Version Manager: A step towards Synthetic-Warehouse-Builder extension

M. Khurram Shahzad
khur-sha@dsv.su.se
Department of Computer Science,
M.A Jinnah Campus, Defense Road, off Raiwand Road,
COMSATS Institute of Information Technology, Lahore, Pakistan.

Abstract--To achieve vital advantages of simulating business scenarios, extensions are made to versioning concept by dividing alternative versions into two types, Materialized Alternative Versions and Virtual Alternative Versions. This paper presents querying method from multi-version data warehouse in the presence of real versions, materialized and virtual alternative versions. This is done by making extensions to our Synthetic Warehouse Builder with new component called Version Manager.

Version manger with its confinement process divide query into smaller parts called mono-version queries; execute it on relevant and heterogeneous version/s and merging the result to present it for analysis. It is expected that, Version Manager is very useful for short term data requirements as well as rapidly changing requirements of simulating business alternatives.

Keywords-- Data Warehouse, evolution, versioning, querying DW, architecture, data retrieval.

I. INTRODUCTION

Data Warehouse (DW) provides integrated access to transactional data source/s. But dynamics of this source/s has resulted in derivation of inconsistent analytical results from DW [1]. These dynamics can be of two types: a) Content changes, due to execution of data manipulation language b) Schema changes, due to execution of data definition language [2].

Various naive approaches have been proposed to handle business dynamics, these approaches can be categorized, schema evolution [3], Versioning approach [4]. With evolution approach, schema is upgraded, but has high maintenance cost [5].

For better what-if-analysis and to improve the quality of decision-making the concept of simulating business scenarios has been proposed [1,6], but conventional DW does not support simulating business scenarios. Hence, two types of versions are proposed: a) real version b) alternative versions. The DW which maintains these versions is called multi-version data warehouse (MV-DW). Retrieval from MV-DW has been made possible by multi-version query language [7].

For solving querying problems of we have developed Synthetic Warehouse Builder (SWB) [8]. SWB provides transparent access to users for querying multiple versions of DW without caring about implementation details of versions. Recent study shows that vital advantages of simulating business scenarios can only be achieved by using three types of versions a) Real versions b) Materialized Alternative Versions c) Virtual Alternative Version [9].

Observations-- a) SWB do not provide flexibility of simulating business scenarios in MV-DW b) Vital advantages of, simulating business scenarios, cannot be achieved by exploiting alternative versions c) The conception of handling simulating business scenarios by alternative versions is required to be re-worked d) To extend the functionality of SWB, Version-Manager is required.

Contribution-- The objective of this paper is 1) to present the architecture of Version-Manager (an added component of SWB) to provide flexible environment for what-if-analysis and simulating business scenarios in order to improve the quality of decision-making. 2) Querying method for three types of versions, without letting the users to know about location of data in any version.

Rest of paper is organized as follows; section no 2 gives the motivation for extending the SWB functionality, section 3 overviews existing approaches to address evolutionary problems of DW and simulating business alternatives. Introduction to SWB, its properties along with functionalities are given in section 4 while section 5 presents the proposed architecture, and query processing using Version Manager is given in section 6 of the paper. Finally, paper is concluded in section 7 with small discussion.

II. MOTIVATION

In today's knowledge oriented society, success of organization depends upon quality of decision-making [5]. Simulating business scenarios for better what-if analysis has recently been proposed [6, 1] by maintaining real and alternative versions, but various businesses are found to have vibrant simulating scenarios i.e. they can be changed at runtime. Since alternative versions have instances attached with them so changes are

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 289–293.

absolutely not possible to be made, hence restricts the scope of the user's what-if-analysis.

For achieving vital advantages of alternative business scenarios and providing users with a flexible simulating environment, alternative versions are divided into two types [9].

a) Materialized Alternative Versions (MAV), have instances attached with them and can also share data with its child versions. MAV's are used for simulations, whose structure are static and are used most of the time. These are also called static simulations.
b) Virtual Alternative Versions (VAV), have no instance attached with them instead they are stored like virtual tables in multi-version catalog and gets data on-the-fly from mapped set of real and materialized alternative versions. These types of alternative versions are used for vibrant (dynamic) simulations.

Storing and mapping of versions is not the real task instead it is required to retrieve data from various versions and make it useful for malicious decision-making, which has not been done.

SWB provides transparent access to real and alternative versions, but does not provide flexible simulating environment, hence raises the need for addition of component called Version-Manager (VM). VM is expected to not only provided interface for simulating business scenarios and run-time addition of virtual dimension and fact tables but also provide facility for querying a) Real Versions b) Materialized Alternative Versions c) Virtual Alternative Versions. The only extra effort required is one-time mapping of VAV's to respective MAV/s and RV/s.

III. Related Works

Transactional systems act as a source of data for DW. It is found that changes to these sources may result in derivation of inconsistent results [1, 5]. Semi-Star [5] can handle these changes but are suitable only for small and medium sized dynamic organization. Other methods proposed to handle these changes are: a) Schema and data evolution method [3, 4], according to this approach changes to schema are applied to new schema and data is transported to new schema. But this approach gives unsatisfactory response time in the presence of increased number of changes [8]. b) Temporal versioning method [10, 11], in which changes new DW versions are created with a time stamp but its disadvantages has already been discussed in [12,13] and quoted in [1].

For achieving the objective of better what-if-analysis, concept of simulating business alternatives has been reported [1], which can be handled along with source dynamics by using two types of versions i.e. real and alternative versions. For querying purpose multi-version query language has been proposed in [7] by extensions to standard query language. We have already removed shortcomings by building SWB for transparent querying method [8].

But, a) Vital advantages of simulating alternative business scenarios (SBS) has not been achieved b) It is required to provide users with a flexible environment for using simulations to improve quality of decision-making c) SWB has restricted querying scope i.e. only the set of three querying possibilities exists, that are: i) Querying only current DW version ii) Query the set of real DW versions iii) Or querying set of selected alternative DW versions. These issues can only be addressed by adding Version Manager to SWB.

IV. Synthetic Warehouse Builder [8]

SWB provides transparent access to set of real and alternative versions. While using SWB users are not required to keep track of set of versions, their derivation hierarchy as well as instances attached with the versions, instead only the knowledge of logically integrated schema will work.

Transparency in SWB has been defined at three levels [8] a) Source transparency, data stored in different versions are made transparent to users. i.e. user will retrieve data from multiple versions independent of the existence of source table/s in version/s, but it should be present in at least one of the versions.

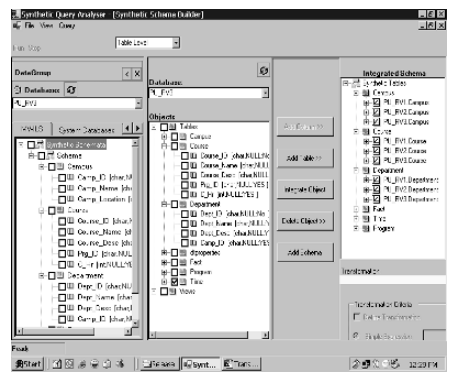

Fig.1. MV-ILS Builder [8]

b) Projection transparency, user should write the query independent of existence of attribute in one or more tables of different versions.
c) Analysis transparency, in the presence of this type of transparency, user will write query independent of predicate-attribute present in one or more versions. But it should be available in at least one version. The advantage of fully transparent access to DW is high level support, which it provides for development of complex relational multi-version DW.