Hindawi
Scientific Programming
Volume 2017, Article ID 7392349, 18 pages
https://doi.org/10.1155/2017/7392349

Review Article

Temporal and Evolving Data Warehouse Design

Sidra Faisal, Mansoor Sarwar, Khurram Shahzad, Shahzad Sarwar, Waqar Jaffry, and Muhammad Murtaza Yousaf

Punjab University College of Information Technology, Lahore, Pakistan

Correspondence should be addressed to Khurram Shahzad; khurram@pucit.edu.pk

Received 30 May 2017; Accepted 16 October 2017; Published 26 November 2017

Academic Editor: Mario Alviano

The data model of the classical data warehouse (formally, dimensional model) does not offer comprehensive support for temporal data management. The underlying reason is that it requires consideration of several temporal aspects, which involve various time stamps. Also, transactional systems, which serves as a data source for data warehouse, have the tendency to change themselves due to changing business requirements. The classical dimensional model is deficient in handling changes to transaction sources. This has led to the development of various schemes, including evolution of data and evolution of data model and versioning of dimensional model. These models have their own strengths and limitations, but none fully satisfies the above-stated broad range of aspects, making it difficult to compare the proposed schemes with one another. This paper analyses the schemes that satisfy such challenging aspects faced by a data warehouse and proposes taxonomy for characterizing the existing models to temporal data management in data warehouse. The paper also discusses some open challenges.

1. Introduction

Today, most applications related to finance, record keeping, scheduling, and weather forecasting demand time-varying nature of data in order to identify the trends by comparing the current state of data with its previous states. The analyses of these trends may lead to identifying the underlying patterns in data, predicting the future and making informed decisions [1]. In the absence of the time-varying nature of data in transaction systems, users may miss important trends in data or may infer the trends incorrectly [2–4]. However, temporal data management for these applications is a challenging task. It is because, besides temporal data management, it is likely that an organization changes itself and to accommodate the organizational changes the corresponding information system is also changed [5]. The reasons for these changes could be implementation of new ideas, compliance with the changes forced by the market and/or relevant standards bodies, or changes in government policies. Here, it is important to note that the nature of many changes cannot be foreseen because there are a number of factors that are associated with it, including dynamic nature of the market, changing weather, new standards, and unforeseen policies of government.

Various temporal data models such as [6, 7] have been proposed for temporal data management; specifically, the database that can store and manage time-varying data by using temporality types, called Temporal Database (TDB). Although TDBs provide management of temporal data, they cannot support analysis on the basis of historical data aggregation. However, these databases can serve as a useful source for supporting analysis and thereby decision support. Such a database is called a data warehouse (DW). A DW is a specialized storage area that captures processed and integrated data from several heterogeneous data sources, which may include TDBs, for analytical purposes [8]. Golfarelli and Rizzi [9] argue that a DW has "rapidly spread in the industrial world due to undeniable contribution to increasing the effectiveness and efficiency of the decisions." Since data in a DW is used for decision support, therefore the data model for a DW should be designed in a way that it is optimized for this purpose [10]. A data model of DW (also known as Star or multimultidimensional model) consists of a central *fact table* and many *dimension tables* around it. The values in a dimension table are usually textual, relatively stable, discrete, and used as input conditions for analysing measures. To support various levels of analysis, the dimension tables

may have hierarchies in it. A fact table contains foreign keys for all other dimension tables and one or more measures that describe the critical values for the subject of interest. Measures are the nonkey and numeric values in the fact table.

A DW is dependent on its data sources for data population, which is done through a process called Extraction, Transformation, and Loading (ETL). However, the challenging task is the presence of TDBs as data sources. Recall from a preceding paragraph that an organization changes itself and to accommodate the organizational changes the corresponding database system is also changed. This requires adjustment to the multidimensional model [10], that is, to accommodate the changes introduced by TDB [11].

The adjustments become further challenging if the data sources are heterogeneous and they change with time, independently of each other. This may lead to two types of changes to a DW, *content changes* and *schema changes* [4, 10]. The *content changes* include insert, update, and delete operations on records while *schema changes* include adding, modifying, and dropping an attribute or a table in the DW data model. To summarize, the existence of TDBs and changes in them lead to the development of a Temporal Data Warehouse (TDW). A TDW requires consideration of several aspects in its DW, including temporal support in its data models [12–16], changes to its schema, and changes to its data [17–21].

Our work analyses the challenging aspects faced by TDWs and the solutions that have been proposed to address these challenges. Currently, the studies to analyse the capabilities of TDW are scarce [9, 22] and these studies have at least one of the following limitations: (a) they focus on explaining the concept of Temporal Databases and the ways of handling changes in the DW but do not aim to offer a comparative analysis of the existing approaches, (b) they are old and, thus, do not include the latest development in the area, and (c) they do not provide a comparison of temporal support in the existing schemes based on various time stamps. For instance, Golfarelli and Rizzi [9] discuss the handling of schema and data level changes in data warehouses and data marts, design of TDWs, and querying temporal data. However, they do not include the recent efforts such as [23, 24] as well as a comparative analysis of these approaches. Wrembel [22] also focuses on a subset of approaches, called Multiversioned Data Warehouses (MVDWs). Additionally, it includes a language for querying a MVDW, as well as the detailed structure for sharing data and indexing data in a MVDW. Wrembel [23] only discusses the challenges that arise in the design, construction, and management of evolving external data sources as well as evolution of an ETL layer. All these approaches do not provide a comparison of the temporal support provided by the existing schemes.

In the remaining part of this section, we present a working example to illustrate the concept of a temporal and evolving DW as well as better understand the issues involved with it. This is followed by a taxonomy for temporal and evolving DW.

1.1. Working Example. Today, most applications related to finance, record keeping, scheduling, weather forecasting, and so forth demand time-varying nature of data in order to identify the trends by comparing the current state of data with its previous states. The analyses of these trends may lead to identifying the underlying patterns in data, predicting the future and making informed decisions [1]. In the absence of the time-varying nature of data in transaction systems, users may miss important trends in data or may infer the trends incorrectly [2–4]. However, temporal data management for these applications is a challenging task. It is because, besides temporal data management, it is likely that an organization changes itself and to accommodate the organizational changes the corresponding information system is also changed [4]. The reasons for these changes could be implementation of new ideas, compliance with the changes forced by the market and/or relevant standards bodies, or changes in government policies. Here, it is important to note that the nature of many changes cannot be foreseen because there are a number of factors that are associated with it, including dynamic nature of the market, changing weather, new standards, and unforeseen policies of government.

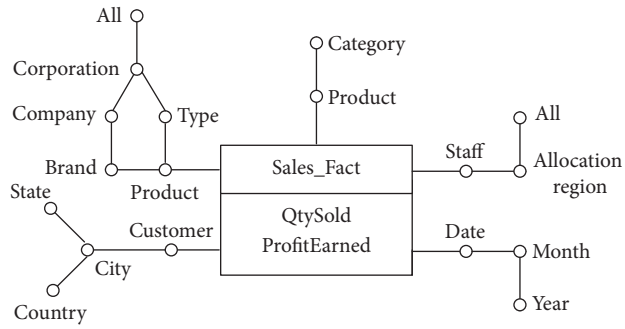


Figure 1: S_0: example of dimensional model for the sales company.

Existing studies, such as [25], suggest the use of a working example for better understanding of the changes in a DW. Thus, to further understand some of changes that can happen in a DW as a result of the changes in its data sources, consider a working example of a sales company that operates in several states. Each state is divided into a number of regions. The company has a number of staff members who operate in those regions. The staff members are responsible for selling a number of products to its customers dispersed in those regions. The sales company maintains data about customers and their city and state for analysis. The products sold to customers are of different brands and are classified into different categorizes based on their features. Consider S_0 as an excerpt conceptual data model of a DW depicted at 1/1/2012 (say t_0) in Figure 1 using the Dimension Fact Model (DFM) formalism, proposed in [26].

The central element Sales_Fact shown in Figure 1 is a fact table having two measures, namely, Sales_Quantity and ProfitEarned. The five elements surrounding the central element are dimensions, namely, product, staff, time, customer, and Product_Category. Each dimension has a hierarchy in which an arc shows many-to-one association, that is, many-to-one associations in the product dimension are between product and brand and company and brand. This means that a brand (say b_1) can have many products say $(a_1, a_2, \ldots, a_n)$. Similarly,

a company (say c_1) can have many brands (say $b_1, b_2, \ldots, b_n$).

The adjustment refers to two types of changes that are possible in the DW data model, *schema changes* and *content changes* [10].

(i) *Schema changes* include adding, modifying, and dropping an attribute or a table in the DW data model [17].

(ii) *Content changes* include operations such as insert, update, and delete on the records of DW.

Thereafter, we use S_0 shown in Figure 1 to illustrate the two types of changes. The motivation for choosing the example changes is that the working example and corresponding changes are subsequently used to elaborate the temporal and evolving DW in the rest of the paper.

Schema Changes. They refer to the changes in the data model of the DW. The possible set of schema change operations defined in [17] are insert level, delete level, insert attribute, delete attribute, connect attribute to a dimension level, disconnect attribute from a dimension level, connect attribute to a fact, disconnect attribute from a fact, insert classification relationship, delete classification relationship, insert fact, delete fact, and delete dimension.

For the sales company example, consider that by 1/1/2013 (say t_1) the analyses requirements lead to some adjustments in the DW data model. The adjustments and their corresponding schema change operations are as follows: (a) instead of analysing the sales performance on monthly basis, the company is interested in analysing the sales performance on weekly basis. Here, the change operations are delete level and add level; (b) instead of analysing the sales performance by Product_Type, the company is interested in analysing product by brands and their types. The change operations therefore become connect attribute to dimension level and disconnect attribute from dimension level; (c) subcategories are defined in each product category. The corresponding operations are insert level and connect attribute to dimension; (d) the company is not interested in analysing QtySold by states. Therefore, the corresponding change operation for this case is delete level. The DW data model generated as a result of these changes is shown in Figure 2(a).

