cell represent the strength of the association of those two events—the higher the value, the stronger the association. For example, the operative group "stomach" and the radiological examination "chest X-ray exam" have a strong association, which means that many patients have a chest X-ray exam after having surgical operations on their stomach. The numbers "570 (1.11)" in the cell mean that there were 570 patients who mentioned both the distal gastrectomy and the chest X-ray exam, and that its relative frequency was 1.11. Formally, the relative frequency of the two-dimensional maps is calculated by using the following formula:

$$\mathrm{relfreq}(c_1, c_2, S_i) = \frac{\mathrm{freq}(c_1 \; and \; c_2, S_i)}{|S_i| \dfrac{\mathrm{freq}(c_1, S_i)}{|S_i|} \dfrac{\mathrm{freq}(c_2, S_i)}{|S_i|}}$$

where S_i is a set of patients selected by the target selection and $freq(c_1 \; and \; c_2, S_i)$ is the number of patients who have events c_1 and c_2. The events c_1 and c_2 belong to the categories of the x- and y-axes, respectively.

The pull-down menu pointed to by 'A' in Figure 5 can select from "default", "V=C", "V=(E)=C", "V<=C", "V<C", "V>=C" and "V>C". As examples, "V<C" means that an event on the vertical axis occurs before an event on the horizontal axis, and "V=(E)=C" means that events on the vertical axis and on the horizontal axis happen on the same day and that their `EVENTID`s in the table `EVENT` are same.

Two-dimensional chronological viewer

The two-dimensional chronological viewer is a function to display the chronological distribution around a selected event. ***Figure 6*** shows distribution of surgical operations around dates of admission indicated by a vertical red line. In each row of the figure, a red plot line, a blue plot line, and a black plot line show the distribution of surgery events in each care group for male patients, female patients, and all patients, respectively. The vertical gray grid lines indicate the number of days following admission. A physician can find care groups of patients who spent a long time in the hospital before their surgical operations, which may be important for better management of hospitals.

Pattern enumeration

The pattern enumeration viewer is a function to enumerate frequently concurring patterns from a set of events or a set of event sequences.[11,12] Events contained in each pattern are interactively selected before running this function, and it returns generalized patterns by using the category tree.[13,14] ***Figure 7*** shows the discovered frequent sequential patterns when the events of admissions, discharges, and surgical operations are used as items. For example, the pattern in the first row means that 398 patients had a surgical operation in the care group for liver, gallbladder, and pancreas in an average

