© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

A REVIEW OF TEMPORAL DATAWAREHOUSE

Aastha Sharma, Dhruvika sharma
Student, Information Technology, Maharishi Dayanand University
Dronacharya College Of Engineering , Haryana,India

Abstract- **Data warehouses are become widespread these days, as they are specialized in taking decisions .Data warehouses have many applications. They have been rapidly spreading within the industrial world over the last decade, due to their undeniable contribution to increasing the effectiveness and efficiency of the decisional processes within business and scientific domains. Since, the process that requires analysis of historical trends, for it time and management acquire a huge importance. In this paper we consider the variety of issues, often grouped under term temporal data warehousing, implied by the need for accurately describing how information changes over time in data warehousing systems. We can recognize with reference to a three-levels architecture, these issues can be classified into some topics, namely: handling data changes in the data warehouse, handling data changes in the data mart, querying temporal data, and designing temporal data warehouses. So, these are the various issues that we are dealing in this research paper. After introducing the main concepts and terminology of temporal databases, we separately survey these topics. Finally, we discuss the open research issues also in connection with their implementation on commercial tools.**

Index Terms- **Communication Handling data changes in data mart, handling data changes in data warehouse, querying temporal data and its designing.**

I. INTRODUCTION

At the core of most business intelligence applications, data warehousing systems are specialized in supporting decision making. They have been rapidly spreading within the industrial world over the last decade, due to their undeniable contribution to increasing the effectiveness and efficiency of the decisional processes within business and scientific domains. This wide diffusion was supported by remarkable research results aimed at improving querying performance, at refining the quality of data, and at outlining the design process, as well as by the quick advancement of commercial tools.

In the remainder of the paper, for the sake of terminological consistency, we will refer to a classic architecture for data warehousing systems, relies on three levels:

- The data sources, that store the data used for feeding the data warehousing systems. They are mainly corporate operational databases, hosted by either relational, but in some cases they may also include external web data, flat files, spreadsheet files, etc.
- The data warehouse a normalized operational database that stores detailed, integrated, clean and consistent data extracted from data sources and properly processed by means of ETL tools.
- The data marts, where data taken from the data warehouse are summarized into relevant information for decision

making, in the form of intricate cubes, to be typically queried by OLAP and reporting front-ends.

Cubes are structured according to the multidimensional model, whose key concepts are fact, measure and dimension. A fact is a focus of interest for the decisional process, occurrences correspond to events that dynamically occur within the business world. Each event is quantitatively described by a set of numerical measures. In the intricate model, events are arranged within an n-dimensional space whose axes, called dimensions of analysis, define different perspectives for their identification. Dimensions commonly are discrete, alphanumerical attributes that determine the minimum granularity for analyzing facts. Each dimension is the root of a placing that includes a set of levels, each providing a way of selecting and aggregating events. Each level can be described by a set of properties.

As a consequence of the fact that the decisional process typically relies on calculating historical trends and on comparing snapshots of the enterprise taken at different moments, one of the main characterizations of data warehousing systems is that of storing historical, non volatile data. Thus, time and its management acquire a huge importance. In this paper we discuss the variety of issues, often grouped under term temporal data warehousing, implied by the need for accurately describing how information changes over time. These issues, arising by the never ending development of the application domains, are even more pressing today, as several mature implementations of data warehousing systems are fully operational within medium to large business contexts. Note that, in comparison with operational databases, temporal issues are more critical in data warehousing systems since queries frequently span long periods of time, thus, it is very common that they are required to cross the boundaries of different versions of data and/or schema. Besides, the criticality of the problem is obviously higher for systems that have been established for a long time, since unhandled developments will determine a stronger gap between the reality and its representation within the database, which will soon become obsolete and useless.

So, not surprisingly, there has been a lot of research so far regarding temporal issues in data warehousing systems. Basically, the approaches devised in the literature can be accommodated in the following categories:

- Handling changes in the data warehouse

  This mainly has to do with maintaining the data warehouse in sync with the data sources when changes on either of these two levels occur.