By 1/1/2014 (say t_2) the analyses requirements further lead to some adjustment in the DW data model. The adjustments and their corresponding schema change operations are as follows: (a) instead of analysing the sales performance on weekly basis, the company is again interested in analysing the sales performance on monthly basis. The corresponding change operations are insert attribute, insert level, and delete attribute; (b) instead of analysing products by brands (as done in S1) the company is interested in simply analysing products by Product_Types, that is, without any association with company but with corporation. Similar to the preceding changes, these changes can be mapped to *schema change operations*. The data model generated as a result of these changes is shown in Figure 2(b).

Further, assume by 1/1/2015 (say t_3) the analyses requirements lead to some adjustment in the DW data model. The adjustments are that the company is not interested in the brands of any product. The DW data model generated as a result of these changes is shown in Figure 2(c).

Content Changes. They refer to the changes in the content of the DW. The possible set of content change operations is as follows: insertion of a record, deletion of a record, and update of a record in dimension tables or fact tables. For the sales company example, consider the staff dimension table and the Sales_Fact fact table in Figure 1. The staff dimension has a hierarchical structure in which staff members are assigned to regions. Also, consider a set of records at time t_0 (i.e., 1/1/2012) for S0 of the dimension and fact table as shown in Table 1. From the content it can be seen there are two allocation regions, AR1 and AR2. Two staff members S1 and S2 belong to AR1, whereas one staff S3 belongs to AR2. The QtySold in region AR1 is 150 whereas that of AR2 is 100.

Consider at time t_1 (i.e., 1/1/2013) that a contention change occurs to the DW data model S0. According to the change, the staff member S2 that belonged to AR1 is moved to AR2, as shown below in Table 2. As a result, the QtySold in region AR1 becomes 100 and AR2 becomes 150. This is an example of content changes in the DW that may require adjustment in it. The two types of changes demonstrated above are used in the rest of the paper to exemplify the existing approaches.

1.2. Taxonomy of Temporal and Evolving DW. Recall that a DW is dependent on its data sources for data population, called Online Transaction Processing, or OLTP. For temporal data management in OLTP, various temporal data models have been proposed. Specifically, an OLTP that can store and manage time-varying data is called a Temporal Database (TDB). A key feature of TDBs is that the entities and their relationships are captured and maintained along with their time stamps. One such time stamp is called *valid time*. *Valid time* represents the time during which the data values are valid. For the sales company example, consider a staff member S_1 having monthly salary USD 2000 from 1/1/2014 to 31/12/2014. Also, consider that her salary gets a 10% increase from 1/1/2015 making her salary USD 2200. In this example, the *valid time* of salary amount USD 2000 is from 1/1/2014 to 31/12/2014, and the *valid time* of salary USD 2200 is from 1/1/2015 until the new value is added. Other than *valid time*, *transaction time* is another type of time stamp in a TDB. Since, a DW is dependent on its data sources, which could be a TDB; therefore the nature of temporal support in the DW is dependent on the time stamps captured by the TDB. This means that if the *valid time* of a data item is not stored in the TDB, it may not be possible to have *valid time* for that data item in the DW.

Since the aim of this study is to analyse temporal data management as well as handling changes in a DW, so we build a taxonomy to group the existing approaches based on their features. This paper follows the taxonomy given in Figure 3 to list the features. According to the taxonomy, for clearly distinguishing the features of existing approaches, *temporal support* in a *DW* is separated from *handling the changes* as well as from their *support in commercial tools*. This is shown in the first level of the taxonomy.

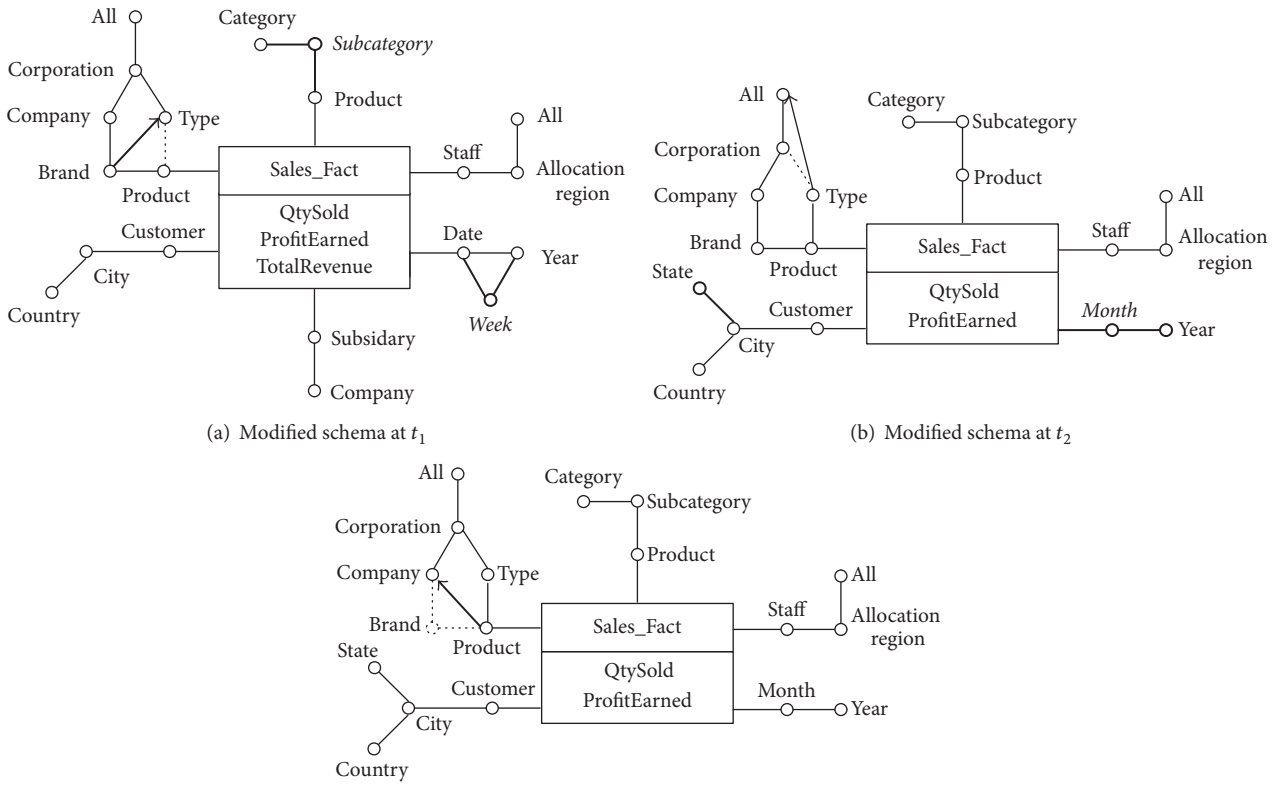


Figure 2: Schema modifications.

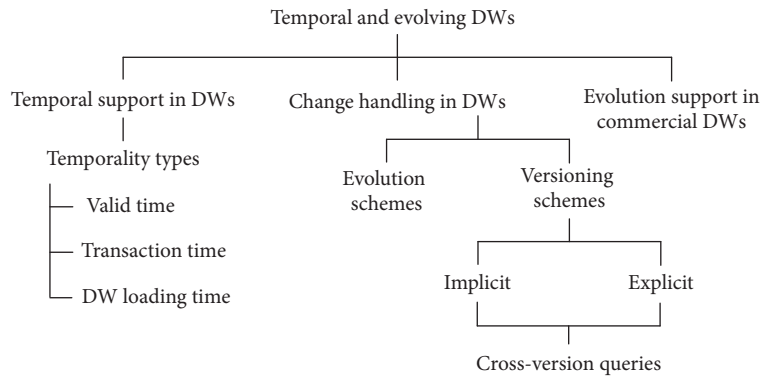


Figure 3: A taxonomy of temporal and evolving DWs.

The taxonomy is summarized below.

(i) The *temporal support* in a DW is required, particularly, when its source is a TDB [12–15]. Therefore, we dedicate Section 2 to discussing the insight of the various time stamps for temporal data management. The types of time stamps are labelled as temporality types in the figure.

(ii) For *handling changes* in a DW (recall *schema* and *data changes*), a number of efforts [17–21] have been made. According to Eder and Wiggisser [27] and Eder and Wiggisser [28], there are two ways to handle these changes: (a) updating schema, transforming existing data into the new schema, discarding the schema, and using the updated schema for future data population: this is called *evolution* scheme [2]; (b) creating a new schema, maintaining both schemas, and using only the new schema for future data population [24]: however, both schemas can be used for retrieval from the DW. The is called *versioning* scheme. Our classification of change handling is aligned with the discussion as shown in Figure 3. The details of the classification schemes as well as the types of changes are discussed in Section 3.

(iii) Furthermore, the last segment of the taxonomy is regarding assessment of the adaptation of research in *commercial tools* for DWs. We provide an overview of the tool support for temporal data management and handling changes to DWs by the leading vendors in Section 4.

Section 5 concludes this paper and gives some future research directions.

2. Temporal Data Warehouse

DW is dependent on its sources for data population; therefore, if the data about some business object is not captured in its source, it is likely that the business object cannot be made part of the DW design. The exception is to involve an external data source to extract the missing information. For the sales company example, if the data about Customer_Gender is not maintained in the source OLTP, it cannot be designed as part of the DW data model. However, it is possible that, by involving some external data sources, such as social network identity or customer name, we may derive/extract Customer_Gender. In line with that, if temporal data management is not part of data sources, it is likely that the temporal data management cannot be made part of the DW. Due to this dependency of a DW on its data sources, we discuss the temporal data management in its sources as well as in the DW. We also discuss how the meaning of a time stamp in a TDB, say *valid time*, is different from that of a DW. Thereafter, we discuss the special case where a time stamp can be made part of a DW, even if it is not part of a source OLTP. Below, we explain the time stamps and their meanings in the context of source OLTP and DW.

Time Stamps for Source OLTP. The data model that captures time-varying information is called temporal data model or temporal model. In temporal models, objects and their relationships are provided with time stamps. For source OLTPs, the two key time stamps are *valid time* (VT) and *transaction time* (TT) [15, 29, 30].