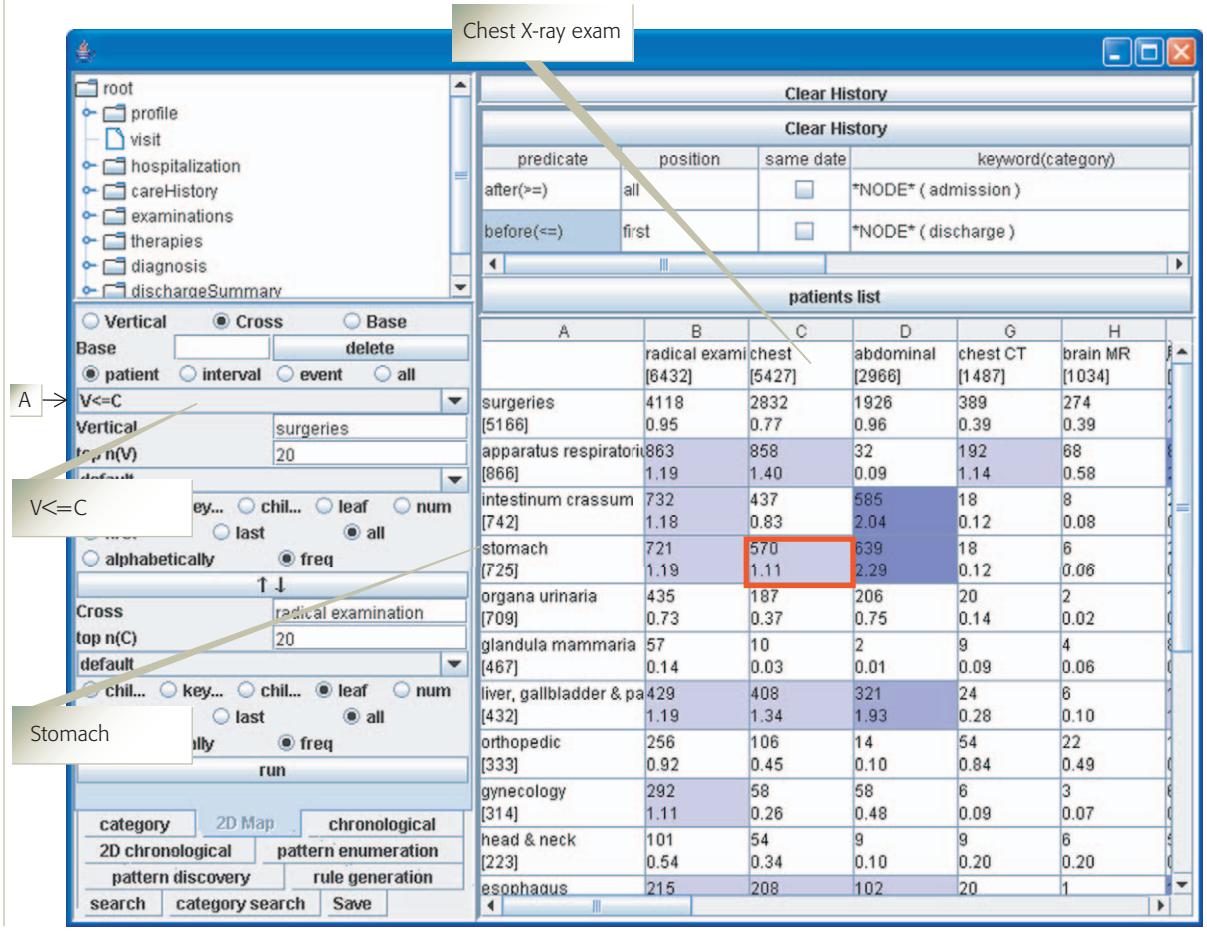

Figure 5
Two-dimensional map viewer

time of 4.51 days after their admission and were released from the hospital in an average of 25.63 days after their admission.

Rule generation

There are two functions to discover rules. The first function clusters members with higher objective variables into one group and members with lower objective variables into another group. By using this function, a physician may be able to discover the cause of prolonged hospitalizations. A pattern P to describe the group is evaluated by a measure of interclass variance defined as

$$ICV(P) = |S_1|(\overline{S_1} - \overline{S})^2 + |S_2|(\overline{S_2} - \overline{S})^2,$$

where S is a set of patients and the set S is divided into S_1 that satisfies P and S_2 that does not satisfy P, $|S_i|$ is the number of patients in group i, and $\overline{S_i}$ is the mean of the objective variables.[15] For example, one result shows that the average number of hospital days for the 42 patients who have two surgical operations in the orthopedic surgery group is 92.19, although the average hospital stay is about 17.59 days.

The second rule-discovery function is a rule generator based on a traditional decision tree.[16] Because most medical records of laboratory tests are time series data, it is not easy to feed the data to a decision tree. Therefore, when a physician selects one event as a base date, MedTAKMI-CDI interactively collects explanatory variables from the medical records of the laboratory tests that are the newest records before the selected base date. Objective variables are also interactively selected from the records after the base date. Because

