Study of Different Types of Queries in Sequential Multi Dimensions Database

Pradeep Kumar

Research Scholar, CMJ University Shillong, Meghalaya

ABSTRACT: A wide range of database applications manage time-varying data. In contrast, existing database technology provides little support for managing such data. The research area of temporal databases aims to change this state of affairs by characterizing the semantics of temporal data and providing expressive and efficient ways to model, store, and query temporal data. It concisely introduces fundamental temporal database concepts, surveys state-of-the-art solutions to challenging aspects of temporal data management. Applications such as these rely on *temporal databases*, which record time referenced data. Temporal database management is a vibrant field of research. Temporal Database manipulations are effective based on life span time. This paper illustrates the graphical representation of the temporal database manipulations in terms of visual query with LSA. The Life Span Analyzer (LSA) provides the effective results on temporal manipulations It also explains the semantics, Patterns and Visual Query Operators.

----------♦----------

1. INTRODUCTION

There are three main classes of query languages devoted to spatial databases: Textual languages (natural, SQL and extensions), Graphical languages (QBE) and Visual languages. Since these langue's are not sufficient to work with temporal databases [4]. The updations are made with the existing visual query languages to manipulate with the temporal databases. Visual query with temporal database provides graphical representation of query analysis and performance of a query in temporal databases.

In various fields there is a need to manage geometric, geographic, or spatial data, which means data related to space. The space of interest can be, for example, the two-dimensional abstraction of (parts of) the surface of the earth – that is, geographic space, the most prominent example –, a man-made space like the layout of a VLSI design, a volume containing a model of the human brain, or another 3d-space representing the arrangement of chains of protein molecules.

Characteristic for the technology emerging to address these needs is the capability to deal with large collections of relatively simple geometric objects, for example, a set of 100 000 polygons. This is somewhat different from areas like CAD databases (solid modeling etc.) where geometric entities are composed hierarchically into complex structures, although the issues are certainly related [2]. Several terms have been used for database systems offering such support like pictorial, image, geometric, geographic, or spatial database system. The terms "pictorial" and "image" database system arise from the fact that the data to be managed are often initially captured in the form of digital raster images (e.g. remote sensing by satellites, or computer tomography in medical applications).

Image database systems may include analysis techniques to extract objects in space from images, and offer some spatial database functionality, but are also prepared to store, manipulate and retrieve raster images as discrete entities. In this survey we only discuss spatial database systems in the restricted sense. So the spatial database or a simple image database system is not sufficient in this case. Temporal database provides several patterns to access the images though the databases and the visual queries provide the effective timings and prominent accessing and retrival of patterns.

2. TEMPORAL DATABASES AND VISUAL QUERYIES

Most applications of database technology are temporal in nature. Examples include financial applications, recordkeeping applications such as personnel, medical-record, scheduling applications such as airline, train, and hotel reservations and project management and scientific applications such as weather monitoring [1]. Temporal database management is a vibrant field of research, with

an active community of several hundred researchers who have produced some 2000 papers over the last two decades [7].

2.1 Temporal Data Semantics:

Before considering temporal data models and query languages, we examine, in data model-independent terms, the association of times and facts, which is at the core of temporal data management.

The Main Goals of Temporal Database:

- Identification of an appropriate data type for time
- Prevent fragmentation of an object description
- Provide query algebra to deal with temporal data
- Compatiable with old database without temporal data

The ***transaction time*** of a database fact is the time when the fact is current in the database. Unlike valid time, transaction time may be associated with any database entity, not only with facts [3]. For example, transaction time may be associated with objects and values that are not facts because they cannot be true or false in isolation [6].

Thus, all database entities have a transaction-time aspect. Based on the lifespan of the system, we could use [t1, t2] to repersent the valid time of the data model and use mathematic "SET ALGEBRA" to operate them. When we want to handle temporal data, we define is the set of all historical domains.

D = {D1,D2,.....Dn}

T = universal time lifespan

TD = {TD1, TD2, TD3,...TDn},

where for each i , TDi = {fi|fi:T->Di} is the set of all partial function from T into the value domain Di.

TT = {g|g:T->T} is the set of all partial functions from T into itself.

U={A1, A2,.....,An} be a (universal) set of attributes.