If the source OLTP is a snapshot system, it supports VT to uncover changes and compare data with the previous copies of data. VT represents the time during which the data remains constant and true in the real world [31]. The user specifies the value of a VT. For the sales company example, consider a staff member S_1 having monthly salary USD 2000 from 1/1/2014 to 31/12/2014. Also, consider that her salary gets a 10% increase from 1/1/2015 making her salary USD 2200. In this example, the VT of salary's amount USD 2000 is from 1/1/2014 to 31/12/2014, and the VT of salary USD 2200 is from 1/1/2015 until the new value is added.

If the data source is a logging system it supports TT to enlist all activities in log files. TT in TDBs defines the time instant when the system performs a transaction associated with an activity in real life. Specifically, it represents the time stamp at which data was made current and retrievable and remains logically present in the database [31]. When data in a database gets modified, its corresponding TT is also changed accordingly. In fact, TT is entirely maintained by the system and users are not allowed to modify it [3]. For the salary example given in the preceding paragraph, the TT of a staff member S_1, having monthly salary USD 2000 with valid time from 1/1/2014 to 31/12/2014, is the time when the salary USD 2000 was entered into the system. Here it is important to note that it is not necessary that the salary amount USD 2000 was entered on 1/1/2014, meaning that the actual salary amount can be entered in the system before or after that time, say TT 1/6/2014, and it can be made applicable from a previous date, say 1/1/2014. Similarly, the salary amount can be entered before the applicable time.

Time Stamps for DW. In the data model of a DW, dimensions, facts, and/or their relationships can be provided with time stamps. For a DW, the key time stamps are VT, TT, and DW load time (DWLT) [15, 32]. A brief description of the three time stamps is as follows.

In a DW, VT represents the time during which dimensions and aggregate data remain true in the real world. VT is important in a DW since it permits correct aggregation of measures [33]; that is, in the absence of VT misleading aggregation of measures can be generated. For the sales company example, the total amount paid to staff S_1 since 1/1/2014 can be calculated by multiplying the salary of staff with the number of months, say 18 months. Since the current salary is USD 2200, the total amount paid to S_1 becomes USD 39600. However, given that the VT of salary USD 2000 is from 1/1/2014 to 31/12/2014 and the VT of salary USD 2200 is from 1/1/2015 to date, the given aggregation of measures is misleading. The correct amount paid becomes 37200 ($2000 * 12 + 2200 * 6$). This difference in the two aggregates becomes increasingly misleading with the passage of time.

Recall that TT used in a source OLTP represents the time when the data is "current"; that is, TT is changed when the data in source gets modified. However, data in a DW is neither modified nor deleted for each single data change. Thus, the TT generated in a DW has different meaning from TT in a source OLTP [32]. There are two possibilities for that: (a) to carry the source TT to the DW and (b) to generate a new time stamp, called DW load time (DWLT). In the *former case*, TT defines the time instance when the data was recorded in the source OLTP; that is, the value of TT in the DW is the same as the value in source OLTP. In the latter case, DWLT is generated in the same way as TT was generated in the source OLTP. Here, it defines the instant when the records or modifications are loaded in the DW. For the salary example, in the source OLTP, the TT of staff S_1 salary is 1/6/2014 and TT of staff S_2 salary is 2/6/2014. For this data modification, the DWLT is always greater than TT of the source OLTP; that is, for S_1, DWLT can be any time after midnight 1/6/2014 (i.e., 00:00:00) and for S_2 DWLT can be any time after midnight 2/6/2014. It is because data cannot be extracted from an OLTP until it is inserted into the OLTP.

2.1. Temporal Support in Data Warehouse. The temporal support provided in a DW mainly depends on the temporal support provided by the source OLTPs as well as the needs of analysis. Here, we describe the various possibilities regarding the availability of time stamps in an OLTP and the support

Table 1: Sample data for staff dimension at t_0.

t_0 = 1/1/2012		
Allocation region	Staff	Sales
AR1	S1	100
	S2	50
AR2	S3	100

Table 2: Sample data for staff dimension at t_1.

t_1 = 1/1/2013		
Allocation region	Staff	Sales
AR1	S1	100
AR2	S2	100
	S3	50

Table 3: Temporal support in source OLTP and DW.

Cases	Source OLTP		DW		
	VT	TT	VT	TT	DWLT
1	No	No	No	No	Yes$^{+/-}$
2	Yes	No	Yes	No	Yes$^{+/-}$
3				*Yes**	*Yes*$^{+/-*}$
4	No	Yes	No	Yes	Yes$^{+/-}$
5			*Yes*		Yes$^{+/-}$
6	Yes	Yes	Yes	Yes	Yes$^{+/-}$

$*$ and +/− have a special interpretation (see description).

for specific time stamps in the DW. There are six possibilities that are shown in Table 3 and discussed below.

Case 1 describes the scenario when no time stamps (neither VT nor TT) are provided in the source OLTP and the decision-making users want to know the history of source data. In this case, DW cannot support VT or TT of source. However, the aggregate measures can be time-stamped with DWLT to represent the time when the aggregates are loaded into the DW. The special sign (+/−) with DWLT represents that the value may or may not be "Yes," depending upon the user. The special sign is used to indicate that this time stamp is independent of sources and can be used independently of the existence of any time stamp in the source OLTP. For the salary increase example, if the VT and TT of total amount paid to staff S_1 are not known it is not possible to support VT or TT in the DW. However, it can be time-stamped to mark that the value was loaded at the end of the second quarter.

Case 2 describes the scenario when VT is provided in source OLTP and TT is not provided. In this case, the DW can support VT but cannot support TT of source. For the salary increase example, if VT of the total amount paid to staff S_1 is given in source and TT is not given, it is possible to compute the true value of the amount paid in the DW. However, since it is not known when the salary amount was entered into source OLTP during the previous years, the time stamp of update cannot be known. This means that true aggregates can be computed and aggregate measures may be time-stamped with DWLT. However, the changes cannot be retraced.

Case 3 describes the scenario when VT is provided in the source OLTP and TT is not provided. In this case, the DW can support VT but cannot support TT of source. However, a new TT can be generated in the DW in the same way as TT was generated in the source OLTP. The new time stamp (formally, DWLT) allows knowing when data was inserted, deleted, or modified in the DW. Nevertheless, the DW data is neither deleted nor modified for each instance change; thus this new time stamp indeed represents the time when it was loaded into the DW [7, 32]. So, in a way TT as well as DWLT can be known and their value is the same. This is a special case with a specific interpretation of TT so it is marked with an asterisk ($*$) in Table 3. The value of DWLT is marked with asterisk to represent that the value of its each instance will be the same as TT. Another distinction of this case is a time stamp (TT in this case) that can be provided to DW, even if it (TT) is not available in the source OLTP. For the salary example, assume that, in the source OLTP, TT of staff S_1 salary is not known, but VT is known (say, 1/1/2014 to 31/12/2014). From this data, it is not possible to know when the amount was entered in the source OLTP but it can be known when it was inserted in the DW because of DWLT, say 30/06/2014. Since this is an insertion to the DW which is the same in meaning as TT, therefore, it can be said that the value of TT is 30/06/2014 00:00:00.

Case 4 describes the scenario when TT support is provided in the source OLTP and VT is not provided. In this case, the DW can support TT but cannot support VT. It is the simplest case and does not require any illustration.

Case 5 describes the scenario when TT support is provided in the source OLTP and VT is not provided. In this case, it is possible to support TT as well as VT; that is, according to [3] VT may be provided to the DW in addition to TT and DWLT. Here, TT of source is used as VT of DW. This means, by the time the value is entered in source OLTP, the entered value becomes valid until a new value is entered in the system. This is a special case, where a time stamp (VT in this case) can be provided to the DW, even if it (VT) is not available in the source OLTP. The case is defined with "italic font" in the table. For the salary increase example, assume a salary USD 2000 was recorded in the system with TT value 1/5/2014, without specifying VT. Later on, a new salary of USD 2200 was recorded in the system with TT value 1/1/2015 and without specifying VT. In this case, the VT of salary USD 2000 is from 1/5/2014 till 1/1/2015 and the VT of salary USD 2200 is from 1/1/2015 till the new value is recorded in the system.

Case 6 describes the scenario when both time stamps (VT and TT) are provided in the source OLTP. In this case, VT and TT of source are moved to the DW and DWLT is also generated [32]. The presence of these time stamps raises consistency issues. To handle these issues, Bruckner and Tjoa [50] present a conceptual model to describe how to manage temporal consistency. For instance, a salary USD 2000 with VT from the 1/1/2014 to 31/12/2014 was recorded on 1/5/2014 (transaction time TT_1) in the data source. Later on, a new salary of USD 2200 with VT from the 1/1/2015 until "Now" was recorded on 1/1/2015 (transaction time TT_2) in the data source. In this case, when ETL was performed at the end of first quarter (31/3/2014) the value of salary was not known. It is so because the TT of salary USD 2000 was 1/5/2014; that

Table 4: A comparative analysis of various schemes based on supported temporality types.

Approach (last name, year)	Reference	VT	DWLT	$TT_s \rightarrow TT_{DW}$	$TT_s \rightarrow VT_{DW}$
(Blaschka, 2000)	[2]	−	−	−	−
(Abello and Martín, 2003)	[3]	+	−	−	+
(Bębel et al., 2004)	[4]	−	+	−	−
(Koncilia, 2003)	[7]	+	+	−	−
(Sarda, 1999)	[13]	+	−	−	−
(Ravat and Teste, 2000)	[14]	+	−	−	−
(Bruckner et al., 2001)	[15]	+	+	+	−
(Blaschka, 1999)	[17]	−	−	−	−
(Marotta, 2000)	[18]	−	−	−	−
(Letz et al., 2002)	[19]	+	−	−	−
(Kaas et al., 2004)	[20]	−	−	−	−
(Rechy-Ramírez and Benítez-Guerrero, 2006)	[21]	+	+	−	−
(Bliujute et al., 1998)	[29]	−	−	+	−
(Malinowski and Zimanyi, 2006)	[32]	+	+	+	−
(Rizzi, 2007)	[34]	−	−	−	−
(Golfarelli et al., 2004)	[35]	−	−	−	−
(Eder et al., 2003)	[36]	+	−	−	−
(Body et al., 2003)	[37]	+	−	−	−
(Vaisman, 2001)	[38]	−	−	−	−
(Vaisman et al., 2004)	[39]	−	−	−	−
(Pedersen et al., 2001)	[40]	+	−	+	−
(Mendelzon and Vaisman, 2000)	[41]	+	−	+	−
(Chamoni and Stock, 1999)	[42]	+	−	−	−
(Hurtado et al., 1999)	[43]	−	−	−	−
(Solodovnikova, 2007)	[44]	−	−	−	−

is, on 31/3/2014 the value was not yet entered in the system. However, when ETL was performed at the end of second quarter (30/06/2014), the value of salary was available with validity from a previous date (1/1/2014). Although some delay has occurred between VT, TT, and DWLT, however, users can analyse values at different time instants.