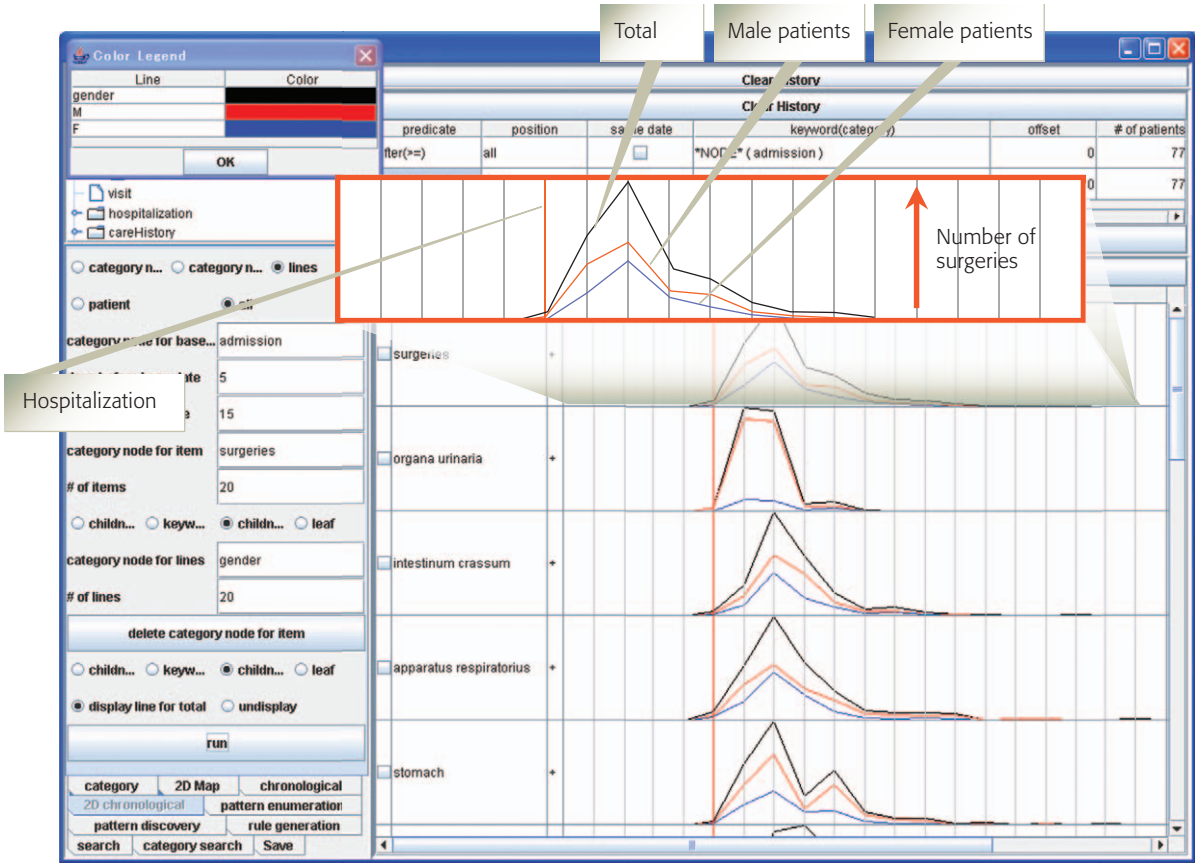

Figure 6
Two-dimensional chronological viewer

MedTAKMI-CDI supports selecting various objective variables, the heads of the discovered rules may be sequences of events. (The head of a rule corresponds to the consequence of a rule: If a rule is represented as A => B, then the head of the rule is B.)

Other functions

MedTAKMI-CDI has other auxiliary functions. For example, all of the results from these functions can be exported as comma-separated values (CSV) files. All parameters, including conditions used to narrow down the numbers of patients, can be saved. When data in MedTAKMI-CDI is updated on a daily or weekly basis, the physician can analyze the newly updated data by using the saved parameters. Because a category tree may have more than 25,000 nodes and the `EVENT` table can have more than 280,000 distinct values, it may not be easy to find the category node that a physician would like to analyze or a category node referred to by a preorder of a value that he or she would like to use; therefore, MedTAKMI-CDI supports a search function to search for category nodes and values.

USE SCENARIO

A typical scenario using MedTAKMI-CDI consists of the following steps:

1. *Deciding on a point of clinical care practice to study*—For example, asking, "What is the actual clinical practice for a specific group of patients suffering from a certain disease?" This will be the basis to decide on hypotheses about best practices, deviations from standard care, and potential factors influencing the outcomes.
2. *Interactive analysis of a collection of clinical records*—After a physician has decided to focus on a particular disease and a group of patients, MedTAKMI-CDI provides a suite of analytical