All attributes in the historical relational data model are defined iver sets of parial temporal function.

HD = (TD or {TT}) = { TT, TD1, TD2....TDn }

A relation scheme R = is an ordered 4-tuple where.

- A = {Ar1, Ar2 ...Ar3}(U is the set of attributes od R.we will sometimes abuse notation and refer to A as the scheme of R; no confusion should arise.
- K = {Ak1, Ak2,.....Akm}(A is the set of (primary) key attributes of R
- ALS:A X R -> 2^T is a function assigning a lifespan to each attribute in A in scheme R. We will refer to the lifespan of attribute A in relation scheme R as ALS(A,R).
- DOM:A->HD is a function assigning a domain to each attribute in R, with the restrictions that (1) for all key attributes Ai,DOM(Ai) < CD, that is the key attributes must all be constant-valued; and (2) the temporal domain of each of the partial function in any DOM(A) is contained within ALS(A,R).

Figure 1 gives the relation instance in the Bitemporal Conceptual Data Model(BCDM) [9] that describes the sample rental scenario. This data model time stamps tuples, corresponding to facts, with values that are sets of (transaction time, valid time) pairs, captured using attribute T in the figure.

Customer ID	TapeNum	T
C101	T1234	{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3) … (UC, 2), (UC, 3) (UC, 4)}
C102	T1245	{(3, 2), (3, 3), (3, 4), (4, 2), (4, 3) … (UC, 4), (UC, 5) (UC, 6)}
C102	T1234	{(5, 2), (5, 3), (5, 4), (6, 2), (6, 3) … (UC, 2), (UC, 3)… (UC,9)}

Figure 1: Bitemporal Conceptual CheckedOut Instance

The presence of a pair (*tt*, *vt*) in a timestamp of a tuple means that the current state of the database at time *tt* records that the fact represented by the tuple is valid at time *vt*.

The special value UC ("until changed") serves as a marker indicating that its associated facts remain part of the current database state, and the presence of this value results in new time pairs being included into the sets of pairs at each clock tick. The idea behind the BCDM is to retain the simplicity of the relational model while also capturing the temporal aspects of the facts stored in a database. Because no two tuples with mutually identical explicit attribute values (termed *valueequivalent*) are allowed in a BCDM relation instance.

3. VISUAL QUERIES WITH RELATIONAL DATABASES

Visual query languages moved beyond QBE's tabular entry by providing users with a more accurate view of the database structure by introducing Extended Entity-Relationship (EER) diagrams. EER diagrams are an effective method for modeling the structure of information stored in a relational database, and can be converted directly into relational tables. It is not surprising therefore, that both tables and EER diagrams have been explored as the basis for database query. Users of Microsoft's Access database, for example, have been exposed to one example of tablebased query specification. Indeed, the introduction of visual query languages has dramatically affected database query.

3.1 Temporal Patterns

We consider a temporal pattern as a sequence of Events and inter-event TimeSpans that can be restricted in some way. The fundamental building blocks for a pattern are therefore Events, and between Events, TimeSpans. A pattern is a sequence of Events and TimeSpans of arbitrary length, as shown in the following Figure:

Figure : A pattern as a sequence of Events and TimeSpans.

3.2 Example For Visual Query Operators

Visual Query Operators may acts as a tool for manipulating real time examples. Here we have choosen geographical data bases for database manipulation with visual queries. Visual queries can be composed to manipulate these datas along with temporal databases. Large volumes of federal data (together with their associated geospatial properties) are being collected and managed with database systems. Accessing these databases with command-line-based query languages is difficult, error-prone, and tedious. This system concentrates on enabling non-specialist users to access the system easily. Here we are using GIScience to facilitate interaction with GIS and in database and application fields to develop visual query systems (VQS) that make it easier for users to compose queries. The Geographic Visual Query Composer (GVQC) [5] [7] can be viewed as a tool that allows users to visually formulate and execute a query statement directed to federal databases containing geospatial data. GVQC also allows users toget/edit textbased SQL statements, which are automatically extracted from the visual design.