- Handling data changes in the data mart

  Events are continuously added to data marts, while recorded events are typically not subject to further changes, in some cases they can be modified to allocate errors or late notifications of up-to-date values for measures. Besides, the instances of dimensions and hierarchies are not entirely static.

- Handling schema changes in the data mart

  The data mart structure may change in response to the evolving business requirements. New levels and

measures may become necessary, while others may become obsolete. Even the set of dimensions characterizing a fact may be required to change.

- Querying temporal data (sixth section). Querying in presence of data and schema changes require specific attention, especially if the user is interested in formulating queries whose temporal range covers different versions of data and/or schema.

- Designing temporal data warehouses

The specific characteristics of temporal data warehouses may require ad-hoc approaches for their design.

II. TEMPORAL DATABASES

Databases where time is not represented are often called short-lived databases. Within an ephemeral database, only the current representation of real-world objects is stored and no track of changes is kept, so it is impossible to reconstruct how the object was in the past. Conversely, temporal databases focus on representing the inherent temporal nature of objects through the time-dependent recording of their structure and state. Two different time dimensions are normally considered in temporal databases, namely valid time and deal time. Temporal database systems are called valid-time databases, deal-time databases depending on their capacity to handle either or both of these two time dimensions. The main benefit of using a bi-temporal database is that not only the history of the changes an object is subject to is recorded, but it is also possible to obtain the same result from a query independently of the time when it is developed.

In the real world, objects change in both their state and their structure. This means that, within a database, both the values of data and their schema may change. Obviously, values of data are constantly modified by databases applications. On the other hand, altering the database schema is a less frequent, though still common, occurrence in database administration. With reference to changes in the database schema, the literature commonly distinguishes three possibilities:

- Schema alteration is supported when a database system allows changes to the schema definition of a populated database, which may lead to loss of data.

- Schema development is supported when a database system enables the alteration of the database schema without loss of existing data.

- Schema versioning is supported when a database system allows the accessing of all data, both retrospectively and prospectively, through user-definable version interfaces.

The significant difference between development and versioning is that the former does not require the maintenance of a schema history, while in the latter all past schema versions are confirmed.

The concepts introduced in this section were originally devised for operational data-bases, and in particular for relational databases. While in principle they can also be applied to data warehousing systems, that in ROLAP implementations that are based on relational databases, the peculiarities of the intricate model and the strong relevance of time in the OLAP world call for more specific approaches.

III. HANDLING CHANGES IN THE DATA WAREHOUSE

When considering temporal data, it is first of all necessary to empathize how time is reflected in the database, and how a new piece of information affects existing data. From this point of view, Devlin proposes the following classification:

- Ephemeral data: alterations and deletions of existing records physically destroy the previous data content.

- Periodic data: once a record is added to a database, it is never physically deleted, nor is its content ever modified.

- Semi-periodic data: in some situations, due to performance and/or storage constraints, only the more recent history of data changes is kept.
- Snapshot data: a data snapshot is a stable view of data as it exists at some point in time, not containing any record of the changes that determined it. A series of snapshots can provide an overall view of the history of an organization.

Each attribute, or each set of attributes having the same behaviour with reference is stored in a separate table so that a change occurred to one concept does not affect the other concepts. Obviously, such normalized and time-oriented structure is not suited for querying, that will take place on de-normalized data marts fed from the data warehouse.

Since the data warehouse can be thought of as a set of derived, materialized views defined over a set of source schemata, the problem of developing the content and the schema of derived views in connection to the source changes is highly relevant in the context of temporal data warehouses.

View maintenance consists in maintaining a materialized view in response to data modifications of the source relations.

View adjustment consists in re-calculating a materialized view in response to changes either in the schema of the source relations or in the definition of the view itself. Changes in the source schemata may be due to an development of the application domain they represent, or to a new physical location for them. Changes in the definition of the view may also be due to new requirements of the business users who query the data marts fed by the data warehouse. Performing a schema change leads to creating a new view, by means of an extended view definition language that incorporates two clauses: conceal which describes a set of attributes to be hidden, and add, that allows a view to own additional attributes that do not belong to source relations. In the EVE framework, in order to automate the redefinition of a view in response to schema changes in the data sources, the database administrator is allowed to embed her preferences about view development into the view definition itself. The preference-based view rewriting process, called view synchronism, identifies and extracts appropriate information from other data sources as replacements of the affected components of the original view definition, in order to produce an alternative view that somehow preserves the original one.

