Future Generation Computer Systems 72 (2017) 264–272

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A hybrid index for temporal big data

Mei Wang[a], Meng Xiao[a], Sancheng Peng[b,c,*], Guohua Liu[a]

[a] *School of Computer Science and Technology, Donghua University, Shanghai, 201620, PR China*
[b] *School of Informatics, Guangdong University of Foreign Studies, Guangzhou, Guangdong Province, 510420, PR China*
[c] *Laboratory of Language Engineering and Computing, Guangdong University of Foreign Studies, Guangzhou, Guangdong Province, 510420, PR China*

HIGHLIGHTS

- A novel segmentation hybrid index SHB+-Tree for temporal big data is proposed.
- The proposed index integrates the advantages of temporal index and object index.
- The segmented storage strategy is proposed.
- The bottom-up index construction approach is provided.
- The experiments are conducted to verify the effectiveness of the proposed method.

ARTICLE INFO

Article history:
Received 15 November 2015
Received in revised form
14 May 2016
Accepted 6 August 2016
Available online 26 August 2016

Keywords:
Big data
Temporal database
Temporal index
SHB+-Tree index
Segmented storage

ABSTRACT

Temporal index provides an important way to accelerate query performance in temporal big data. However, the current temporal index cannot support the variety of queries very well, and it is hard to take account of the efficiency of query execution as well as the index construction and maintenance. In this paper, we propose a novel segmentation-based hybrid index B+-Tree, called SHB+- tree, for temporal big data. First, the temporal data in temporal table deposited is separated to fragments according to the time order. In each segment, the hybrid index is constructed by integrating the temporal index and the object index, and the temporal big data is shared by them. The performance of construction and maintenance is improved by employing the segmented storage strategy and bottom-up index construction approaches for every part of the hybrid index. The experimental results on benchmark data set verify the effectiveness and efficiency of the proposed method.

1. Introduction

In the era where data are being produced over time and shared in an unprecedented pace, mining the information in the big data has become increasingly crucial. Temporal information is the natural and basic description for the development and changes of real-world objects, and almost everything has explicit or implicit temporal features. While the traditional snapshot databases always record the information in a given specific time, it is difficult to reflect the dynamic changes of real world sufficiently and accurately. It is becoming increasingly urgent for the management and retrieval of temporal big data in most modern database systems.

Temporal big data management has already attracted wide concerns in both academic and industrial fields. Tang [1] proposed the concept of bi-temporal data at an earlier time. In this work, each tuple of the temporal table carries two time intervals $[start_t, end_t]$ and $[start_v, end_v]$, representing transaction time and valid time (a.k.a system time and application time, respectively). He also proposed to take time interval as a key, which makes a breakthrough in traditional databases which only take digit or character as a key. In this basis, many temporal database prototypes have been implemented, such as TimeDB [2] and TempDB [3]. Under the impetus of the above research and real applications, ISO/IEC published the edition of the SQL standard in December 2011, SQL: 2011 [4,5], which includes an important functionality to create and manipulate temporal tables. In the meantime, many popular commercial databases such as Oracle [6], IBM DB2 [7], SAP HANA [8] also include temporal features. With the developments of temporal databases, some key technologies in the traditional databases have been re-examined. As an important way to accelerate query performance, index has received great attentions. Some index structures have been proposed to support

* Corresponding author.
E-mail address: psc346@aliyun.com (S. Peng).

http://dx.doi.org/10.1016/j.future.2016.08.002
0167-739X/© 2016 Elsevier B.V. All rights reserved.

different temporal query operations such as temporal join [9,10], temporal aggregation [11,12] and time travel [13].

The existing temporal index technologies are always the extensions of traditional B+-Tree or R Tree. The Time index [14] is one of the earliest temporal indexing methods, which maps temporal relation to a multi-dimensional space. So the popular index such as B-tree can be readily adopted to support temporal operations. However, such kind of methods suffer low efficiency when data size increases. As data size reached TB or PB level, the cost of index construction and maintenance is extremely expensive. In addition, almost every method does not support all types of temporal operations and these methods do not work efficiently on modern hardware and architectures. In the latest study, SAP HANA proposed a novel index structure called Timeline index [15]. The Timeline index can support different kinds of temporal operations as well as is easy to be constructed and maintained by adopting the simple sequential structure. However, when using the Timeline index to process a query, it needs to do a linear scan of the index table. When the data size becomes large, the cost of continual linear scan is very expensive.