3.3 Query Processing

A query formulated in some high-level, user-oriented query language is typically translated into an equivalent query, formulated in a DBMS-internal, algebraic query language. The DBMS then optimizes this algebraic expression by transforming it into an equivalent expression that is

expected to be more efficient to process, the result being better query processing performance. Optimization of temporal queries offers new challenges over optimization of conventional queries. At the core of the matter, temporal database queries are often large and complex [10]. Because of this added complexity, it is not only more important, but also more challenging, to optimize temporal database queries. Specifically, the predicates used in temporal queries make these queries difficult to optimize. In nontemporal database applications, predicates are often equality predicates. As a reflection of this, much research in query processing has concentrated on equality predicates, and existing DBMSs are optimized for equality predicates (which occur in, e.g., equi-joins and natural joins). In contrast, temporal queries typically involve numerous inequality predicates.

The perhaps most prominent source of such predicates is the test of overlap among two intervals. Inherent in temporal joins, this test occurs frequently in temporal queries and results in two equality predicates. Specifically, two intervals *i* and *j* overlap if the begin value of *i* is less than or equal to the end value of *j* and the begin value of *j* is less than or equal to the end value of *i*. Conventional DBMSs typically resort to nested-loop implementations of joins involving such inequality predicates, with their associated inefficiency. There are new and unexploited opportunities for query optimization when time is present. The current time advances continuously; and for transaction time, the time value used most recently in updates is the largest value used so far.

As another example of an optimization opportunity, the integrity constraint that the begin value of an interval is less than or equal to its end value holds for all intervals in the database [4]. Next, for many relations, the intervals associated with a key value are contiguous in time, with one interval starting exactly when the previous interval ended. Semantic query optimization can exploit these integrity constraints, as well as additional ones that can be inferred.

A wide variety of binary joins have been considered, including *time-join* and *time-equijoin* (TE-join), *event-join* and *TE-outer join*, *contain-join*, *contain semi join* and *intersect-join*, and *temporal natural join* (e.g., [7, 9]). The various algorithms proposed for these joins have generally been extensions to nested loop or merge joins that exploit sort orders or local workspace, as well as partitioning-based joins, but incremental techniques have also been proposed.

3.4 Taxonomy Of Query Modules

GVQC views a complex query as a hierarchical composition of simple queries. GVQC consists of a finite set of query designs, each of which will either formulate a simple query or process input from simple queries to formulate a composite query. An atomic query is a query that cannot be decomposed into simpler ones [12]. This GVQC formulate an atomic query to manipulate complex query designs. According to the data type we used in the system this composer will analyse the data for the manipulations on temporal database.

3.5 Formulating And Configuring Queries

The system can get inputs from a numeric range query design, which represents an atomic query. This query returns the numeric information's for the manipulations.

4. VISUAL QUERY BY TIME INTERVAL

Chittaro and Combi have proposed three alternative visual metaphors for querying temporal intervals [11]. The authors based the expressivity of their visual language on Allen's classification of the relations that may hold between two intervals. Three semantically equivalent representations (elastic bands, springs and paint strips) depict horizontal bars whose ends can be constrained in such a way as to capture all 13 of Allen's interval relationships.

The results of controlled user studies indicated that users were better able to guess the meaning of queries than formulate their own. Posing queries to find complex patterns is without question a complex task, but it may be that the interval representation did not support realistic user tasks, or that it did not match the user's mental model of the for query formulation.

Forms-based direct manipulations are introduced to represent Temporal Visual Query Language (TVQL) for specifying interval endpoint constraints [10] to support Allen's 13 relational primitives. Four double-sided sliders allow users to express the relationship between each pair of endpoints among two intervals. Although users interact exclusively with the sliders, a visual representation of the interval interaction is dynamically updated to provide the user feedback on the meaning of the query defined. Our approach obviously differs from these interval approaches insofar as we query events fundamentally as points in time.

Although these proposed interval approaches enable expression of relationships, they are too succinct to allow users to filter results by types or value ranges or to specify absolute time ranges as does ours. We feel there needs to be a combination of both the power of visualizing the relationships, while still being able to express specific event and time constraints. The Query Analyzer Tool which deals time intervals effectively.

4.1 Visualizing Temporal Patterns

Interestingly, none of the previously-described systems address the visualization of the returned results, but instead focus only on the query. A query paradigm is necessary; however, visualizing the results is essential for understanding the underlying data and provides a feedback loop to help users more thoroughly understand the query interface itself.

