SBT-QL: A Stock Time Series Query Language based on Specialized Binary Tree Representation

Tak-chung Fu*, Fu-lai Chung, Robert Luk, Man-yee Au and Chak-man Ng

Abstract

—Stock time series query is one of the fundamental components in technical analysis. Technical analysts would like to query either a whole time series or a segment of a time series in different resolutions for long-term or short-term investigations. Moreover, unnecessary data points should be filtered during the query process for easy analysis. In this paper, query approaches for both the whole stock time series and a subsequence of the stock time series based on the SB-Tree representation scheme are proposed. An approximate approach is proposed to improve the performance of the subsequence query process. Moreover, the local and global pruning methods are suggested to filter the unimportant data points. Besides studying these various query methods, an user-oriented query language, called SBT-QL, is proposed for: using easily, implementing efficiently and providing the ability to specify the query conditions to limit the retrieved data. The efficiency and effectiveness of the proposed approach are shown by the experiments.

1. INTRODUCTION

STOCK time series query is one of the fundamental components in technical analysis. Unlike transactional databases with discrete items, the natures of time series data include: large in data size, high dimensionality and updating frequently. Indeed, stock time series has its own characteristics over other time series data. For example, a stock time series is typically characterized by a few critical points and multi-resolution consideration is always necessary for long-term and short-term analyses. In addition, technical analysis is usually used to identify patterns of market behavior, which have high probability to repeat themselves. These patterns are similar in the overall shape but with different amplitudes and/or durations.

Recent research on time series query can be found in [1,2]. First, Shape Definition Query (SDL) is introduced in [1] for retrieving objects based on the shapes contained in the histories associated with these objects. In [2], it proposes a Chart-Pattern Language (CPL) to enable financial analysts to define patterns with subjective criteria and incrementally compose complex patterns from simpler patterns. However, both of them are query the time series data by defining a shape of the pattern but leak of the ability to retrieve a time series or its subsequence in different resolutions.

In this paper, time series retrieval methods come along with a query language, which is based on a tree representation scheme - a specialized binary tree (i.e. SB-Tree) is proposed. The representation scheme is first proposed in [3] based on a SB-Tree structure and it is customized for stock time series applications. Based on the SB-Tree representation scheme, multi-resolution data retrieval and subsequence retrieval can be achieved. Moreover, a query language has been built to simplify the retrieval task for the users. The paper is organized into five sections. The SB-Tree representation scheme and the corresponding retrieval process are presented in section 2. Section 3 proposes a query language which based on the proposed retrieval mechanisms of the SB-Tree. The simulation results are reported in section 4 and the conclusion is made in the final section.

* Corresponding Author: Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: cstcfu@comp.polyu.edu.hk).

2. A TIME SERIES REPRESENT ATION FOR DATA RETRIEVAL

In this section, the financial time series representation scheme, which is adopted in this paper, will be revisited briefly first and the retrieval processes are proposed afterwards.

A. Specialized Binary Tree Data Structure

In view of the importance of extreme points in stock time series, the identification of perceptually important points (PIP) is firstly introduced in [4]. The frequently used stock patterns are typically characterized by a few critical points. These points are perceptually important in the human identification process and should be considered as more important points. The proposed scheme follows this idea by reordering the sequence *P* based on PIP identification, where the data point identified in an earlier stage is considered as being more important than those points identified afterwards. The distance measurement for evaluating the importance is the vertical distance (VD) [4].

After introducing the concept of data point importance, a binary tree (B-tree) structure has been proposed to store the time series data and is called specialized binary tree (SB-Tree) [3]. To create a SB-Tree, the PIP identification process [4] is adopted. A sample time series and the corresponding SB-Tree built are shown in Fig.1. Detail creating and accessing processes of the SB-Tree can be found in [3].