More importantly, in the current temporal big data management, the proposed index structures are always only built on the temporal attributes, which means they focus on fast query search like "what happened in a given time period" (denoted as temporal query). However, for many practical applications, users also pay close attention to the objects that they are interested in, in this case the query is likely as "what the given object happened" (denoted as object query) or in the more complex case, the query is likely as "What the given object happened in the given time period" (denoted as complex query). For the last two kinds of queries, it is difficult to avoid the linear scanning of the whole records in the process of query execution for the current temporal index structures. Although by building a traditional secondary index can reply "what the given object happened" more efficiently, the cost of update for the secondary index is very huge as the objects change over time.

To deal with the above problems, in this paper, we propose a novel segmentation hybrid index for dealing temporal big data. In the proposed method, the temporal big data is divided into segments by chronological order firstly, and then in each segment the local hybrid structure with temporal index and object index is created where both indexes share the same local segment data. Furthermore, the bottom-up index construction approach is incorporated in the proposed method to improve the performances of construction and maintenance of the index structure. Our contributions are summarized as follows:

- A hybrid index structure is proposed for the temporal data table. The query methods base on the proposed index are also provided in the paper. The proposed index structure readily incorporates the advantages of both temporal index and object index. As a result, it could meet the diverse needs of users' query more effectively compared with the previous methods.
- By segmenting the temporal big data in advance, our index is built on each local data segment, which greatly reduces the time of index construction and maintenance. Such process could be further speed up by parallel execution architectures. Combined with the bottom-up construction approach, the performances of construction and maintenance of the index have been improved significantly.
- The experiments have been conducted on benchmark data sets verify the effectiveness and the efficiency of the proposed method. Specifically, the SHB+-Tree (Segmentation Hybrid B+-Tree) index performs excellent in complex query, the cost of query time for the SHB+-Tree is reduced to 10% of the Timeline index.

The remainder of this paper is organized as follows. Section 2 gives an overview of existing work on temporal data management and temporal index technologies. Section 3 presents the SHB+-Tree index. Section 4 describes SHB+-Tree and the algorithms on how to process different kinds of temporal operators using the SHB+-Tree in detail. Section 5 provides the experimental results. Finally, we conclude this paper and suggest our future work in Section 6.

2. Related work

In this section, we investigate related work in three dimensions. The first dimension is the traditional database and index; the second is related to the temporal indexing technology; and the last is related to the temporal operations.

2.1. Traditional database and index

Traditional relational database is attribute–tuple two-dimensional structure [16], it is suitable for processing the permanent stability data. However, it is important to know that the traditional databases only save a snapshot instead of the complete history. The snapshot cannot reflect the historical changes of the objects, and it is hard to meet the real-time requirements for the industrial applications. Temporal database is a DBMS which supports time dimension. The Temporal DB can describe not only the information at some point, but also the history and future of the data. An important direction of work is how to model and organize temporal data. Index technology is one of the key technologies to improve the query efficiency of mass data, and the research of index has attracted great attentions in recent years.

Traditional databases have rules to create effective indexes. First, create an index on the column that is joined in queries frequently, and not a foreign key. Second, create an index on the column which is used to sort or group frequently. Third, create an index on the column that is used in the conditional expression frequently and has more different values. For this reason, the traditional index can support the query such as what the object do very well. However, as time goes on, there are constant updates on the temporal database, which makes the cost of update and maintenance of the index very expensive. Therefore, a large body of researches on indexes created on the temporal attribute has established.

2.2. Temporal indexing technology

In order to accelerate query performance in temporal database, various algorithms and data structures have been proposed for different temporal data models. Since most of these index structures were developed in the mid-to-late '90s, they are designed for hard-disk efficiency, optimizing the number of I/O operations for updates and queries. The Time index [14] is one of the earliest temporal indexing methods. Technically, the Time index is a B+-Tree over versions, and provides explicit support for multiple temporal operations. The multi-version B-tree [17] is one of the most advanced temporal indexing methods. It provides an index for both key- and time-dimensions with optimal I/O behavior. Temporal XML has also attracted more and more attention in recent researchers. The temporal XML indexing method TXSIM [18] based on suffix tree is proposed. The Timeline index [15] was proposed by SAP HANA [5] for processing different kinds of temporal queries on temporal data. SAP HANA is a commercial database system which employs both a column store and a row store for in-memory data processing. Basic support for temporal data is already available natively in HANA. The bitemporal Timeline index [19] is also proposed for processing