The key idea of adjustment techniques is to avoid recalculating the materialized view from scratch by relying on the previous materialization and on the source

relations. The adjustment of the data warehouse in response to schema changes arising on source relations located on multiple sites. To adapt the extent of the data warehouse in response to these changes, they adopts rewriting algorithms that make use of containment checking, so that only the part of the new view that is not contained in the old view will be recomputed. In the same context, a distinctive feature of the Auto Med system is the capability of handling not only schema developments in materialized data integration scenarios, but also changes in the data model in which the schema is showed.

With reference to the problem of keeping the data warehouse in sync with the sources, Bebel propose a model for handling changes in the operational data sources, which supports the automatic spotting of structural and content changes in the sources and their automatic propagation to the data warehouse.

IV. HANDLING DATA CHANGES IN THE DATA MART

Content changes result from user activities that perform their day-to-day work on data sources by means of different applications. These changes are reflected in the data warehouse and then in the data marts fed from it. The intricate model provides direct support for representing the sequence of events that constitute the history of a fact: by including a temporal dimension in the fact, each event is associated to its date. For instance, if we consider an ORDER fact representing the measures in the lines of orders received by a company selling PC consumables, the dimensions would probably be product, order Number, and order Date. Thus, each event would be associated to the ordered product, to the number of the order it belongs to, and to the order date.

On the other hand, the intricate model implicitly assumes that the attributes and the related levels are entirely static. This assumption is clearly unrealistic in most cases, for instance, considering again the order domain, a company may add new categories of products to its list while others can be dropped, or the category of a product may change in response to the marketing policy.

Another common assumption is that, once an event has been registered in a data mart, it is never modified so that the only possible writing operation consists in appending new events as they occur. While this is acceptable for a wide variety of domains, some applications call for a different behaviour , for example the measure of a product ordered in a given day could be wrongly registered or could be communicated after the ETL process has run.

These few examples emphasize the need for a correct handling of changes in the data mart content. Differently from the problem of handling schema changes, the issues related to data changes have been widely addressed by researchers and practitioners, even because in several cases they can be directly managed in commercial DBMSs. In the following subsections we separately discuss the issues related to changes in dimensional data and factual data, that events.

V. CHANGES IN DIMENSIONAL DATA

By this term we mean any content change that may occur within an instance of a placing, involving either the dimension itself, or a property. For instance, considering a product placing featuring levels type and category,
the name of a product may change, or a new category may be introduced so that

the existing types have to be reassigned to categories.

The study of changes in dimensional data has been pioneered by Kimball, who coined the term slowly-changing dimension to point out that, differently from data in fact tables, changes within the dimension tables occur less frequently. He proposed three basic modelling solutions for a ROLAP implementation of the intricate model, each inducing a different capability of tracking the history of data. Conversely, in the Type II solution, each change produces a new record in the dimension table: old events stay related to the old versions of hierarchies, while new events are related to the current version. In order to allow two or more tuples representing the same placing instance to be included in the dimension table, surrogate keys must necessarily be adopted. Finally, the Type III solution is based on augmenting the schema of the dimension table by representing both the current and the previous value for each level or attribute subject to change.

The first one proposes a temporal star schema that, differently from the traditional one, omits the time dimension table and time-packs each row in every table instead, treating the fact table and the dimension tables equally with respect to time. Similarly, the second one proposes to handle changes by adding time-packs to all the components of a intricate schema: the values of both dimensions and facts, the inter-level partial order that shapes placing instances and the fact-dimension relationships.

VI. CHANGES IN FACTUAL DATA

We start this section by preliminarily mentioning the two basic paradigms introduced by Kimball for representing inventory-like information in a data mart: the model, where each increase and decrease in the inventory level is recorded as an event, and the snapshot model, where the current inventory level is periodically recorded.