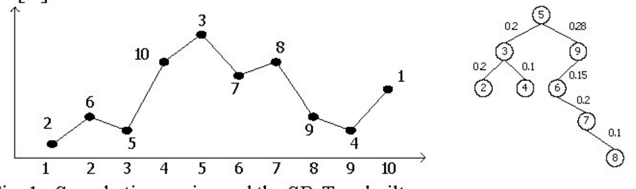

Fig. 1. Sample time series and the SB-Tree built

B. Retrieving the Whole Time Series from the SB-Tree

Supposing a SB-Tree is given, data then can be retrieved according to the data point importance [3]. Besides, retrieving all the data points, it may not necessary to retrieve all the data points like the limitation of the visualization devices, or a specified number of data points are required for pattern matching or analysis. It is related to the multi-resolution issue and the number of data points required, that is the resolution *n,* is supposed to be given. Then, the SB-Tree is accessed recursively, started from the root and the time series data points will be retrieved according to their importance.

Additionally, the retrieval method should be able to prune the unnecessary nodes of the SB-Tree. Unimportant signals (i.e. data points) of a time series should be filtered according to a threshold λ. As the tree is accessed from the top and the VD of each node is considered, when the VD of a node is smaller then λ, the fluctuation does not vary widely and the descendants are considered as less important to the users. Thus, this node and all the descendants of it can be filtered. It is called local pruning method. Fig.2 shows the local pruning of a sample SB-Tree which the threshold is set to 0.15 (i.e. λ <0.15).

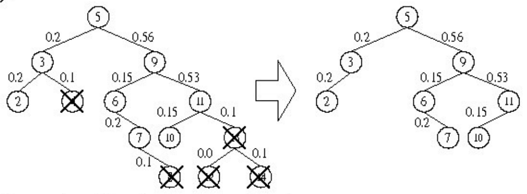

Fig. 2. Example of local pruning method

Another approach is to prune the retrieved data by a global error ratio α. This pruning method is based on the ratio of fluctuation. By determining the maximum (*max*) and minimum (*min*) prices within the retrieved range of the time series, the fluctuation ratio *fr* is calculated by: *fr*=(*price-min*)/(*max-min*) where *price* is the amplitude of the evaluating point. If *fr* is less than α, this point and all the descendants will be pruned. This approach can help the users to filter the unimportant points which are less critical to the overall shape of the time series and is called global pruning method.

C. Retrieving a Time Series Subsequence from the SB-Tree

When a sub-tree is considered, it represents a subsequence of the time series. To retrieve a subsequence, either an exact or an approximate approach can be applied based on the SB-Tree representation. For the exact approach, because the overall shape of a subsequence is different from its whole sequence, the data points of the subsequence should be retrieved in an exact order based on their importance according to this subsequence. In other words, the SB-Tree of this subsequence is needed to build.

However, it is time consuming to build the SB-Tree of the subsequence every time when retrieving any subsequence. An approximate approach is proposed. The subsequence is retrieved according to the data point importance in the whole SB-Tree of the corresponding time series. It is generally faster than the exact approach as no re-calculation is necessary. It is achieved only based on accessing the SB-Tree with additional criteria. Given the starting and ending points of a subsequence, the subsequence can be retrieved from the SB-Tree of the time series it belongs to. The data points within that subsequence can be identified based on their positions in the SB-Tree. Therefore, a similar retrieval mechanism as accessing the whole SB-Tree is introduced with additional condition checking. Furthermore, retrieving the subsequence in different resolutions can also be achieved by retrieving a smaller number of data points from the tree comparing to the length of the subsequence, that is resolution *n*.

For example, if the subsequence from point 5 to point 11 of a sample time series in Fig.3a is needed to retrieve, the corresponding nodes needed to retrieve are shown in Fig.3b and the order of retrieving the data points is: 5→9→11→10→6→7→8.

