HSTB-index: A Hierarchical Spatio-Temporal Bitmap Indexing Technique

Cesar Joaquim Neto[1, 2], Ricardo Rodrigues Ciferri[1], Marilde Terezinha Prado Santos[1]

[1] Department of Computer Science – Federal University of São Carlos (UFSCar)
Rodovia Washington Luís, km 235 - SP-310 – 13565-905 – São Carlos – SP – Brazil

{cjneto, ricardo, marilde}@dc.ufscar.br

[2]International Business Machines (IBM)
Rodovia Jornalista Francisco Aguirre Proença, km 09 (SP-101) Chácaras Assay – 13186-900 – Hortolândia – SP – Brazil

cesarjn@br.ibm.com

Nível: Mestrado

Ingresso no Programa: Março de 2013

Exame de Qualificação: Março de 2014

Data Esperada de Conclusão: Março de 2015

Abstract. *Through years, the database group at the Federal University of São Carlos has been researching new ways of optimizing queries in data warehouses containing non-conventional data. The need for such research comes from a number of partners dealing with spatial and temporal data where a simple query can demand huge levels of computational efforts. Advances were carried out by creating new index structures that deal with join bitmap indexes, minimum boundary rectangles and temporal data (in separate or meshed together) and some variant proposals. This work intends to explore and extend the STB-index (Spatio-Temporal Bitmap Index) by creating a new hierarchical index of the temporal data called HSTB-index (Hierarchical Spatio-Temporal Bitmap Index). The attempt is to better prune the number of false candidates in temporal predicates that might arise before the final result can be calculated and fetched.*

Keywords: data warehouse**,** spatio-temporal databases, spatio-temporal data warehouses, bitmap index

1. Introduction

A data warehouse (DW) is a specialized multidimensional database adjusted for querying data among large integrated, subject-oriented, and historical datasets. Its structure, usually a star or a snowflake schema, allows data to be seen in distinct points of view as well as in different levels of granularity. In such model, the fact and dimension tables (through the defined dimension hierarchies) allow users to perform many OLAP (on-line analytical processing) operations such as 'slice and dice', pivot, drill-across, drill-down, and drill-up.

For years, the concept that dimension data could not be changed was defended by researchers until they realized such data was not as immutable as they thought. The concept of slow changing dimensions has been carved by Kimbal (2002) and soon researchers started to think changes could also be needed to the fact table data. In this line of thought, the temporal data warehouses concepts started to be shaped due to the need of capturing the changes a data warehouse can suffer through time being those related to data instantiation or their own structure. According to Jensen et al. (1994), there are two main mechanisms a temporal data warehouse can use to capture data instantiation changes: the validity time (also called "real world time" – the time a given information is valid for the business) and transaction time (also called "database system time" – the time a given information is valid for the database). Regarding structural changes, Roddick (1995) classifies them as schema modification, schema evolution, or schema versioning. Querying such databases has also become a challenge as structure additions that support evolving data and schema needed to be considered. Queries could fail by fetching invalid data or spanning different schema versions without proper processing. Solutions for such problem came in the form of additions to the SQL query syntax or pre-processing of queries (using the captured schema metadata) so schema versions and data evolution could be respected.

Spatial data warehouses (SDWs) had their own line of research with the inclusion of at least one spatial data type (like points, lines, circles and polygons) in one or more dimension tables or as a spatial measure in the fact table. Queries over SDWs have a spatial predicate, such as, "retrieve the profit per month per store for stores that intersect a given query window". The above mentioned inclusion brought immediate difficulties as the large volumes of data found in DWs together with the inherent complexity of dealing with spatial data made processing times to grow a lot and made clear the need for new index and query processing techniques. New indexing techniques and arrangements like the SB-index and its evolution, the HSB-index, were developed so they could be made available and used in future works (Siqueira et al. 2012).

As time evolved, researches found that for some applications there was a need for capturing changes to spatial data as well. As temporal data warehouses already had what was needed to comply with keeping data and structural history, it seemed natural to combine both temporal and spatial data warehouse concepts thus arising a new data warehouse type called spatio-temporal data warehouse (STDW). With such new data warehouse type, difficulties while querying data were also combined and a new line of research was found trying to speedup queries using both concepts and this is where this work fits in.