- Flow facts record a single deal or summarize a set of deals that occur during the same time they are monitored by collecting their occurrences during a time interval and are cumulatively measured at the end of that period. Examples of flow facts are orders and enrolments.
- Stock facts refer to an instant in time and are evaluated at that instant, they are monitored by periodically sampling and measuring their state. Examples are the price of a share and the level of a river.

By the term changes in factual data we mean any content change an event may be subject to, involving either the values of its measures or the dimensional elements it is connected to. Changes in factual data are a relevant issue in all those cases where the values measured for a given event may change over a period of time, to be consolidated only after the event has been for the first time registered in the data mart. The late measurements typically happens ,when the early measurements made for events are subject to errors when events inherently develop over time. This problem becomes even more evident as the timeliness requirement takes more importance .This is the case for zero-latency data ware-housing systems, whose goal is to allow organizations to deliver relevant information as fast as possible to knowledge workers or decision systems that need to react in near real-time to new information.

In these contexts, if the update state is to be made timely visible to the decision makers, past events must be continuously

updated to reflect the incoming late measurements. Unfortunately, if updates are carried out by physically overwriting past registrations of events, some problems may arise. In fact, accountability and traceability require the capability of preserving the exact information the analyst based her decision upon. If the old registration for an event is replaced by its latest version, past decisions can no longer be justified. Besides, in some applications, accessing only up-to-date versions of information is not sufficient to ensure the correctness of analysis.

Supporting accountability and traceability in presence of late measurements requires the adoption of a bi-temporal solution where both valid and deal time are represented by means of time packs. Only few approaches in the literature are specifically focused on studying this specific topic. Bruckner discuss the problem of temporal consistency in consequence of delayed discovery of real-world changes and propose a solution based on valid time, revelation time and loading time. Loading time is the point in time when a new piece of information is loaded in the data mart, while revelation time is the point in time when that piece of information was realized by at least one data source.

VII. HANDLING SCHEMA CHANGES IN THE DATA MART

According to schema, changes in the data mart may be caused by different factors:

- Subsequent design iterations in the context of an incremental approach to data mart design.

- Changes in the user requirements, produced for instance by the need for producing more sophisticated reports, or by new categories of users that subscribe to the data mart.

- Changes in the application domain, that means, arising from alterations in the business world, such as a change in the way a business is done, or a changing in the organizational structure of the company.

- New versions of software components being installed.
- System tuning activities.

As stated in the second section, depending on how previous schema versions are managed, two main classes of approaches may be distinguished: schema development, that allows alterations of the schema without loss of data but does not maintain the schema history, and schema versioning, where past schema descriptions are confirmed so that all data may be accessed through a version specified by the user. In the two following subsection these two classes of approaches will be separately surveyed.

VIII. DEVELOPMENT

In this context, FIESTA is a methodology where the development of intricate schema is supported on a conceptual level, thus for both ROLAP and MOLAP implementations. Core of the approach is a schema evolution algebra which includes a formal intricate data model together with a wide set of schema development operations, whose effects on both schema and instances are described. Essentially, the operations allow dimensions, placing levels, properties and measures to be added and deleted from the intricate schema. Since OLAP systems are often implemented on top of relational DBMSs, the approach also shows how a intricate schema can be mapped to a relational schema by means of a meta-schema that extends the list of the underlying DBMS. Each list of development operations is then transformed into a sequence of relational

development commands that adapt the relational database schema together with its instances, and update the contents of the meta-schema accordingly.

The development problems investigated with particular reference to its impact on the logical level for ROLAP implementations, namely, on star and snowflake schema. Eight basic development operators are defined. For each of them, the changes implied on star and snowflake schemata are described and their impact on existing SQL queries in reporting tools is discussed. Remarkably, an in-depth comparison reveals that the star schema is generally more potent than the snowflake schema against schema changes. A comprehensive approach to development is the one jointly devised at the Universities of Toronto and Buenos Aires. The fundamentals are laid, who propose a formal model for updating attributes at both the schema and instance level, based on a set of alteration operators.