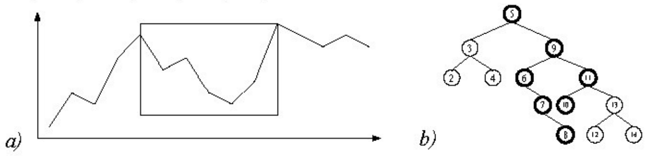

Fig. 3. Example of retrieving a time series subsequence from the SB-Tree

3. QUERY LANGUAGE FOR THE SB-TREE

Query language is a high-level computer language which is primarily oriented towards the retrieval of data held on files or databases [5]. Indeed, a query language for stock time series storage should be used easily, implemented efficiently and is able to specify the retrieval conditions to limit the retrieved data. Based on the retrieval methods proposed in Section II, a SQL-like query language is developed to help the users to retrieve necessary time series data based on the SB-Tree representation, specify the query conditions and reduced the data size. It is called Specialized Binary Tree Query Language (SBT-QL). In this stage, the functions of SBT-QL include whole sequence / subsequence retrieval, multi-resolution retrieval (i.e. all points / *n* number of important points) and the two proposed pruning methods. Fig.4 shows the complete syntax of the SBT-QL while Table 1 describes the corresponding meanings of the syntax. Query examples will be presented in Section IV.

```
[ Get Chart / Prices ]
[ Storage RDBMS / XML ]
Stock stock_id
[ from start_trade_date to end_trade_date [ Exact
/ Approximate ] ]
[ Resolution no_of_pip ]
[ [ Local / Global ] Pruning threshold ]
```

Remarks: Case-insensitive; Each line can be issued in arbitrary order; Syntax within brackets [ ] are optional; Bold words are index words; Underlined words are exact values to be input; Italic words are the value to be input.

Fig. 4. Syntax of the proposed query language

Table. 1. Definition of the syntax

Syntax	Meaning
`Get` <u>`Chart`</u> `/` <u>`Prices`</u>	The retrieved data can be displayed by a chart or listed in text. Default is set to `Prices`.
`Storage` <u>`RDBMS`</u> `/` <u>`XML`</u>	Specify the data source, either from RDBMS or XML file. In this paper, only RDBMS will be considered.
`Stock` *`stock_id`*	Stock to be queried, where *`stock_id`* is the identifier of the stock.
`from` *`start_trade_date`* `to` *`end_trade_date`*	The range of trade dates of the query data. Whole time series retrieval is assumed if the range is not specified.
`Exact / Approximate`	Specify either exact or approximate retrieval approach to be applied in the subsequence query. Default value is set to approximate approach for subsequence query.
`Resolution` *`no_of_pip`*	Resolution, number of PIP requested. If it is not specified, all the data points within the range will be returned. Otherwise, *`no_of_pip`* number of important points will be returned where *`no_of_pip`* should be smaller than or equal to the length of the queried time sequence.
`[ Local / Global ] Pruning` *`threshold`*	Specify the pruning approaches to be used. Either using local pruning method, (*`threshold`*= λ) or global pruning method (*`threshold`*= α).

4. EXPERIMENTAL RESULTS

The simulation results are reported in this section. The proposed approach for time series representation was implemented in Java 2 Platform, Standard Edition (J2SE). The experiments were conducted on Intel Pentium(R) M Processor 1.4Ghz with 512M RAM. Ten stock time series with average 2000 data points were imported to the time series database for the simulation. Time series with 2532 data points captured from the past ten years of the Hong Kong Hang Seng Index (HSI) was used to demonstrate the visualization results.

A. Accuracy vs. Approximate Subsequence Query Approaches

In this subsection, the performance and accuracy of retrieving a subsequence using the exact and the approximate approaches are evaluated. The effect of retrieving different numbers of data point (resolution) is first evaluated. The lengths of subsequence used were fixed to 500 and 2500, which is about 10% and 90% of the whole time series data respectively. The query used was:

```
Get Prices Storage RDBMS Stock hsi from s to e
approach Resolution n;
```

where `from` *`s`* `to` *`e`* = `from 1993-02-08 to 1995-02-10` and `from 1989-01-03 to 1999-01-03`, *`approach`* = `Exact` and `Approximate`, *`n`* = `9, 50, 100, 200, 300` and `400`. The Chart option in Get directive was used for displaying the visualization result.

Fig.5 shows that the performance of the approximate approach generally outperformed the exact approach. The charts of accuracy shows that the accuracy was in insignificant differences of both approaches – it was less then 0.007 for retrieving 9 data points and less then 0.001 for retrieving a larger number of points. Fig.6 shows the visualization effect of the query results when the difference was 0.0061 and the accuracy was around 0.94. When only some of the data points are retrieved, distance is existed between the overall shapes of the retrieved series when comparing to the original time series, no matter the exact or the approximate query is adopted.