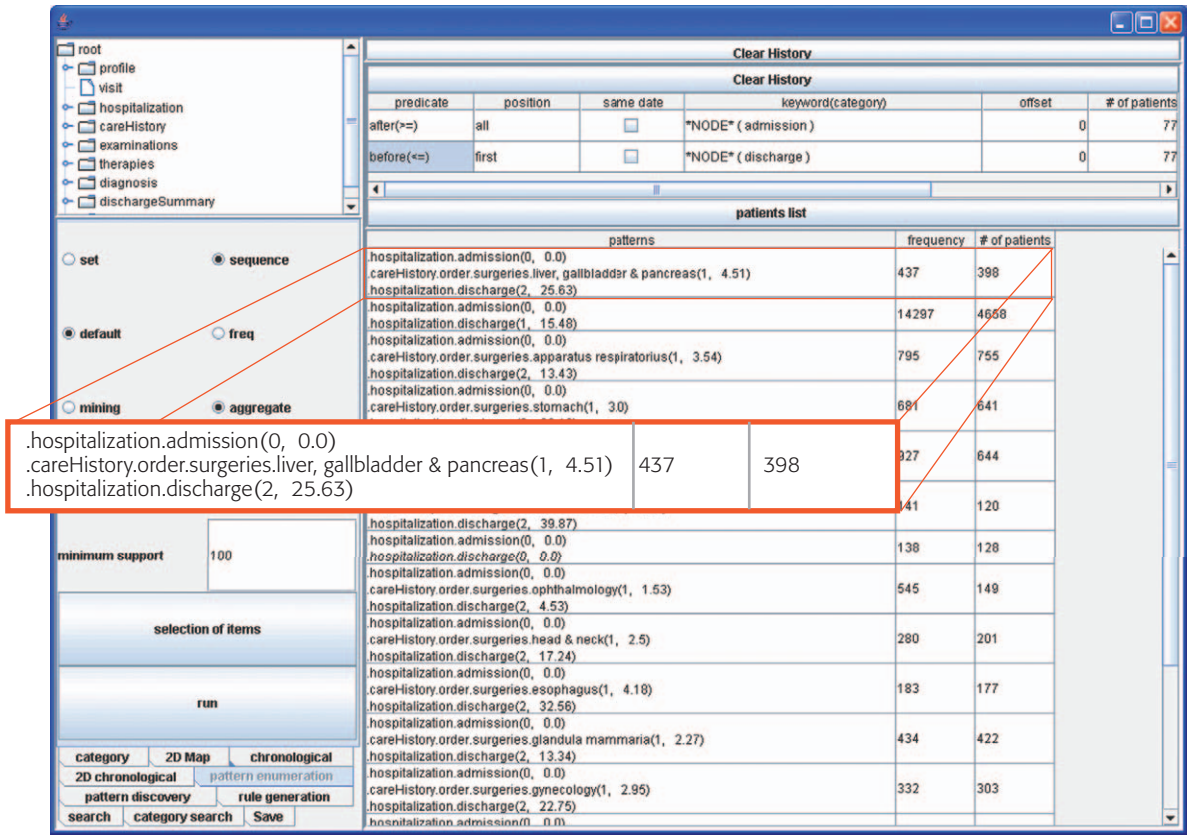

Figure 7
Pattern enumeration

functions that can be used to test and verify the hypotheses and to gain insight.

3. *Extraction of clinical care patterns and predictive rules*—After the group of patients has been identified, MedTAKMI-CDI can generate a list of clinical care patterns that are dominant among these patients. It can also generate a list of predictive rules about a particular attribute or attributes that are observed in the group of patients. This type of knowledge can be further verified and refined to define the standard of clinical care.

Figure 6 shows a distribution of surgical operations in terms of time line, gender, and organ. The span of the time line starts five days before hospital admission and ends 15 days after discharge, which is specified in the menu on the left side of the screen. We can restrict the data to only the first hospital admission for each patient, but the view in Figure 6 shows each hospital admission-event and discharge-event pair for each patient as a separate entity. The number of surgical operations each day for male and female patients is shown by the red and blue plot lines, respectively. The total number of surgical operations is shown by the black plot line. Each row of the view corresponds to a surgical operation for a particular organ. We can see, for example, that when the surgical operation takes place within a few days after hospitalization, the required examinations and diagnosis have been performed on the patients as outpatients, making it possible for patients to have surgery shortly after their hospital admission. Stomach cancer surgery, however, shows two peaks in the graph—three days and five days after admission. If the second peak of stomach cancer surgery is caused by a delay because of missing or duplicated laboratory tests, then it could be addressed by improving the outpatient clinical treatment process. Because many of the categories