Various applications such as TimeSearcher [11], Spirals [10], DataJewel [3], KNAVE [7] and LifeLines [6] have been proposed to visualize temporal abstractions by clustering results and emphasizing temporal patterns in the returned results.

4.2 Time Visualizations

TimeSearcher and Spirals attempt to visualize time-series. TimeSearcher is a flexible tool that allows users to explore the data by specifying queries using "TimeBoxes". TimeBoxes are rectangular query locators that specify the region(s) in which the users are interested. This tool allows the user to explore patterns in the data or to find similar patterns. Weber et. al proposed visualizing time series on spirals. Each ring of the spiral represents a periodic section of time series. Color and line thickness are used to distinguish the data values. Although an interesting approach, it is only compatible when dealing with events that happen periodically.

DataJewel is an excellent example of an application that bridges information visualization with data mining. For each day, the frequency of the events is displayed using horizontal histograms. Although the DataJewel algorithms are extensible, the visualization parameters are somewhat restrictive: a maximum of 10 event attributes can be displayed at a time, and all time events are represented at the day granularity.

The CalendarView is primarily intended as a compact representation for visually detecting patterns coupled with basic browsing capabilities: users may select a subset of days for display, can order events by frequency, rescale the view, and access details on demand. We instead provide ad hoc query for temporal patterns.

For Example we can view the importance of visual queries in medical field. Viewing, exploring, and analyzing data collected over time presents a challenge in both the database and information visualization fields. The amount of data that is collected over time becomes unmanageable if there are no techniques to cluster, analyze and visualize them. Most medical databases use time-stamped instant data as the only temporal representation of patient information. Many previous efforts have attempted to provide frameworks in which medical databases could be queried in relation to time. These, however, have required either a sophisticated database representation of time, including time intervals, or a time-stamp-based database coupled with a nonstandard temporal query language In the medical field, final outcomes are important, but recognizing patterns over time is essential to adapting treatment plans and understanding complex interactions. For example, monitoring disease evolution and drug response over time are typical progress evaluation tasks. However, if time dependent events are collected from different sources, joining these databases becomes a challenge as the granularity of time capture may differ across databases. Additionally, there exist strong relationships between consecutive time-stamped observations, with establishes a context for disease evaluation. Common temporal confounds in medicine include delays between source causes and observables, as well as the distortion of temporal relationships between variables due to concurrent processes (e.g., multiple drug treatment).

Furthermore, in clinical domains, a final diagnosis is not always the main goal. What is often needed is a coherent intermediate-level interpretation of the relationships among data and events.

We can also see the Patterns in Health Care. Diverse data for individuals are collected over long periods of time, and then analyzed and interpreted at many levels. Patients may be assessed individually by nurses and physicians, but may also be analyzed en masse by clinical researchers, public health officials, auditors, etc. Collectively, the potential size of the data set is staggering, and issuing queries and interpreting results is a real challenge. Visualizing temporal data and the patterns within helps different types of end users interpret the data and make decisions.

Numerous applications in the medical field attempt to make the interpretation and analysis of the temporal data easier and faster by visualizing, grouping or abstracting the data. LifeLines was one of the first such systems, which provided a compact hierarchical timeline visualization for personal histories [6]. In both the medical and legal domains (for which LifeLines has been applied) history is the key factor in authoritative decision making. Proving an overview for large datasets has been found beneficial to such decision makers and LifeLines' multi-faceted approach allows for a compact, single-screen overview.

Users then navigate history details by selecting specific facets for expansion, zooming, adjusting the time scale, filtering records and accessing details on demand. LifeLines supports both discrete time events, displayed as icons, and interval events, displayed as lines. Line thickness and color encode event attributes such as significance and

relationship to other events. Lifeline primarily supports directed browsing with text search, but does not offer a higher-level query mechanism. Neither does it support discovery across multiple records, such as how many people had heart surgery on a specific date and what are the trends of events surrounding those surgeries.