2.2. Analysis for Temporal Support in DW. In the initial part of this section, we explain the various time stamps available in the source OLTP and DW. The preceding subsection explains the various cases in which temporal support can be provided in DW due to the dependency of DW on source OLTP. In this subsection, we analyse and compare the existing studies that support these time stamps. For the choice of the existing studies, we searched through major academic databases using several keywords such as temporal DW, temporal structures in DW, multi-version DW, schema evolution in DW, change management in DW, and DW maintenance. The retrieved articles were assessed for their relevance to the scope of the study by employing a *relevance screening procedure* [51]. A comparison of the studies selected for temporal support in DW is presented in Table 4.

In Table 4, the studies considered for comparison are listed in *first* column. The *second* column explains whether VT of the source OLTP is explicitly included in the DW or not. The *third* column represents whether TT of source is loaded in the DW as TT of source. The value "+" represents that TT from source is loaded in the DW as TT. In this case, it would be possible to analyse when the value was changed in the source OLTP. For instance, in case of salary increase example, when the salary amount was recorded in the source OLTP, the value "–" means that TT is not available in the DW; instead a new value is generated as DWLT. In this case, it is not possible to analyse when the value of salary was recorded in source but it is possible to record when the value was loaded in the DW. The *fourth* column represents whether TT of source is used as VT in the DW. The value "+" means that TT from source is loaded in the DW as VT. In this case, it would be possible to analyse the true value of salary based on TT recorded in source. The value "–" represents that the scheme does not allow TT to be used as VT in the DW. In this case, it is not possible to analyse the validity time of salary and misleading aggregates may be generated.

From Table 4, it can be seen that nearly half of the studied schemes support VT and the other half do not. For the schemes that do not support VT, it would be challenging to avoid the generation of misleading aggregates. Besides that, DWLT can be used as a time stamp even if there is no time stamp in the source OLTP. However, it can be observed that majority of the schemes do not support DWLT. For these approaches, it will not be possible to make immediate decisions and informed adjustment about the timing of DW instantiation (i.e., adjusting load frequency). In some cases, the decision maker may get delayed information. For

instance, in the sales company example it will not be possible to immediately know the rise or fall in sales, as they happen, until the next quarter. Therefore, the decision of at what time stocks should be made available may not be taken in time. Another observation in the usual practice is to ignore TT coming from sources. However, in this way traceability application such as fraud detection cannot be implemented. Therefore, it is discouraged to ignore the TT coming from source in cases where it may be required to retrack events that might happen. Abello and Martín [3] transform TT from a source system to represent VT in the TDW. This is semantically incorrect because data may be included in a database after their period of validity has expired, for example, clients' previous region.

Observing the horizontal trends reveals that if a scheme supports TT then it also supports VT; and if VT is not supported then all other time stamps are not supported. This indicates that the existing schemes support temporality prioritizes VT because it may generate misleading aggregates.

There are, however, fewer studies [15] that support all time stamps. These are effective techniques and by using them correct aggregates can be made available, events can be retracked, and adjustments to loading frequency can be made.

3. Handling Changes in DW Metadata

In Section 2.1 we elaborated the dependency of DW on OLTP for various time stamps, as six cases. Similarly, if the source OLTP is changed over time, the DW may cause maintenance anomalies [52] and adjustments to the DW may be required. The development, instantiation, and maintenance are for the sake of facilitating decision makers who query the DW for decision support. Therefore, besides DW handling, querying data in the presence of schema and content changes becomes a nontrivial task.

Various solutions regarding DW adjustment are available in order to handle *schema* and *data changes* in source OLTPs. Another reason for adjustments to a DW could be evolving business requirements. For instance, measures may become obsolete, a dimension level is removed, new level is added, or temporal granularity is changed. In the sales company example, removal of date and week from time dimension is an example of change in temporal granularity. This change can be observed in Figures 2(a) and 2(b). Total revenue in Figure 2(a) becomes obsolete in Figure 2(b) which is an example of measure becoming obsolete. Similarly, examples of removal of dimension level and addition of a new level can be seen in Figures 2(a) and 2(b).

Eder and Wiggisser (2010) describe two schemes of adjusting DW: (a) updating the schema, transforming the existing data into the new schema, discarding the schema, and using the updated schema for future data population: these are called *evolution schemes* [17]; (b) the second scheme consists of creating a new schema, maintaining both schemas, and using only the new schema for future data population [24]. However, both schemas can be used for retrieval from the DW. These are called *versioning schemes*. Similar to the temporal cases, there are six cases for change handling schemes as shown in Table 5 and they are thereafter explained.

Case 1 describes the scenario when both DW schema and data are changed due to change in the source OLTP. In this case, both *evolution* and *versioning* schemes can be used. If *evolution scheme* is used, the DW schema is changed, the existing data is transformed to a new schema, and the previous schema is discarded. A key limitation of this case is that schema changes are not available for the use of decision makers and data changes may or may not be available to users. For the sales company example, consider Figure 2(b) is changed in a way that brand is removed *(schema change)* and the data about brand is also discarded *(data change)* to form Figure 2(c). After this evolution, the attribute brand as well as the data in the attributes will not be available to users, meaning that QtySold and the ProfitEarned cannot be analysed with respect to brand. However, if a *versioning scheme* is used, a new DW schema is created and both schemas are maintained, but for future instantiation Figure 2(c) will be used. For the sales company example, for the change of discarding brand, the Figure 2(b) schema as well as the Figure 2(c) schema will be maintained. After that, it will still be possible to analyse QtySold and ProfitEarned with respect to brand for time t_2. However, subsequently this analysis will not be available to users.

Case 2 describes the scenario when DW schema is changed but data is not changed. In this case, both *evolution* and *versioning* schemes can be used. If an *evolution scheme* is used, the existing data is transformed to a new schema and the previous schema is discarded. However, this change does not require any data change in the DW. Similar to the preceding case, schema changes are not available to decision makers but data changes are not needed. For the sales company example, consider Figure 2(a) is changed in a way that state is added *(schema change)* to form Figure 2(c), but the data in the DW does not require any change *(no data change)*. After this evolution, the newly added attribute will be available for future instantiation. This means that the two measures, QtySold and ProfitEarned, cannot be analysed with respect to state at this moment, but the analysis of these measures will be available after future instantiation. If a *versioning scheme* is used, a new DW schema is created and both schemas are maintained, and Figure 2(b) schema will be used for future instantiation. For the sales company example, at this moment it will not be possible to analyse the two measures, QtySold and ProfitEarned, with respect to state, but the analysis of these measures will be possible after future instantiation.

Case 3 describes the scenario when DW schema is not changed but data is changed. This is the case in which new records are added to DW, as a result of ETL process. If that is the case then both *evolution* and *versioning* schemes may not be needed.

Case 4 describes the scenario when DW schema is not changed but data is changed. This is a special possibility of *data change* that may lead to *schema change*. For such a change, Case 3 is converted to Case 1, where both schema and content are changed. In [4] a solution was presented to handle this special possibility of *data change* as multiversion data

Table 5: DW changes and change handling schemes.

Cases	DW changes		Change handling scheme	
	Schema	Data	Evolution	Versioning
1	Yes	Yes	Yes	Yes
2	Yes	No	Yes	Yes
3	No*	Yes	No	No
4			Yes	Yes
5	No	No	No	No
6			Yes*	Yes*

*Special interpretation (see the case description).

warehouse. Due to this distinction the value of this schema change is marked with (∗).

Case 5 describes the scenario when DW schema and data changes do not happen. In this case *evolution* and *versioning* schemes are not needed. However, DW becomes a snapshot; that is, no change of content happens in DW. In this case, the analyses of measures gradually become obsolete, making it less useful for users. For the sales company example, at this moment it will be possible to analyse the two measures, QtySold and ProfitEarned, with respect to state, till the last update, say 31/6/2015. However, the values of these measures will become obsolete if they are not updated afterwards, making it less useful for users.

Case 6 describes the scenario when DW schema and data changes do not happen; however both *evolution* and *versioning* schemes can be used. In this case, the reason of change is evolving business requirements that require adjustments to DW. The evolving business requirements could be, for instance, changed temporal granularity, removal of dimension level, and addition of a new level. These evolving business requirements may lead to changes in DW schema. Due to this distinction, the value of this schema change is marked with (∗).

In the remaining part of this section, we further elaborate the *evolution* and *versioning* schemes with an illustrative example for *schema* or *data* changes and compare the existing approaches of the corresponding scheme.

3.1. Evolution Schemes. *Evolution* schemes support only a single version of a DW schema (i.e., the current version) for both *schema* and *data* changes. In evolution, first the schema is updated and then data is transformed from the old schema into the new schema, and the old schema is discarded. However, it involves high maintenance cost. These schemes time stamp the data coming at different time periods and store them in the new schema. The studies [2, 13, 17–20, 39, 43] describe some of the approaches that may be used for this purpose. A comparative study of the schemes for managing *schema* and *data changes* requires the analysis of various parameters. Before carrying out a comparison of the different schemes, we describe the operations corresponding to schema and data changes.