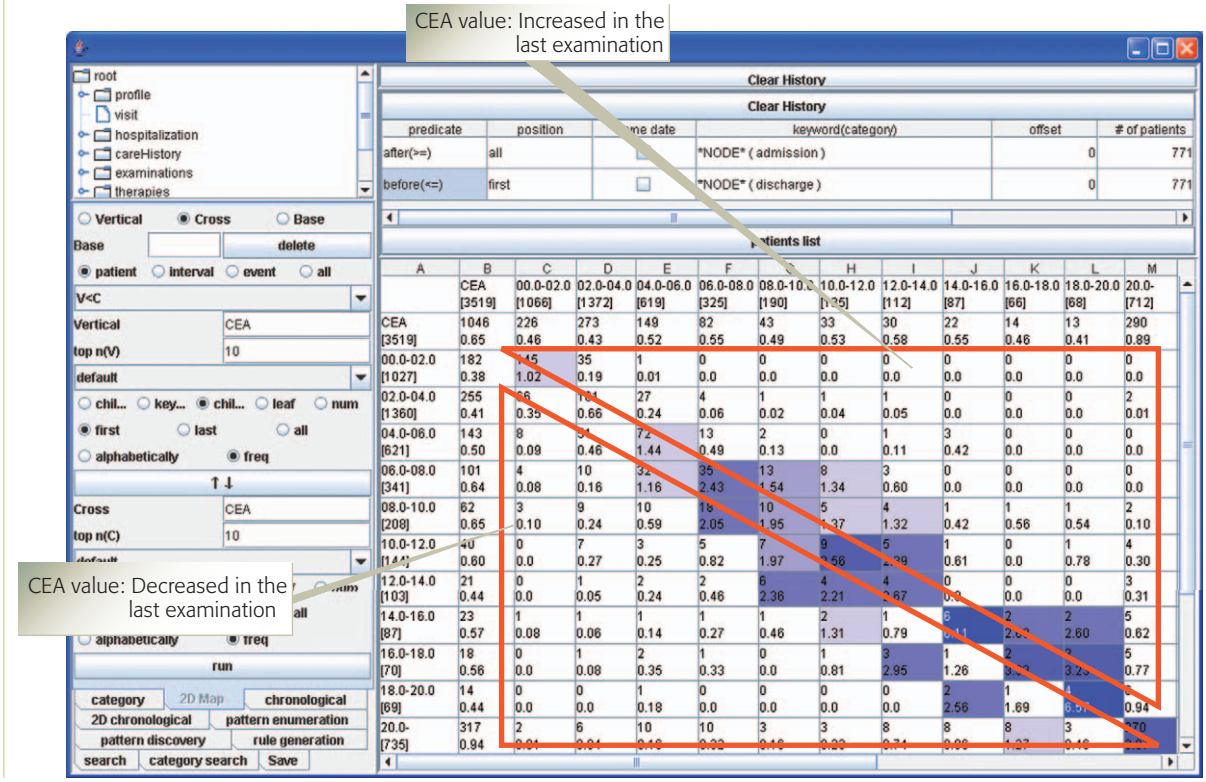

Figure 8
Association between CEA value ranges at hospital admission and discharge

for such analyses are defined hierarchically based on a standard or local ontology (for example, icd10 or SNOMED CT**[17]), physicians can work seamlessly from coarse-grained to fine-grained concepts for clinical analysis.

As an example of the analytic functions, ***Figure 8*** shows the two-dimensional map view that captures the self-correlation between the CEA (carcinoembryonic antigen) values of pancreatic cancer patients at hospital admission and discharge. CEA is known as one of the biomarkers for certain cancers. The two-dimensional map view in Figure 8 shows the CEA value ranges at hospital admission in the vertical axis and the CEA value ranges at hospital discharge in the horizontal axis. The blue cells in the view indicate strong correlations between two CEA value ranges. Most of the diagonal blue cells in the view imply that the patients' CEA values remain unchanged at hospital admission and discharge, but a significant number of blue cells in the area indicated by the lower left red triangle (with the darkest blue indicating the strongest correlation) indicate that the patients' CEA values did improve after hospitalization. Further analysis of those patients with improved CEA values and/or other tumor-marker values could lead to better clinical care for pancreatic cancers.

A clinical care pattern is defined as a sequence of more than one event for a particular patient identified from clinical records. A physician can specify certain types of events that constitute a sequence. In Figure 7, five types of events—hospital admission, discharge, surgical operation, endoscopic therapy, and radiation therapy—are selected for pattern enumeration. Most of the extracted patterns are simple ones, such as from admission to surgical operation to discharge, but we can observe that there are cases when two or more surgical operations and endoscopic therapies take place. We expect that fine-grained analysis of enumerated patterns would reveal dependencies of the outcomes of clinical care upon particular subsequences of events in some cases. ***Figure 9*** shows one instance of a predictive rule, showing that among 3,966