In addition, when the length of the subsequence was set to 500 points, less than 0.4 second was required for the approximate queries, while 0.5 to 0.6 second was needed for the exact queries. When the length of the subsequence was set to 2500 points, the approximate approach required less than half of the time compared to the exact approach. Thus, it can be concluded that the approximate approach is more appropriate than the exact approach.

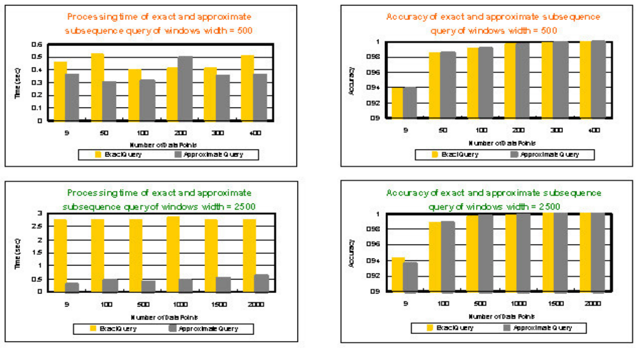

Fig. 5. Processing time and accuracy of exact and approximate subsequence query of windows width = 500 and 2500

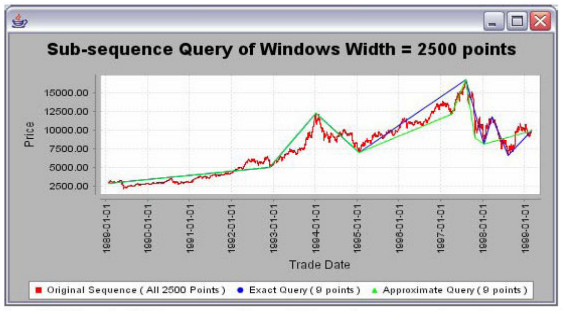

Fig. 6. Different PIPs are retrieved based on exact and approximate subsequence query approaches (width of subsequence query is 2500 and number of PIP retrieved is 9)

The second experiment evaluated the query performance of retrieving fixed number of points at different lengths of subsequence. Particularly, the retrieval of 9 PIPs is interested because it can match the widely recognized technical patterns like double tops and head-and-shoulder. The query used was:

```
Get Prices Storage RDBMS Stock hsi from s to e
approach Resolution n;
```

where *`s`* and *`e`* specify the range of subsequence and obtains the length of subsequence which are `10, 500, 1000, 1500, 2000` and `2500`, *`approach`* = `Exact` and `Approximate`, *`n`* = `9` and `100`.

The accuracy of all the experiments in this subsection was between 0.93 and 0.99 which is similar to the previous experiment. Fig.7 shows the processing time of the exact and the approximate subsequence query approaches of retrieving different numbers of PIP (i.e. 9 and 100). From the result, it shows that when the length of the query subsequence increased, the performance of the approximate approach kept stable while that of the exact approach increased accordingly. For the approximate query, it took around 0.5 second while the exact query increased from less than 1.2 second to more than 2.5 seconds when the length of the subsequence rose from 150 to 2500. Again, it is found that the approximate subsequence query approach outperformed the exact subsequence query approach.

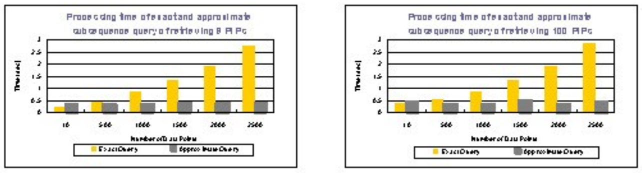

Fig. 7. Processing time of exact and approximate subsequence query of retrieving different numbers of PIP, no. of PIP = 9 (left) and no. of PIP = 100 (right)

B. *Local vs. Global Pruning Methods*

In this subsection, the performance and corresponding visualization effects by using the two proposed pruning methods: local pruning and global pruning are shown. First, the processing time, the accuracy and the number of points retrieved by applying the local pruning method with different thresholds λ are shown in Table 2. The query used was:

```
Get Prices Storage RDBMS Stock hsi Local Pruning λ;
```

where λ =`1, 100, 1000` and `2000`.