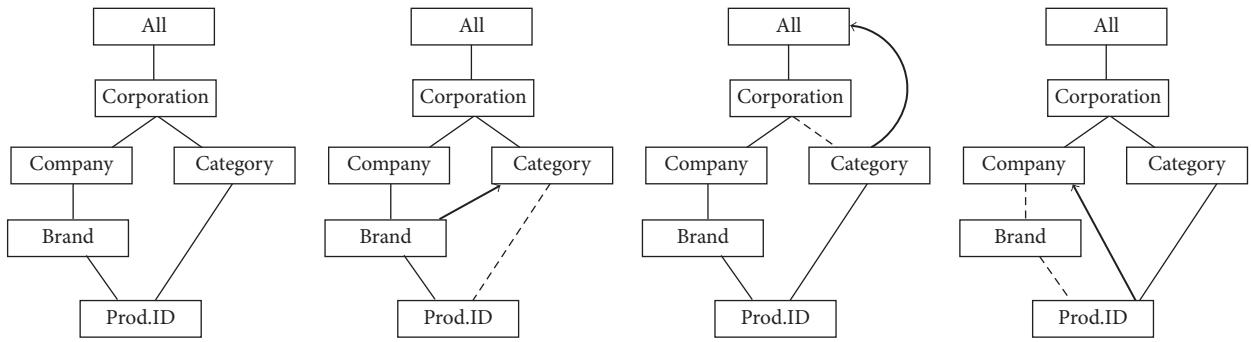
Schema Changes. *Schema changes* such as addition of weekly analysis modify the source OLTP schema and this change can also result in adjustment of the structure of the DW. An illustrative example of a schema change can be seen by comparing the time dimension of Figures 1 and 2(a). *Schema changes* can be creation/deletion of dimension, fact, hierarchy, level, member attribute, measure, and level movement in the hierarchical structure. These changes are later used to compare existing evolution schemes, as shown in Table 6. The five atomic update operators to tackle these changes are *relate levels*, *unrelate levels*, *delete level*, *generalize*, and *specialize*. These operations are explained and illustrated in the remaining part of this section. For illustration, we rely on the working example of a sales company presented in Section 1.2. Specifically, an excerpt of the sales company example, that is, product dimension only, shown in Table 2, is used to illustrate the atomic operators.

The *relate levels* operator specifies a roll-up operation between two dimension levels. The change is done in such a way that all the levels before the change are still reachable. In Figure 4(b), brand and category are the two levels of the product dimension. The creation of a relation between brand and category, represented by an arrow, is an example of the *relate levels* operator. Besides that, Prod.ID and category relation are removed as a collateral action, because it should still be possible to reach category from Prod.Id via brand. This removal of relation is represented by dotted line in Figure 4(b).

The *unrelate levels* operator removes a relationship between two levels. Similar to the previous case, the change is done in such a way that the levels before the change are still reachable. In the example in Figure 4(c), to unrelate the category and corporation levels, the relation between the category and corporation levels is deleted. To ensure reachability, a new relation between category and all is created. Because of this new relationship, it is still possible to reach all even when the relation between category and corporation has been removed.

The *delete level* operator removes a level and its relationship with other levels. The change is done in such a way that the levels above the removed level are still reachable. In the example in Figure 4(d), the delete operation on the brand level deletes the relations between Prod.Id and brand and brand and company. It then creates a new relation between Prod.Id and company to reach the company level that was originally reachable through the brand level.

The *generalize* operator adds a new level and rolls up an existing level. Finally, the *specialize* operator creates a level



(a) Schema of dimension product (b) Relate operation between brand and category levels (c) Unrelate operation between category and corporation levels (d) Delete operation on brand level

Figure 4: Structural update operations.

Table 6: A comparative analysis of evolution schemes for schema changes.

Evolution schemes (last name, year)	References	Level of changes		Schema level changes						
		Schema	Data	Dimension	Fact	Hierarchy	Level	Attribute	Measure	Level movement
(Blaschka, 2000)	[2]	+	+	+	+	−	+	+	+	−
(Sarda, 1999)	[13]	+	+	+	+	−	−	+	+	−
(Blaschka et al., 1999)	[17]	+	+	+	+	−	+	+	+	−
(Marotta, 2000)	[18]	+	+							
(Letz et al., 2002)	[19]	+	+	+	−	+	+	+	−	−
(Kaas et al., 2004)	[20]	+	+	+	−	−	+	+	+	−
(Rechy-Ramírez and Benítez-Guerrero, 2006)	[21]	+	−	+	+	+	+	+	+	+
(Bliujute et al., 1998)	[29]	−	+	+	−	+	−	+	−	−
(Vaisman, 2001)	[38]	+	+	+	−	−	+	−	−	−
(Vaisman et al., 2004)	[39]	+	+	+	−	+	+	+	−	−
(Hurtado et al., 1999)	[43]	+	+	+	−	+	+	+	−	−

Note. In [18], changes from source schema and support changes in attribute and relations are adapted. Rest of the schemes support changes arise due to changing business requirements.

and drills it down to the lowest level, thus making the lowest level in the dimension hierarchy.

Table 6 provides a comparative analysis of various evolution schemes based on their support for *schema changes*. The "+" sign in the table represents that the type of change is handled by the proposed approach and "−" sign represents that the type of change is not handled by the proposed approach. These parameters are the elements of the metamodel of DW design as described in [53]. The elements indicate the level where evolution support is provided in the DW. The elements used as parameters are dimension, fact, hierarchy, level, attribute, and measure.

The table shows that majority of the schemes support *schema* and *content changes*. Among these schemes, in [21] a conceptual model is proposed for DW schema that facilitates the modification of a DW schema in an implementation independent manner, without affecting its operations. Similarly, authors in [29] handle slowly changing dimensions as well as state-oriented data without fundamentally changing the design. Letz et al. [19] on the other hand elaborate how update operations on dimensions can be utilized to preserve consistency, while Sarda [13] and Hurtado et al. [43] present a formal model for defining the schema and a primitive operator for dimension updates. A key deficiency in the majority of these schemes is that they focus on handling changes to dimension while ignoring aggregates (measures), although the real use of a DW is associated with aggregates. This of course does not mean that dimension updates are not significant but instead we contend that sufficient attention is not paid to aggregates. Sarda [13] is among the important exceptions, which explicitly defines operators for modification to facts such as insertion and deletion of fact or insert dimension level to fact.

Data Changes. *Data changes* such as data insertion of new products in the data sources do not modify the source OLTP schema. However, such changes can modify the structure of a DW. An illustrative example to this effect is given in Tables 1 and 2 where allocation region of staff S_2 is changed from AR1 to AR2. The handling of this change requires

adjustments to the DW schema. Instance level changes include the *transformation*, *merging*, *splitting*, *reclassification*, *creation*, and *deletion* of member(s). For instance, a brand may be *split* into two or more brands, or two brands may be *merged* into one brand. Similarly, a member may be *transformed* due to change in an attribute name or meaning, or a dimension member may be *reclassified* in the dimension structure. Furthermore, a brand can be *added* or an existing brand can be *deleted*. These changes are used to compare existing evolution schemes in terms of their ability to handle *data changes*.

Table 7 provides a comparative analysis of various evolution schemes based on their support for *instance changes*. For comparison, we mainly use the possible set of changes that can be made at *instance level*, as discussed in the previous paragraph. These are member creation, deletion, transformation, merging, splitting, or reclassification. In addition, we also consider whether dimensions are shared or not.

From the table we observe that at *instance levels* transformation, merging, and splitting are not supported by existing schemes. Also, fewer dimensions support constellation schema, that is, sharing of dimensions with multiple fact tables. However, some approaches such as [19] support creation and deletion of dimension members while preserving consistency. This is done by analysing schema, detecting the levels where conflicts may occur, and performing modification by insertion or reclassification.

3.2. Versioning Schemes. Versioning schemes support multiple versions of the DW schema. Versioning may be implicit or explicit.

Versioning schemes support multiple versions of a DW schema for both *schema* and *data* changes. In versioning, the schema is changed to a new schema while maintaining both schemas. For future instantiation the new schema is used, but both schemas can be used for querying. Versioning does not involve high maintenance cost compared to evolution, but querying and presenting results from multiple versions become a challenging task that also affects the interpretation of results (details in Section 3.2.1).

According to Solodovnikova [44], there are two ways of versioning a DW, *implicit* versioning and *explicit* versioning. In *implicit* versioning, two versions of schema are maintained and transformation functions are defined to record adjustments in the DW [54]. In *explicit* versioning, two versions of *schema* as well as *data* are maintained [48]. Each version represents a schema and data version. Due to the existence of versions, the comparison of versioning schemes based on the DW design elements (used in Section 2) is not interesting. On the other hand, due to the existence of several versions the bigger challenge is querying the DW and interpretation of results. Also, in the presence of versioning the focus is shifted from data analysis only to data as well as change analyses. Therefore, for the comparison of versioning schemes we rely on another set of elements such as cross-versioning query, augmented schema, and multiversioned facts. Below, we discuss the two types of versioning schemes separately.

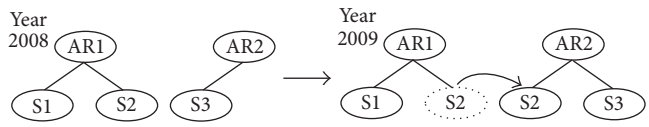


Figure 5: Evolution of hierarchical structure for the dimension staff.

3.2.1. Implicit Versioning. In *implicit* versioning schemes, versions are recorded implicitly by providing temporal extensions to the DW schema only. Chmiel [45] proposed a bitmap approach for sharing data between multiple versions. However, Rizzi [55] argues that the use of bitmap approach limits what-if analysis, because processing queries to retrieve the data that is shared across multiple versions is a challenging task. Due to the challenging nature of querying versions, a comparative study of these schemes requires the analysis of various parameters regarding querying DW. A query to a multiversion DW (with implicit versioning) can be a version slice query that searches for data within a single version, or a historical query that searches across evolved versions, or a combination of both. Cross-version queries span over multiple schema versions. Golfarelli et al. [25] propose the use of augmented schema to increase flexibility while querying cross-version data. According to this approach, whenever a new schema version is generated, an augmented schema is made to represent the new schema and extends the previous schema. The augmented schema is stored together with *schema* versions. Other than augmented schema, some approaches, for example, [21], propose an extension to SQL for querying a DW. Once data has been retrieved, the interpretation of results is also challenging. It is so because in the presence of versioning the focus is shifted from data analysis only to data as well as change analyses.

For the sales company example, the staff member S_2 in the staff dimension has been reallocated from allocation region AR1 to AR2 between 2012 and 2013, as shown in Figure 5. Assume data for the years 2012 and 2013, as given in Table 8.