2. Related Work

Bitmap join indexes (BJI) are an evolution of the common bitmap indexes and they have found their application in data warehouses due to their ordered organization and ease to process. It can map rows from a fact table directly to their relative dimension rows without the need of the inner join operation. Queries using BJIs can be processed using bit operations which are resolved by processors in a single clock cycle.

Although, due to DWs' own property of having huge volumes of data, even bitmap indexes have problems to scale and such difficulty could be addressed by three different techniques to be presented below: Binning, compression and codification.

The binning technique proposed by Wu, Stockinger & Shoshani (2008) consists of grouping various keys in common bitmap structures called "bins" and, therefore, demanding less bitmaps to code a certain column. As example we can have a numeric column with values ranging from 0 to 100. Instead of having 100 bit vectors representing its joint to the fact table, we can create 10 vectors (or bins) containing the mapping bits for values in the intervals (0, 10], (10, 20], and so on. This arrangement makes it possible to exclude many bins in advance thus saving processing time. The drawback is that false candidates can be introduced requiring further refinement of the results but the gains obtained by excluding bins overpass the refinement step loss.

Wu, Otoo & Arie (2006) details the WAH (Word Aligned Hybrid code) technique which is a specialized compression algorithm. Due to the own DW nature of having huge number of rows to be mapped between fact and dimension tables, the resultant bit vectors are often sparse meaning many sequences of consecutive '0's and '1's can be found. The WAH algorithm makes use of such property by separating the bit vector in equal pieces (31 or 63 bit in length – according to the processor's architecture) and making two types of "runs" called 'tail' and 'filling'. Tails have "0" as starting bit indicating all remaining bits need to be respected as they are. Fillings have "1" as starting bit, with the second bit indicating the bit to be repeated, and the remaining bits indicating the number of piece repetitions being compressed. In an example found in Stockinger & Wu (2007), a bit vector containing 5456 bits (and separated in 31 bit pieces) could be represented using 96 bits (one tail piece followed by one filling piece indicating 174*31 "0" bits followed by another tail piece).

The coding technique presented by Wu & Buchmann (1998) has as central point the use of binary codes representing all the domain values found in the attribute to be indexed. Such coding is then used as base for the mapping vector resulting in a reduction on the number of vectors needed for the coding itself. As example we can have a column with 5 character values ("a" to "e") which would require 5 bit vectors to be mapped against the fact table. The same values could be mapped using 3 bits (000 for "a", 001 for "b", and so on) and the same mapping could be done using 3 codified bitmap vectors instead of 5. Extrapolating the idea we could have a 1 million domain values attribute coded by 20 vectors (instead of the original 1 million). The drawback is that the result needs to be de-codified before being returned but such cost would be acceptable taking in account the savings in number of mapping vectors and the calculations that can be done using them.

With all these bitmap functionalities at hand (gathered by their authors in a software called Fastbit) and aiming to improve query performance in SDWs, Siqueira (2009) introduced a new index structure called Spatial Bitmap Index (SB-index). The index is represented by a vector where its index match the keys for a specific spatial column of a dimension table. The vector contents are the minimum boundary rectangles (MBRs) of the spatial column value and a pointer linking to the corresponding BJI. The idea is to have a first step scanning through the MBRs contained in the index structure and filtering out the rows outside the query window. As MBRs are used for this first scan, results have to be further evaluated as false candidates might have been introduced. This index' aim is to trade from many "costly" polygon intersection operations to many "cheaper" square intersection operations plus a few polygon intersection operations (during false candidates filtering).

Although gains were soundly verified by the use of SB-index, advances on it were thought by the same author and the results were outlined in Siqueira et al. (2012). His idea was to improve the index scan by exchanging the vector based structure to a hierarchical structure (for instance, an R*-tree). Such tree-like organization made possible to further group MBRs into broader groups and prune tree branches not pertaining to a particular MBR group. A main memory buffering technique was also used so to have data loaded for processing as long as possible. Such approach was called Hierarchical Spatial Bitmap index (or HSB-index).

As further development of the SB-index, Tsuruda (2013) has broadened its concept to contemplate STDWs. The outcome of her work was a new index called Spatio-Temporal Bitmap index (STB-index) which idea is the inclusion of initial and final validity times to the index' MBRs. Such temporal information allow temporal clauses to be evaluated first and, therefore, based on the time related to MBRs, the STB-index can discard invalid MBRs in advance.