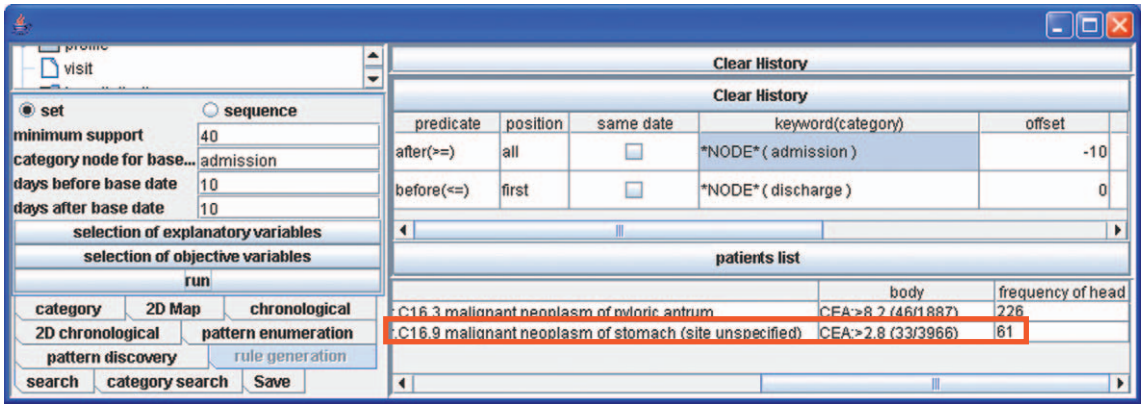

Figure 9
Generation of predictive rules

patients with CEA values higher than 2.8, 33 of them suffered from a malignant neoplasm of the stomach. Rule generation is tightly connected with the underlying event modeling and time-line analysis. Naïve rule generation might easily mislead physicians by concluding there is a disease based on biomarker values from after or from too long before the development of the disease. Therefore, MedTAKMI-CDI asks for specifications of the spans of the analysis (as given in Figure 6), together with explanatory variables.

DISCUSSION AND RELATED WORK

It is critical to achieve query response times that are as fast as possible when interactively analyzing a very large number of medical records. A key strategy to speeding up the aggregation of the records is to use indexing technology. As mentioned earlier in the section "Implementation," we use the preorders and postorders to check ancestor-descendant containment in a category tree. Although the method was proposed in 1982, it recently drew attention as the method to index Extensible Markup Language (XML) database data and to map XML data into a relational database,[18] as each XML document is modeled as a Document Object Model (DOM) tree. Several methods—such as prefix label,[19] Dewey order,[20] prime label,[21] VLEI code,[22] and embedding into a k-ary tree[23]—are used to index XML. A disadvantage of such methods as preorder-postorder and prime label is that they need the reassignment of preorders and postorders of nodes when inserting some nodes into a tree. As each node has the same label as a prefix of its children in such methods as prefix label and Dewey order, they do not need to reassign the labels when inserting some nodes. However, because they need to compute functions to process strings to check ancestor-descendant containment, they need more computation time than the preorder-postorder method. Because we assume that a category tree is rarely updated and records in the table `EVENT` are often inserted, we use the preorder-postorder method to index the category tree.

The performance evaluation is described in Reference 8. Although the evaluation was conducted by using biomedical documents, the model and implementation used in the evaluation are the same as those in the work discussed in this paper. In the evaluation, we selected 503,989 abstracts from MEDLINE**[24] which contain structured information, such as authors and mesh terms, and unstructured information, such as titles and abstracts. After preprocessing, the numbers of annotated keywords that correspond to records in the table `EVENT`, categories that correspond to records in the table `CATEGORY`, and distinct $c : v$ were 193,185,919, 340,154, and 1,4331,595, respectively. In most cases, the results for various queries appeared within one second. Although we needed a few minutes to get results for some queries in the worst case, we could reduce the response time by dividing the table `EVENT` into multiple tables that did not contain identical patient (document) identifiers.[8]

Levene and Loizou[25] is an example of the many papers that mention the design of star and snowflake schemas as methods to represent a dimension hierarchy (paths on an ontology). In these papers there is no discussion of interactive analysis specifying arbitrary intervals, although theories of dependencies, normalizations, and summarizability are discussed. Some methods need to reconstruct databases with every change and redesign the analytical attributes. For example, it is often difficult to reconstruct the existing data warehouse after any change of analyzed data and attributes.[26] To the best of our knowledge, few research papers that apply complex clinical records and genomic data to multidimensional analysis are published.[1,27,28] Recently, BioStar, which has the properties of extensibility and flexibility that are applicable to clinical records and genomic data, was proposed.[28] BioStar stores complex data containing many-to-many relationships by introducing tables, called *m-tables*, which associate between a central fact table and each dimension table. The schema is based on the entity-relationship approach. In contrast, the method proposed in this paper employs a metaschema with patient identifier, time stamp, attribute name, and attribute values, rather than a collection of rigid relational schema for clinical information. The advantages of our method are the capabilities to interactively specify arbitrary intervals, to easily integrate some ontologies that are not balanced tree, and to achieve fast response time without precomputation.