The sample query "Find out the total amount of sales per region and year" can be interpreted in three different possible ways. The *first* interpretation gives the sales amount corresponding to each particular region, as shown in Table 9(a). The *second* interpretation returns the sales amount based on the consideration that the allocation regions have always been structured as these were in 2012, as represented in Table 9(b). Finally, the *third* interpretation returns the sales amount based on the assumption that allocation regions have always been as these are now in 2013, as shown in Table 9(c). The up, down, and horizontal arrows in the "evolution" column show whether sales for an allocation region has gone up, come down, or stayed the same from years 2012 to 2013.

The above example shows that the results may greatly vary or may even be contradictory, depending on query interpretation. Therefore, it is necessary to guide the end-user about the possible interpretation choices. Let us assume that the allocation region AR1 has been split into two regions AR11 and AR12 during the year 2014, as shown in Table 10.

The same query "Find out the total amount of sales per region and year" can again be interpreted in three

Table 7: A comparative analysis of evolution schemes for data changes.

Evolution schemes (last name, year)	References	Instance level changes						Shared dimensions
		Member creation	Member deletion	Transformation	Merging	Splitting	Reclassification	
(Blaschka, 2000)	[2]	−	−	−	−	−	+	+
(Sarda, 1999)	[13]	+	+	−	−	+	+	−
(Letz et al., 2002)	[19]	+	+	−	−	−	+	−
(Kaas et al., 2004)	[20]	+	+	−	−	−	−	−
(Rechy-Ramírez and Benítez-Guerrero, 2006)	[21]	−	−	−	−	−	−	−
(Bliujute et al., 1998)	[29]	+	−	−	−	−	−	−
(Vaisman, 2001)	[38]	+	+	−	−	−	−	−
(Vaisman et al., 2004)	[39]	+	+	−	−	−	−	−
(Hurtado et al., 1999)	[43]	+	+	−	−	−	−	−

Table 8: Sample data for the dimension staff.

Year 2012			Year 2013		
Allocation region	Staff	Sales	Allocation region	Staff	Sales
AR1	S1	100	AR1	S1	100
	S2	50	AR2	S2	100
AR2	S3	100		S3	50

The word sales is an alias of QtySold measure.

Table 9: Query interpretations in 2012-2013.

(a) Query interpretation I

Allocation region	2012	2013	Evolution
AR1	150	100	↓
AR2	100	150	↑

(b) Query interpretation II

Allocation region	2012	2013	Evolution
AR1	150	200	↑
AR2	100	50	↓

(c) Query interpretation III

Allocation region	2012	2013	Evolution
AR1	100	100	→
AR2	150	150	→

different possible ways. The *first* interpretation gives the sales amount corresponding to each particular region, as shown in Table 11(a). The *second* interpretation returns the sales amount based on the consideration that the allocation regions have always been structured as these were in 2013, as shown in Table 11(b). This correspondence is straightforward and acquired just by appending the sales of allocation regions AR11 and AR12. The *third* interpretation returns the results based on the consideration that the allocation regions have always been as these are now in 2010, as shown in Table 11(c). Here, additional information is needed to calculate the amount of sales for the allocation regions AR11 and AR12 during the year 2013. The additional information may be given in the form of estimated percentage.

The above example shows that the first interpretation results are less detailed but show the true data as compared to the last interpretation where approximations are made. However, it is unable to facilitate the evolution of data and, therefore, data is mapped onto the latest structure version. Furthermore, it also shows the requirement of distinguishing mapped data from source and, finally, data reliability.

For analysis and interpretation, both the detailed and aggregated data can be used. Since detailed data is required for routine decision-making tasks, therefore data should be current and access latency should be minimized. Facilitating such support in DWs is a major task [6]. It is so because the delay in the discovery of real-world changes results in delayed propagation of these changes to the DW. Hence, DWs may suffer from the problem of temporal consistency when the real-world changes are discovered after a delay. For consistency reasons, analytical applications require temporal components in the data model. A temporally consistent information representation is a stable view of historical data at any time regardless of propagation delays. Temporally consistent information is therefore a key aspect of comparison.

For comparative study of the schemes that implicitly record DW versions, various parameters of analysis should be considered. But, due to the existence of versions, the DW design elements (used in Section 2) based comparison is not interesting. Instead, from the above discussion we conclude the bigger challenges are querying DW, interpretation of result, and availability of temporally consistent information. This is the case because in the absence of versions the

Table 10: Allocation region AR1 split into AR11 and AR12.

Year 2013			Year 2014		
Staff	Allocation region	Sales	Staff	Allocation region	Sales
S1	AR1	100	S1	AR11	150
				AR12	50

Table 11: Query interpretations 2013-2014.

(a) Query interpretation I

Allocation region	2013	2014	Evolution
AR1	100	—	?
AR11	—	150	?
AR12	—	50	?

(b) Query interpretation II

Allocation region	2013	2014	Evolution
AR1	100	200	↑

(c) Query interpretation III

Allocation region	2013	2014	Evolution
AR11 (40% of AR1)	40	150	↑
AR12 (60% of AR1)	60	50	↓

decision-making users do not have access to data across versions and, therefore, the types of analyses that can be supported are limited. Corresponding to these challenges, for the comparison of versioning schemes we rely on the following parameters: *cross-version queries*, *augmented schema*, *multiversioned fact table*, and *temporally consistent representation* of information (TCR). Table 12 provides a summary of the comparative analysis of the various schemes based on the above-mentioned parameters.

From the table, it can be observed that majority of the schemes support schema and data changes. Also, it can be seen that no study proposes the use of augmented schema because these schemes implicitly record temporal extension and data is shared, for instance, by the bitmap approach [45]. This limits the "history of change" analysis. However, some studies such as [33, 41, 44] recognize the challenge of querying data when data of interest is distributed and shared across version. Among these, authors in [33] provide temporal attributes to all elements of the DW metamodel and support cross-version queries, as well as fact-constellation schemes. Chmiel [45] on the other hand focuses on optimization of queries. Another observation is that a couple of approaches, [37, 38], support multiversioned fact table. Body et al. [37] support schema and instance level changes and complex hierarchical structures. Also, the study introduces the notions of confidence factor, temporally consistent representation of information (TCR), multiversion fact table, temporal dimension, and temporal relationship. The value of a confidence factor distinguishes the mapped data from source and describes the reliability of the data. For instance, one can describe the range for the values of confidence factors as source data, temporally consistent data, exact or approximated mapped data, or unknown mapping relationship.

3.2.2. Explicit Versioning. In explicit versioning schemes, DW versions are maintained explicitly and result in better query performance, as these schemes do not require schema transformation functions. Such schemes handle changes in the DW *content* and *structure* through the use of a multiversion DW (MVDW), which represents *schema* and *data* at a particular time instant. One approach to deal with data versions is to physically store a copy of data in each version of a DW. As DW versions are explicitly stored and require no transformation functions, the cross-version queries run faster. But this approach is not appropriate if the size of a DW is in terabytes and a tremendous amount of additional disk storage is required for storing data for each version. Thus, a time-space trade-off exists between query performance and disk storage. Moreover, it causes update anomalies, data redundancy, and system overhead for consistently managing the multiple data copies.

Explicit versioning schemes support the what-if functionality as they explicitly maintain record of all schema versions by physically keeping different data versions. A DW can adopt the change operations like insert, update, and delete. Furthermore, complex operations such as split, merge, and move can also be performed using these basic operations. Figure 6 shows how changes are handled through these operations. In structure version SV_3, the dimension member Cat_2 has split into Cat_3 and Cat_4 by using the delete operation for Cat_2 and insert operation for Cat_3 and Cat_4. In SV_4, dimension members Cat_3 and Cat_4 are merged together to form member Cat_2. The goal of merging has been achieved through two operations, delete for Cat_3 and Cat_4 and insert for Cat_2.

Similar to implicit versioning schemes, for comparative study of explicit versioning schemes we rely on the following parameters: *cross-version queries*, *augmented schema*, *multiversioned fact table*, and *temporally consistent representation* of information (TCR). Table 13 provides a summary of the comparative analysis of the schemes that explicitly store data of every DW version.

From the table we observe that two studies, [25, 34], implement the use of augmented schema to support cross-version queries. Golfarelli et al. [25] provide proofs that an input query can be mapped over versions and discusses the summarizability issues with disaggregation when navigating a DW. Specifically, it uses graphs to represent a DW schema and defines algebra of schema operations. Whenever a new version is created, an augmented schema is created to increase flexibility in cross-version querying. It discusses consistent movement of data between schema versioning. Further, authors of [34] discuss the development of a prototype for

Table 12: Approaches supporting implicit DW versions.

TDW approach (last name, year)	Reference	Level of changes		Versioning support			
		Schema	Data	Cross-version queries	Augmented schema	MV fact table	TCR
(Koncilia, 2003)	[7]	+	+	−	−	−	−
(Solodovnikov et al., 2015)	[10]	+	+	−	−	−	−
(Manousis et al., 2015)	[24]	+	+	−	−	−	−
(Malinowski and Zimanyi, 2006)	[32]	+	+	−	−	−	−
(Eder et al., 2002)	[33]	+	+	+	−	−	−
(Body et al., 2003)	[37]	+	+	−	−	+	+
(Vaisman, 2001)	[38]	+	+	+	−	+	+
(Mendelzon and Vaisman, 2000)	[41]	+	+	−	−	−	−
(Chamoni and Stock, 1999)	[42]	+	+	−	−	−	−
(Solodovnikova, 2007)	[44]	+	−	+	−	−	−
(Chmiel, 2010)	[45]	+	+	+	−	−	−
(Kang and Chung, 2002)	[46]	+	−	−	−	−	−
(Quass and Widom, 1997)	[47]	+	+	−	−	−	−

Note. In [44] both implicit and explicit versioning are supported.