IX. VERSIONING

One of the features of a data warehouse is its non-volatility, which means that data is integrated into the data warehousing system once and remains unchanged afterwards. Importantly, this feature implies that the re-execution of a single query will always produce the same result. While non-volatility in the presence of changes at the data level can be achieved by adopting one of the solutions discussed in the third section, non-volatility in the presence of changes at the schema level requires some versioning approach to be undertaken. In fact, it is easy to see that the ability to re-execute previous queries in the presence of schema changes requires access to past schema versions, which cannot be achieved with an development approach .The first work in this direction is COMET a model that supports schema and instance versioning. All classes in the model are time stamped with a validity interval, so multiple, subsequent versions of cubes can be stored and queried. Transformation of data from one version into the immediate one is supported, though the paper reports no details on how a new version can be obtained from the previous one, a comprehensive set of constraints that the versions have to complete in order to ensure the integrity of the temporal model is proposed.

Essentially, they propose two meta-models: one for managing a multi-version data mart and one for spotting changes in the operational sources. A multi-version data mart is a sequence of versions, each composed of a schema version and an instance version. Data migration from the old to the new version is semi-automated, that means, based on the differences between the two versions the system suggests a set of migration actions and gives support for their execution. The key idea of this approach is to support flexible cross-version querying by allowing the designer to enrich previous versions using the knowledge of current schema alterations. For this purpose, when creating a new schema version the designer may choose to create augmented schemata that extend previous schema versions to reflect the current schema extension, both at the schema and the instance level. In a nutshell, the augmented schema associated with a version is the most general schema describing the data that are actually recorded for that version and thus are available for querying purposes. Like for migration, a set of possible augmentation actions is proposed to the designer.

To the best of our knowledge, only two approaches use both valid and deal time in the context of versioning. Each

version has a temporal pertinence composed by a valid time and a deal time, thus enabling the existence of two or more versions with the same valid time, but different deal times. Associated to this model, there are 16 operators for schema changing and a SQL-like language to create and alter versions.

X. QUERYING TEMPORAL DATA

The development of a model for temporal data warehousing is of little use without an appropriate query language capable of effectively handling time. In principle, a temporal query could be directly developed on a relational schema using standard SQL, but this would be exceedingly long and complex even for a skilled user.

- Up-to-date queries, that require the most recent measurement for each event,
- Rollback queries, that require a past version measurement for each event,
- Historical queries, that require multiple measurements for events, that means, are aimed at reconstructing the history of event changes.

All three querying scenario are supported. Also meta-queries, e.g. concerning the instant changes to data took place, can be showed.

In the context of querying, a number of works are related to the so-called temporal aggregation problem, studied mainly in the context of MOLAP systems and consists in efficiently calculating and maintaining temporal aggregates. In fact, time dimensions typically lead to a high degree of sparseness in traditional array-based MOLAP cubes because of their large cardinality, and to significant overhead to answer time-parameterized range queries. Specifically, for count queries, its goal is to provide answers guaranteed to deviate from the exact ones within a given threshold. Their framework allows large amounts of new data to be integrated into the warehouse and historical summaries to be efficiently generated, independently of the extent of the data set in the time dimension. They proposed a general approach to improve the efficiency of range aggregate queries on MOLAP data cubes in a temporal data warehouse by separately handling time-related dimensions to take advantage of their monotonic trend over time. Finally, it introduce a new index structure called the SB-tree, which supports fast lookup of aggregate results based on time, and can be maintained efficiently when the data changes along the time line.

XI. DESIGNING TEMPORAL DATA WAREHOUSES

It is widely recognized that designing a data warehousing system requires techniques that are radically different from those normally adopted for designing operational databases. On the other hand, though the literature reports several attempts to devise design methodologies for data ware-houses, very attention has been posed on the specific design issues related to time.

Pedersen and Jensen, recognize that properly handling time and changes is a must-have for intricate models.