5. AN EXAMPLE QUERY

The example dataset covers the entire USA geospatial structure manipulation with visual query in temporal database. The system manipulates STATES, COUNTRIES, CITIES, ROADS, and LAKES. Each layer has a spatial geometry column (storing points, lines, or polygons), and many other non-spatial data columns (demographic and other census variables). A simple query scenario is detailed below to illustrate the GVQC [12]. The scenario is that the user is interested in the relative impact of main highways on jobs, income, and the general economy of places for blacks and whites. They start by looking at the number of mobile homes, black, and white populations of small towns or small to medium cities. One example query from a set of similar ones might be: for all cities with total population < 50,000, within the 4-state region of PA-WVMD- VA, and that are within 5 miles of a main highway—give me the city names, black population, white population, mobile homes of each city. The SQL (with Oracle spatial extensions) statement for this query is as follows:

```
select CITIES.NAME, CITIES.WHITE, CITIES.BLACK, CITIES.MOBILEHOME,
CITIES.POP1990,ROADS.NAME, CITIES.STATE_NAME from ROADS, CITIES
where ( ( CITIES.STATE_NAME IN ('Maryland', 'Pennsylvania', 'Virginia', 'West Virginia') AND
(CITIES.POP1990) <= 50000.0 ) AND SDO_WITHIN_DISTANCE(ROADS.GEOM,
CITIES.GEOM, 'distance=5') = 'TRUE')
```

With GVQC, the user can visually compose this query with just several mouse clicks without knowing either the SQL syntax or the detailed database schema. The query statement is automatically extracted from the visual design.

6. CONCLUSION

Advantages of this visual query system can be generalized as follows: (1) dynamic visualization of related database schema information to help the user explore the database and construct correct/accurate queries; (2) flexibility in and ease of forming complex spatial and/or non-spatial queries; (3) clear visualization of the semantic hierarchy of a complex query, which is useful for both forming and understanding such queries (4) the ease with which a query can be modified; (5) the ability to extract text-based SQL statements that allow users to read/learn the query language; (6) the extensibility of the system for adding new modules and new visualization capability to each module.

FUTURE ENHANCEMENT

The Life Span Analyzer (LSA) can be updated or enhanced for the future modifications. The Main Feature of the system is Life Span Time calculation and providing the visual results for the Query manipulations.

- The system can be updated for the best results on life span calculations by comparing the different queries along with the Query Analyzer Algorithms.
- The Artificial Intelligence (AI) Technique can be implemented along with LSA (Life Span Analyzer) algorithm to analyze the Life Span Time by Comparing different queries.
- The system can be enriched to sense the query processing time and analysis of visual queries by implementing in Neural Networks.

REFERENCES

[1] T. Abraham and J. F. Roddick. Survey of Spatio-Temporal Databases. GeoInformatica, (1):61–99,March 1999.

[2] Ahlberg, C. and Shneiderman, B., Visual Information Seeking: Tight coupling of dynamic query filters with starfield displays. in Proceedings of ACM's SIGCHI Conference, (1994), ACM Press.

[3] Allen, J.F. (1983) Maintaining knowledge about temporal intervals. Communications of the ACM, 26 (11).

[4] Ankerst, M., Jones, D.H., Kao, A. and Wang, C., DataJewel: Tightly Integrating Visualization with Temporal Data Mining. in ICDM Workshop on Visual Data Mining, (2003).

[5] Dionisio, J.D.N. and Cardenas, A.F. (1996) MQuery: a visual query language for multimedia timeline and simulation data. Journal of Visual Languages and Computing,

[6] Catarci, T., M. Costabile, et al. (1995). "Visual Query Systems for Databases: A Survey." Journal of Visual Languages and Computing.

[7] Egenhofer, M. and W. Kuhn (1999). Interacting with Geographic Information Systems. GeographicalInformation Systems: Principles, Techniques, Applications, and Management. D. Rhind, Wiley:New York.

[8] W. Kim (ed.) Modern Database Systems: The Object Model, Interoperability and Beyond. Addison-Wesley/ACMPress 1995.

[9] C. S. Jensen and R. T. Snodgrass. Semantics of Time-Varying Information. Information Systems, 21(4):311–352, 1996.

[10] R. T. Snodgrass. Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers 2000.

[11] Chittaro, L. and Combi, C. (2003) Visualizing queries on databases of temporal histories: new metaphors and their evaluation. Data & Knowledge Engineering, 44 (2). 239-264.

[12] http://en.wikipedia.org/wiki/ Temporal_database