CONCLUSION

In this paper, we described our approach to building the CDI solution. The proposed system, called MedTAKMI-CDI, employs metaschema and uses ancestor-descendant containment by preorders and postorders to achieve fast query response times. MedTAKMI-CDI provides various functions to analyze medical records in an interactive manner. MedTAKMI-CDI is still under intense development and reflects many of the requirements of physicians and hospital administrators. One of the fundamental requirements is to incorporate time-stamped events explicitly in pattern enumeration and rule generation. This would be easily perceived as showing that a set of events can be used as an abstraction of some sequence of events. The former can abstract all the permutations of event sequences observed in the latter. Similarly, a simple sequence of events can generalize all of the time-stamped sequences of events by ignoring the time intervals between the adjacent events in the sequence. The latter could be used to provide mission-critical analysis of particular clinical treatments.

ACKNOWLEDGMENTS

We thank Professor Yasutomi Kinosada of Gifu University, Dr. Takao Umemoto of Chikaishi Hospital, Mr. Susumu Suzuki, Dr. Naohiko Uramoto, Mr. Hironori Takeuchi, Ms. Akiko Kishiro of IBM Japan, and Ms. Mariko Adachi of IBM Business Consulting Service for their help and advice.

CITED REFERENCES

1. T. Bach Pedersen and C. S. Jensen, "Multidimensional Data Modeling for Complex Data," *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, Sydney Austrialia (1999), pp. 336–345.
2. T. Bach Pedersen and Christian S. Jensen, "Multidimensional Database Technology," *IEEE Computer* **34**, No. 12, 40–46 (2001).
3. DB2 Business Intelligence, Hierarchies, IBM Corporation, http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.db2_olap.doc/cmdhierarchy.htm
4. International Classification of Diseases (ICD-10), Tenth Revision, 1990, World Health Organization, http://www.who.int/classifications/icd/en/.
5. T. Nasukawa and T. Nagano, "Text Analysis and Knowledge Mining System," *IBM Systems Journal* **40**, No. 4, 967–984 (2001).
6. N. Uramoto, H. Matsuzawa, T. Nagano, A. Murakami, H. Takeuchi, and K. Takeda, "A Text-Mining System for Knowledge Discovery from Biomedical Documents," *IBM Systems Journal* **43**, No. 3, 516–533 (2004).
7. P. Bonatti, Y. Deng, and V. S. Subrahmanian, "An Ontology-Extended Relational Algebra," *Proceedings of the IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV (2003), pp. 192–199.
8. A. Inokuchi and K. Takeda, *Online Analytical Processing of Text Data* (in Japanese), Research Report, RT-0670, IBM Research Division, Tokyo Research Laboratory, Yamato, Kanagawa, Japan (2006), http://www.geocities.jp/inokuchiresearch/RT0670.pdf.
9. P. F. Dietz, "Maintaining Order in a Linked List," *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, San Francisco, CA (1982), pp. 122–127.
10. A. Inokuchi and K. Takeda, *SQL-Based Aggregation for Text Mining* (in Japanese), Research Report RT-0634, IBM Research Division, Tokyo Research Laboratory, Yamato, Kanagawa, Japan (2005), http://www.geocities.jp/inokuchiresearch/RT0634.pdf.

11. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Santiago de Chile, Chile (1994), pp. 487–499.
12. R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, Taipei, Taiwan (1995), pp. 3–14.
13. R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*, Zurich, Switzerland (1995), pp. 407–419.
14. R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*, Avignon, France (1996), pp. 3–17.
15. Y. Morimoto, H. Ishii, and S. Morishita, "Efficient Construction of Regression Trees with Range and Region Splitting," *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Athens, Greece (1997), pp. 166–175.
16. J. R. Quinlan, "Induction of Decision Trees," *Machine Learning* **1**, No. 1, pp. 81–106 (1986).
17. SNOMED CT, SNOMED International, http://www.snomed.org/snomedct/.
18. T. Grust, "Accelerating XPath Location Steps," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Madison, WI (2002), pp. 109–120.
19. E. Cohen, H. Kaplan, and T. Milo, "Labeling Dynamic XML Trees," *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Madison, WI (2002), pp. 271–281.
20. I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang, "Storing and Querying Ordered XML Using a Relational Database System," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Madison, WI (2002), pp. 204–215.
21. X. Wu, M.-L. Lee, and W. Hsu, "A Prime Number Labeling Scheme for Dynamic Ordered XML Trees," *Proceedings of the 20th International Conference on Data Engineering*, Boston MA (2004), pp. 66–78.
22. K. Kobayashi, W. Liang, D. Kobayashi, A. Watanabe, and H. Yokota, "VLEI Code: An Efficient Labeling Method for Handling XML Documents in an RDB," *Proceedings of the International Conference on Data Engineering*, Tokyo, Japan (2005), pp. 386–387.
23. Y. K. Lee, S.-J. Yoo, K. Yoon, and P. B. Berra, "Index Structures for Structured Documents," *Proceedings of the 1st ACM International Conference on Digital Libraries*, Bethesda, MD (1996), pp. 91–99.
24. MEDLINE, U.S. National Library of Medicine, http://www.nlm.nih.gov/databases/databases_medline.html.
25. M. Levene and G. Loizou, "Why is the Snowflake Schema a Good Data Warehouse Design?" *Information Systems* **28**, No. 3, 225–240 (2003).
26. T. Critchlow, M. Ganesh, and R. Musick, "Automatic Generation of Warehouse Mediators Using an Ontology Engine," *Proceedings of the 5th International Workshop on Knowledge Representation Meets Databases*, Seattle, WA (1998), pp. 8.1–8.8.
27. T. Bach Pedersen and C. S. Jensen, "Research Issues in Clinical Data Warehousing," *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, Capri, Italy (1998), pp. 43–52.
28. L. Wang, A. Zhang, and M. Ramanathan, "BioStar Models of Clinical and Genomic Data for Biomedical Data Warehouse Design," *International Journal of Bioinformatics Research and Applications* **1**, No. 1, 63–80 (2005).

Accepted for publication July 19, 2006.
Published online January 9, 2007.

Akihiro Inokuchi
IBM Research Division, Tokyo Research Laboratory, 1623-14, Shimotsuruma, Yamato, Kanagawa, Japan (inokuchi@jp.ibm.com). Dr. Inokuchi joined the IBM Tokyo Research Laboratory in 2000 after receiving an M.S. degree in communication engineering from Osaka University. He received a Ph.D. degree in communication engineering from Osaka University in 2004. Dr. Inokuchi's research interests include data mining, machine learning, text mining, OLAP, data warehouse, and medical informatics.

Koichi Takeda
IBM Research Division, Tokyo Research Laboratory, 1623-14, Shimotsuruma, Yamato, Kanagawa, Japan (takedasu@jp.ibm.com). Mr. Takeda joined the IBM Tokyo Research Laboratory in 1983 after receiving an M.E. degree in information science from Kyoto University. Mr. Takeda's research interests include text analytics, information retrieval, and medical informatics.

Noriko Inaoka
IBM Business Consulting Services, 2-4-1, Marunouchi, Chiyoda-ku, Tokyo, Japan (inaoka@jp.ibm.com). Dr. Inaoka is an associate partner of Healthcare Industry, IBM Business Consulting Services. She joined the IBM Tokyo Research Laboratory after receiving a Ph.D. degree in health sciences from the University of Tokyo in 1983. She has worked in the areas of knowledge engineering and medical image recognition. Dr. Inaoka is currently focusing on health-care business performance transformation based on technology.

Fumihiko Wakao
National Cancer Center Hospital, 5-1-1 Tsukiji, Chuo-ku, Tokyo, 104-0045, Japan (fwakao@ncc.go.jp). Dr. Wakao is head of Diagnostic Radiology. As a vice-chairman of the National Cancer Center (NCC) Information Committee since 1996, Dr. Wakao has played a central role in the Information System Project of the new NCC facility, building the Radiological Information System and Image Reference System. He graduated from the Yokohama City University School of Medicine in 1986, and was a resident in the Radiological Diagnosis Division at the NCC. Since 1993, Dr. Wakao has constructed an image database and has worked on three-dimensional image processing, virtual reality, and telemedicine as a member of the Total Support System for Cancer Diagnosis and Therapy integrated team. ■