Considering the leading role played by temporal hierarchies within data marts and OLAP queries, it is worth adopting ad hoc approaches for their modelling not only from the logical, but also from the conceptual point of view. While all conceptual models for data marts allow for temporal hierarchies to be represented like any other hierarchies, to the best of our

knowledge the only approach that provides ad-hoc concepts.

XII. CONCLUSIONS

In this survey we discussed the issues related to temporal data warehousing. An in-depth analysis of the literature revealed that the research community not always denoted a comprehensive attention to all these aspects described above. As a matter of fact, a wide agreement on the possible design solutions has been reached only with reference to changes in dimensional data. As to changes in factual data and changes in schema, though some interesting solutions have been proposed, no broad and shared framework has been devised yet. Already in year 2000, systems such as Business Warehouse by SAP (2000) were allowed to track changes in data and effectively query cubes based on different temporal scenarios by choosing users to choose which version of the hierarchies to adopt for querying. On the other hand, today there still is very marginal support to changes in schema by commercial tools. Also, the Oracle Change Management Pack is aimed to report and track the evolving state of metadata, thus allowed to compare database schemata, and to generate and execute scripts to carry out the changes. In both cases, formulating a single query spanning multiple databases with different schemata is not possible. We believe that, considering the maturity of the field and the wide diffusion of data warehousing systems, in the near future decision makers will be more and more demanding for advanced temporal support. Thus, it is essential that both vendors and researchers be ready to deliver effective solutions. In this direction we envision two main open issues. For instance, support for cross-version queries is not satisfactory yet, and its impact on performance has not been completely investigated, similarly, the effectiveness of view adjustment approaches is still limited. On the other hand, in order to encourage vendors to add full temporal support to commercial platforms, the solutions proposed in the literature should be better harmonized to converge into a complete, flexible approach that could be effortlessly accepted by the market

REFERENCES

[1] Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J., Ramakrishna, R., and Sarawagi. S. On the computation of multidimensional aggregates. *Proc. VLDB*, 1996.

[2] Choi, W., Kwon, D., and Lee, S. Spatio-temporal data warehouses using an adaptive cell-based approach. *DKE*, 59, 1 (Oct. 2006), 189-207.

[3] eCourier.co.uk dataset, http://api.ecourier.co.uk/. (URL valid on June 20, 2009).

[4] Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. Trajectory pattern mining. *Proc. KDD*, 2007.

[5] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. Data cube: A relational aggregation operator generalizing groub-by, cross-tab and sub-totals. *DMKD*, 1, 1 (Mar. 1997), 29-53.

[6] Han, J., Stefanovic, N., and Koperski, K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. *Proc. PAKDD*, 1998.

[7] Jensen, C.S., Kligys, A., Pedersen, T.B., Dyreson, C.E., and Timko, I. Multidimensional data modeling for location-based services, *VLDBJ*, 13 (Jan. 2004), 1–21.

[8] Kalnis, P., Mamoulis, N., and Bakiras, S. On discovering moving clusters in spatio-temporal data. *Proc. SSTD*, 2005.

[9] Lee, J., Han, J., and Whang, K. Trajectory Clustering: A Partition-and-Group Framework. *Proc. SIGMOD*, 2007.

[10] Li, X., Han, J., Lee, J.-G., and Gonzalez, H. Traffic density-based discovery of hot routes in road networks. *Proc. SSTD*, 2007.

[11] Liu, Y., Choudhary, A.N., Zhou, J., and Khokhar, A.A. A scalable distributed stream mining system for highway traffic data. *Proc. PKDD*, 2006.

[12] Marketos, G., Frentzos, E., Ntoutsi, I., Pelekis, N., Raffaeta, A., and Theodoridis, Y. Building Real World Trajectory Warehouses. *Proc. MobiDE*, 2008.

[13] Ntoutsi, I., Mitsou, N., and Marketos, G. Traffic mining in a road-network: How does the traffic flow? *IJBIDM*, 3, 1, (Apr. 2008), 82-98.

[14] Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., and Silvestri, C. Trajectory Data Warehouses: Design and Implementation Issues. *JCSE*, 1, 2 (Dec. 2007), 211-232.