From the result, it can be observed that the time required for different thresholds remained steady around 0.5 second. Larger threshold resulted in less number of points to be retrieved and led to lower accuracy. The visualization effects on different threshold levels are shown in Fig.8. By setting a larger threshold, more data points were filtered.

Table. 2. Processing time, accuracy and number of PIP retrieved by using local pruning method

Threshold λ (*price*)	Process time (second)	Accuracy	Number of point retrieved (from 2532 points)
1	0.551	0.9999	2481
100	0.541	0.9999	802
1000	0.550	0.9820	53
2000	0.551	0.9647	19

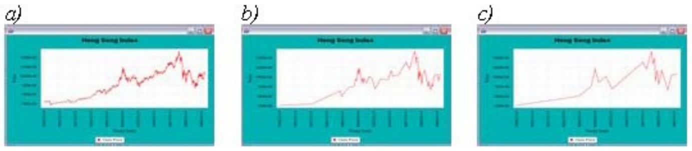

Fig. 8. Visualization of the pruning effect using local pruning approach with different thresholds, (a) λ=100, (b) λ=1000 and (c) λ=2000

Second, the processing time, the accuracy and the number of points retrieved by applying the global pruning method with different thresholds α are shown in Table 3. The query used was:

```
Get Prices Storage RDBMS Stock hsi Global Pruning
α;
```

where α = `0.0001, 0.005, 0.07` and `0.15`.

With the global pruning method, similar effect of using the local pruning method was obtained. However, the pruning process was controlled by fluctuation ratio of the whole query sequence instead of the VD of the data point. Fig.9 shows the visualization effects of using different fluctuation ratios *fr*.

Table. 3. Processing time, accuracy and number of PIP retrieved by using global pruning method

Threshold α (fluctuation ratio *fr*)	Process time (second)	Accuracy	Number of point retrieved (from 2532 points)
0.0001	0.511	0.9999	2461
0.005	0.530	0.9989	1012
0.07	0.531	0.9810	52
0.15	0.540	0.9647	19

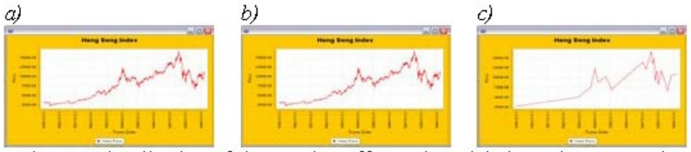

Fig. 9. Visualization of the pruning effect using global pruning approach with different thresholds (a) α=0.005, (b) α=0.07 and (c) α=0.15

5. CONCLUSION

In this paper, the retrieval methods for both the whole time series data and a subsequence of the time series based on the SB-Tree representation scheme are proposed. An approximate approach is proposed to improve the performance of the subsequence query process. Moreover, the local and the global pruning methods are suggested to filter the unimportant data points. Besides studying these various retrieval methods, a query language, called SBT-QL, is proposed for accessing the SB-Tree. SBT-QL is an SQL-like language which can be used easily, implemented efficiently and is able to specify the retrieval conditions to limit the retrieved data. After testing and evaluating the proposed query language, it is found that it functions efficiently and effectively.

REFERENCES

[1] R. Agrawal, G. Psaila, E.L. Wimmers, M. Zait: Querying Shapes of Histories. Proc. of the 21st VLDB (1995) 502-514
[2] Saswat, S.C. Khoo, W.N. Chin: Charting Patterns on Price History. Proc. of ACM SIGPLAN International Conf. in Functional Programming (2001)
[3] T.C. Fu, F.L. Chung, R. Luk, C.M. Ng: A Specialized Binary Tree for Financial Time Series Representation," In: The 10th ACM SIGKDD Workshop on Temporal Data Mining (2004) 96-103
[4] F.L. Chung, T.C. Fu, R. Luk, V. Ng: Flexible Time Series Pattern Matching Based on Perceptually Important Points. In: The 17th IJCAI Workshop on Learning from Temporal and Spatial Data (2001) 1-7
[5] Query languages: a unified approach, British Computer Society. Query Language Group, Heyden on behalf of the British Computer Society (1981)