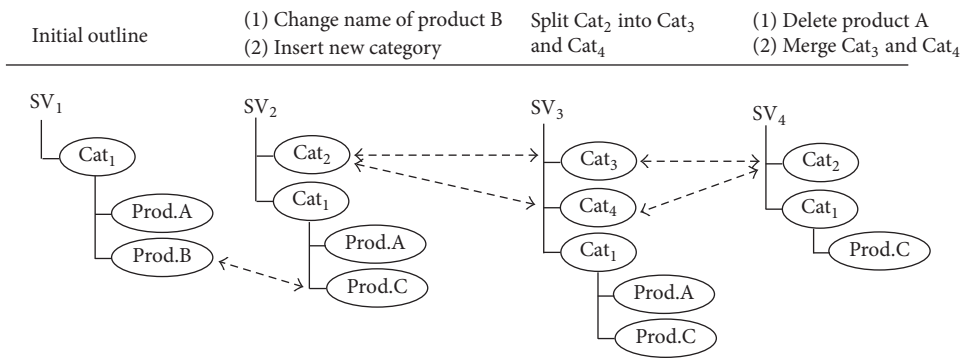


Figure 6: Operations to handle structural changes.

managing and query versions in a DW. The prototype is based on the concept of augmented schema to increase cross-version querying flexibility.

Another observation is that two studies, [5, 21], extend SQL for querying multiple versions of a DW. Rechy-Ramírez and Benítez-Guerrero [21] propose an SQL-like language that allows users to express evolution requirements. The scope of the language is limited to creation and changing version of cube. Morzy and Wrembel [5] on the other hand extend the SQL language and build an interface to express cross-version queries. The query is split into several independent partial queries, each executed on its particular DW version. Later, the results of partial queries are combined to get the required results. Also, they develop a GUI to visualize queries and results. The benefit of query extension over augmented schema is that it provides flexible query and more control over query specification, whereas it increases the complexity of query specification at the same time.

4. Commercial Data Warehousing Tools

There are several commercial DW systems and OLAP tools available in the market. In terms of market value, the most important ones are IBM DB2, SAP Business Warehouse, Oracle Express Server, Ingres Decision Base OLAP Server, NCR Teradata, Sybase Adaptive Server Enterprise, and Hyperion Essbase OLAP Server. Table 14 shows a comparison of tools in terms of their support for handling changes to DW. Similar to the previous tables, the "+" sign in the table indicates that the support for the respective change is available in the tool, "−" sign indicates that the support for the change is not available in the tool, and "+/−" sign indicates that the support for the

Table 13: Approaches supporting explicit DW versions.

TDW approach (last name, year)	Reference	Level of changes		Versioning support			
		Schema	Instance	Cross-version queries	Augmented schema	MVFact table	TCR
(Solodovnikova, 2007)	[44]	+	−	+	−	−	−
(Rizzi and Golfarelli, 2007)	[34]	+	+	+	+	−	−
(Wrembel and Bębel, 2007)	[48]	+	+	−	−	−	−
(Golfarelli et al., 2006)	[25]	+	+	+	+	−	−
(Rechy-Ramírez and Benítez-Guerrero, 2006)	[21]	+	+	−	−	−	−
(Morzy and Wrembel, 2004)	[5]	+	+	+	−	−	−
(Hai et al., 2016)	[1]	+	+	−	−	−	−
(Bellahsene, 1998)	[49]	+	−	−	−	−	−

Table 14: Commercial tools support for DW changes.

Tools	Evolution scheme		Versioning scheme		
	Schema change	Data change	Schema change	Data change	Cross-versioning queries
IBM DB2	+	+/−	−	−	−
SAP Business Warehouse	+	+/−	+/−	−	−
Oracle Express Server	+	+/−	+/−	−	−
Ingres Decision Base OLAP Server	+	+/−	−	−	−
NCR Teradata	+	+/−	−	−	−
Sybase Adaptive Server Enterprise	+	+/−	−	−	−
Hyperion Essbase OLAP Server	+	+/−	−	−	−

change depends upon the type of change; that is, the support for the change may or may not be available in the tool

The table marks that all tools support evolution of DW schema. It is because the evolution approach does not require any additional functionality in the commercial tools; that is, in evolution DW schema is updated and transferred from old to new schema. This functionality is available in all DW systems. Also, it can be seen from the table that majority of these tools do not provide support to manage versions of DW schema and support for cross-version queries. However, SAP Business Warehouse provides support to track dimensional data changes and allows users to choose a version of the hierarchies for querying, whereas commercial tools still provide marginal support to schema changes, like SQL Compare, which can compare and synchronize SQL Server database schema and can push schema changes of a local database to a remote database. Oracle's what-if analysis uses the model clause to express hypothetical analysis or create the hypothetical rankings of records in a query [56]. Oracle Change Management Pack compares database schema and tracks metadata evolution and allows script generation and execution to carry out the necessary changes. However, cross-version querying support is not available in either SQL Compare or Oracle Change Management Pack.

5. Conclusions and Future Work

In this study, we analyse temporal data management and how to handle changes in a DW design, as well. To this end, a taxonomy of existing approaches has been built. According to the taxonomy, for clearly distinguishing the features of existing approaches, *temporal support* in a DW is separated from *handling the changes* as well as from their *support in commercial tools*. Subsequently, we have identified a large number of DW design schemes in literature and analysed their abilities to support temporal aspects and handling changes.

The temporal data in a DW enables us to perform what-if analysis. Time stamps, such as valid time (VT), transaction time (TT), and DW loading time, are used to capture time-varying states. The temporal data model represents original and transformed data items and their structure and time stamps. Although associating time validities with dimensions and facts can fulfil temporal requirements, however, it is useful to define the dimension "time" in a DW for explicit representation of calendars, time hierarchies, and events. Our study of existing design schemes reveals that only half of the studied schemes support VT. Therefore, the use of these approaches is likely to generate misleading aggregates. Further, the study reveals that majority of the schemes do

not support TT and DWLT which limits the analyses support offered by DW. However, some design schemes support all the time stamps. We argue that these design schemes should be considered for implementing DW and particularly for organizations who are sensitive towards change.

It is widely known that a number of external factors play a pivotal role in compelling organizations to change, which enforces changes in operational data sources as well as changes to DW. For DW, these changes can be of two types, *schema changes* and *content changes*. The *schema changes* include creation and deletion of a dimension, hierarchy, level, measure, or member attribute and level movement in the hierarchical structure. The *content changes* include creation of a new dimension member, deletion of an old dimension member, member merging, splitting, transformation, or reclassification of a member. We have first-hand experience of observing changes to organizations which resulted in changes to DW. To enhance the understanding of reader, below, we briefly present simple but diverse examples of changes to DW and provide an overview of how these changes were handled in DW. The examples are as follows. (a) A manufacturing company started manufacturing new products to meet marketing demands. This change led to content change in the DW; that is, data about each of the new products were added to the dimension table and no changes were made to the structure of the dimension table. (b) The company constructed a new production unit at a tax-relaxed industrial zone. This change resulted in schema changes in DW; that is, a new dimension member was added in order to analyse and compare the production performance of multiple units. (c) The company transferred selected employees to the new unit. This change resulted in schema change as well as content changes in the DW. As a schema change, a new member attribute was added to a dimension table indicating the unit to which the employee belongs. As a content change, employees were assigned to the new unit.

There are two types of schemes to handle schema and content changes in DW. These are *evolution* and *versioning* schemes. The evolution schemes support a single version of DW schema for both schema and content changes. In evolution, the DW schema is updated and data is transferred from old to the new schema. The benefits of evolution schemes are as follows: querying mechanisms are not changed and the available commercial tools can be used. However, the schemes lack preserving history and maintenance cost is high. In contrast to that, the versioning schemes support multiple versions of DW schema for each schema change and the content change, which cannot be accommodated in the existing version. The benefit of version schemes is that the history is preserved. However, retrieval of data requires writing cross-version queries which cannot be performed by using the available commercial tools. We thus suggest the trade-offs of the two schemes (evolution and versions) should be carefully examined. Our comparison of the change handling schemes reveals some schemes have been designed to deal with changes to dimensional data and its support is partially available in a couple of tools; however, no common framework has been proposed for schema and factual data changes.

Based on the synthesis we identify the following directions for future academic work: (a) the use of data mining techniques is a scalable approach to detect structural changes in DWs, (b) the exploration of self-adaptive methods to detect structural changes in DWs is an open research area, (c) a common framework for retrieving and presenting data in a temporally consistent manner, and (d) the support of cross-version queries and their impact on interpretation of results needs further exploration. For the practitioner, the study established that no commercial DW provides support for cross-version queries. The querying support can be provided for monitoring different types of changes to support strategic decision-making.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W. Hai, Z. Zeshui, H. Jujita, and L. Shousheng, "Towards felicitous decision making: An overview on challenges and trends of Big Data," *Information Sciences*, vol. 367-368, pp. 747–765, 2016.

[2] M. Blaschka, *FIESTA: A framework for schema evolution in multidimensional databases [Ph.D. thesis]*, Technische Universitat Munchen, Munchen, Germany, 2000.

[3] A. Abello and C. Martín, "The data warehouse: an object-oriented temporal database," in *Proceeding of the 8th Conference on Jornadas Ingeniería del Software y Bases de Datos (JISBD'03)*, pp. 675–684, Alicante, Spain, 2003.

[4] B. Bębel, J. Eder, C. Koncilia, T. Morzy, and R. Wrembel, "Creation and management of versions in multiversion data warehouse," in *Proceedings of the Applied Computing 2004 - Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 717–723, New York, NY, USA, March 2004.

[5] T. Morzy and R. Wrembel, "On querying versions of multiversion data warehouse," in *Proceedings of the 7th ACM International Workshop on Data warehousing and OLAP (DOLAP'04)*, pp. 92–101, New York, NY, USA, 2004.

[6] W. Ahmed, E. Zimanyi, and R. Wrembel, "Modelling Data warehouses with multiversion and temporal functionality," in *Proceedings of the 5th European Business Intelligence Summer School (eBISS'15)*, pp. 1-2, Barcelona, Spain, 2015.

[7] C. Koncilia, "A bi-temporal data warehouse model," in *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, pp. 77–80, Klagenfurt, Austria, 2003.

[8] W. Oueslati and J. Akaichi, "Querying a multi-version trajectory data warehouse," *International Journal of Business Information Systems*, vol. 21, no. 4, pp. 403–417, 2016.

[9] M. Golfarelli and S. Rizzi, "A survey on temporal data warehousing," *International Journal of Data Warehousing & Mining*, vol. 5, no. 1, pp. 1–17, 2009.

[10] D. Solodovnikov, L. Niedrite, and N. Kozmina, "Handling Evolving Data Warehouse Requirements," in *Proceedings of the East European Conference on Advances in Databases and Information Systems (ADBIS)*, vol. 539, pp. 334–345, Springer CCIS, 2015.

[11] G. Garani and C. E. Atay, "Comparison of different temporal data warehouses approaches," *The Online Journal of Science and Technology*, vol. 7, no. 2, pp. 17–27, 2017.

[12] W. Ahmed and E. Zimanyi, "Querying multiversion data warehouse," in *Proceedings of the East European Conference on Advances in Databases and Information Systems (ADBIS)*, vol. 539, pp. 346–357, Springer CCIS, 2015.

[13] N. Sarda, "Temporal issues in data warehouse systems," in *Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 27–34, Kyoto, Japan.

[14] F. Ravat and O. Teste, "A temporal object-oriented data warehouse model," in *Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA'00)*, pp. 583–592, London, UK, 2000.

[15] R. M. Bruckner, B. List, J. Schiefer, and A. M. Tjoa, "Modeling temporal consistency in data warehouses," in *Proceedings of the 12th International Workshop on Database and Expert Systems Applications, DEXA 2001*, pp. 901–905, Munich, Germany, September 2001.

[16] W. Ahmed, E. Zimanyi, and R. Wrembel, "A logical model for multi-version data warehouse," in *Proceedings of the 16th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, vol. 8646, pp. 23-24, Springer LNCS, Munich, Germany, 2014.

[17] M. Blaschka, C. Sapia, and G. Höfling, "On schema evolution in multidimensional databases," in *Proceedings of the 1st International Conference on Data Warehousing and Knowledge (DaWaK'99)*, vol. 1676, pp. 153–164, Springer LNCS, Florence, Italy, 1999.

[18] A. Marotta, *Data warehouse design and maintenance through schema transformations [M.S. thesis]*, Universidad de la República Uruguay, 2000.

[19] C. Letz, E. T. Henn, and G. Vossen, "Consistency in data warehouse dimensions," in *Proceedings of the IEEE International Symposium on Database Engineering Applications (IDEAS'02)*, pp. 224–232, Washington, DC, USA, 2002.

[20] C. Kaas, T. B. Pedersen, and B. Rasmussen, "Schema evolution for stars and snowflakes," in *Proceedings of the International Conference on Enterprise Information Systems (ICEIS'04)*, pp. 425–433, Porto, Portugal, 2004.

[21] E.-J. Rechy-Ramírez and E. Benítez-Guerrero, "A model and language for bitemporal schema versioning in Data Warehouses," in *Proceedings of the 15th International Conference on Computing, CIC 2006*, pp. 309–314, November 2006.

[22] R. Wrembel, "A survey of managing the evolution of data warehouses," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 5, no. 2, pp. 24–56, 2009.

[23] R. Wrembel, "On handling the evolution of external data sources in a data warehouse architecture," in *Integrations of data warehousing, data mining and database technologies: innovative approaches*, D. Taniar and L. Chen, Eds., pp. 106–147, Information Science Reference, Hershey, PA, USA, 2011.

[24] P. Manousis, P. Vassiliadis, A. Zarras, and G. Papastefanatos, "Schema Evolution for databases and data warehouses," in *Proceedings of the 5th European Business Intelligence Summer School (eBISS, 2015)*, Barcelona, Spain, 2015.

[25] M. Golfarelli, J. Lechtenbörger, S. Rizzi, and G. Vossen, "Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation," *Data & Knowledge Engineering*, vol. 59, no. 2, pp. 435–459, 2006.

[26] M. Golfarelli, D. Maio, and S. Rizzi, "The dimensional fact model: A conceptual model for data warehouses," *International Journal of Cooperative Information Systems*, vol. 7, no. 2-3, pp. 215–247, 1998.

[27] J. Eder and K. Wiggisser, "Data Warehouse Maintenance, Evolution and Versioning," *Springer Encyclopedia of Database Systems*, pp. 664–669, 2009.

[28] J. Eder and K. Wiggisser, "Data warehouse maintenance, evolution and versioning," in *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction*, L. Bellatreche, Ed., pp. 171–188, IGI Global, 2010.

[29] R. Bliujute, S. Saltenis, G. Slivinskas, and C. S. Jensen, "Systematic change management in dimensional data warehousing," in *Proceedings of the 3rd International Baltic Workshop on Databases and Information Systems*, pp. 27–41, 1998.

[30] G. Garani, G. K. Adam, and D. Ventzas, "Temporal data warehouse logical modelling," *International Journal of Data Mining, Modelling and Management*, vol. 8, no. 2, pp. 144–159, 2016.

[31] W. Ahmed, E. Zimanyi, and R. Wrembel, "Temporal Data Warehouses: Logical Models and Querying," in *Proceedings of the 11th Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA)*, Bruxelles, Belgium, 2015.

[32] E. Malinowski and E. Zimanyi, "A conceptual solution for representing time in data warehouse dimensions," in *Proceedings of the 3rd Asia-Pacific Conference on Conceptual modelling (APCCM'06)*, vol. 53, pp. 45–54, Springer LNCS, Darlinghurst, Australia, 2006.

[33] J. Eder, C. Koncilia, and T. Morzy, "The COMET Metamodel for temporal data warehouses," in *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, vol. 2348, pp. 83–99, Springer LNCS, Toronto, Canada, 2002.

[34] S. Rizzi and M. Golfarelli, "X-time: Schema versioning and cross-version querying in data warehouses," in *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007*, pp. 1471-1472, April 2007.

[35] M. Golfarelli, V. Maniezzo, and S. Rizzi, "Materialization of fragmented views in multidimensional databases," *Data & Knowledge Engineering*, vol. 49, no. 3, pp. 325–351, 2004.

[36] J. Eder, C. Koncilia, and D. Mitsche, "Automatic detection of structural changes in data warehouses," in *Proceedings of the 5th International Conference on Data Warehousing and Knowledge (DaWaK'03)*, pp. 119–128, Pargue, Czech Republich, 2003.

[37] M. Body, M. Miquel, Y. Bédard, and A. Tchounikine, "Handling evolutions in multidimensional structures," in *Proceedings of the Nineteenth International Conference on Data Ingineering*, pp. 581–591, Banglore, India, March 2003.

[38] A. Vaisman, *Updates, view maintenance and time management in multidimensional databases [Ph.D. thesis]*, University of Buenos Aires, 2001.

[39] A. A. Vaisman, A. O. Mendelzon, W. Ruaro, and S. G. Cymerman, "Supporting dimension updates in an OLAP server," *Information Systems*, vol. 29, no. 2, pp. 165–185, 2004.

[40] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, "A foundation for capturing and querying complex multidimensional data," *Information Systems*, vol. 26, no. 5, pp. 383–423, 2001.

[41] A. O. Mendelzon and A. A. Vaisman, "Temporal queries in OLAP," in *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB 2000*, pp. 242–253, Cairo, Egypt, September 2000.

[42] P. Chamoni and S. Stock, "Temporal structures in data warehousing," in *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, pp. 353–358, Springer LNCS, Florence, Italy, 1999.

[43] C. A. Hurtado, A. O. Mendelzon, and A. A. Vaisman, "Maintaining data cubes under dimension updates," in *Proceedings of the 15th International Conference on Data Engineering, ICDE-99*, pp. 346–355, Sydney, Australia, March 1999.

[44] D. Solodovnikova, "Data warehouse evolution framework," in *Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems (SYRCoDIS'07)*, pp. 6–12, Moscow, Russia, 2007.

[45] J. Chmiel, "Data structures for multiversion data warehouse," in *Proceedings of the International Conference on Advances in Databases and Information Systems (ADBIS'09)*, vol. 5968, pp. 202–210, Springer LNCS, Riga, Latvia, 2010.

[46] H. Kang and C. Chung, "Exploiting versions for on-line data warehouse maintenance in MOLAP servers," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pp. 742–753, Hong Kong, China, 2002.

[47] D. Quass and J. Widom, "On-line warehouse view maintenance," *SIGMOD Record*, vol. 26, no. 2, pp. 393–404, 1997.

[48] R. Wrembel and B. Bębel, "Metadata management in a multiversion data warehouse," *Journal of Data Semantics*, vol. 8, pp. 118–157, 2007.

[49] Z. Bellahsene, "View adaptation in data warehousing systems," in *Proceedings of the 9th International Conference on Database and Expert Systems Applications (DEXA'98)*, vol. 1460, pp. 300–309, Springer LNCS, Vienna, Austria, 1998.

[50] R. M. Bruckner and A. M. Tjoa, "Capturing delays and valid times in data warehouses - Towards timely consistent analyses," *Journal of Intelligent Information Systems*, vol. 19, no. 2, pp. 169–190, 2002.

[51] R. J. B. Vanwersch, K. Shahzad, K. Vanhaecht et al., "Methodological support for business process redesign in health care: A literature review protocol," *International Journal of Care Coordination*, vol. 15, no. 4, pp. 119–126, 2011.

[52] S. Chen, B. Liu, and E. A. Rundensteiner, "Multiversion-based view maintenance over distributed data sources," *ACM Transactions on Database Systems (TODS)*, vol. 29, no. 4, pp. 675–709, 2004.

[53] D. Fasel and K. Shahzad, "A datawarehouse model for integrating fuzzy concepts in meta table structures," in *Proceedings of the 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS 2010*, pp. 100–109, Oxford, UK, March 2010.

[54] A. Gosain and K. Soraha, "Storage structure for handling schema versions in temporal data warehouses," in *Proceedings of the Progress in Intelligent Computing Techniques: Theory, Practice, and Applications, Springer Advances in intelligent Systems and Computing*, vol. 518, pp. 501–511, 2017.

[55] S. Rizzi, "What-if Analysis," in *Encyclopedia of Database Systems*, L. Liu and T. Özsu, Eds., Springer, 2009.

[56] P. Lane, *Oracle Database Data Warehousing Guide*, 10g Release 1, 2005.