3. Our Proposal

3.1. Description

In the same way the SB-index has been evolved into a hierarchized structure (the HSB-index), this work's proposal is to evolve the STB-index into a hierarchized structure for the valid times and, thus, decrease the number of candidates to be explored by pruning candidates in early steps. It seemed natural the new index to be called Hierarchical Spatio-Temporal Bitmap index, or HSTB-index. Table 1 below summarizes the four indexes and their main differences.

Table 1. Index features comparison

Index	Collection organization	Run method	Type of DW
SB-index	Vector	Analyze all MBRs	SDW
HSB-index	Tree	Prune MBR records	SDW
STB-index	Vector	Analyze all MBRs	STDW
HSTB-index	Tree / Other	Prune MBR records	STDW

Although the main idea is already set, its execution is generating majority of discussions. The first big question is regarding the most recommended structure to hold the MBRs. So far a B+-tree had been considered but it was also thought that a bitmap index using the binning technique might lead to good results too. Another pending question regards the granularity of the temporal data for the index organization where the "YYYYMM" format seems to be the most natural so far. A third point to be considered is which spatial index to use so the best results can be found (SB-index or HSB-index) regarding the selectivity of the queries. The last, yet very important, question is regarding the execution order of the index meaning that the decision whether to limit candidates using the temporal attributes earlier than the spatial attributes or vice-versa is still to be investigated. Such questions and discussions have led to some variants described below and which should be evaluated throughout the work.

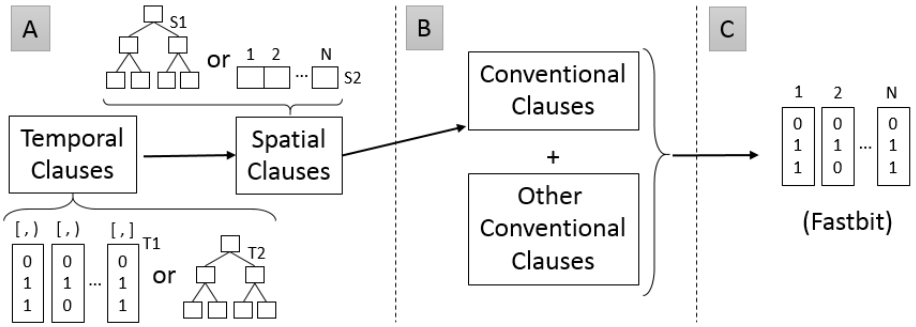

Figure 1. First strategy evaluating temporal clauses before spatial clauses

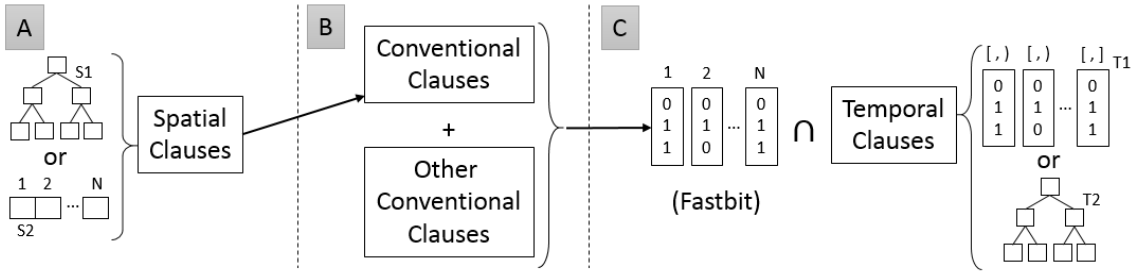

Figure 2. Second strategy evaluating spatial clauses before temporal clauses

Figure 1 depicts how the three main query execution phases can be executed. Phase A is the index processing until we can have the results in the form of conventional clauses which will be used in phase B so to incorporate all other conventional clauses for execution in Fastbit at phase C. It is important to mention the processing options that can be used in phase A were named T1 (temporal bitmap vectors using binning technique), T2 (B+-tree option for organizing records), S1 for HSB-index, and S2 for SB-index.

Figure 2 above outlines the second approach where the roles are inverted while running spatial and temporal query predicates. Phase A would have a first index processing by running option S1 (HSB-index) or S2 (SB-index) and the formed results would be incorporated to the other conventional clauses in phase B. Phase C would then come to limit data according to temporal clauses intersected to Fastbit results. It is worth to mention that we still have to investigate T1 (temporal bitmap vectors using binning

technique) and T2 (B+-tree organization for temporal data). Furthermore, T1 option has a promise of good performance as it can be intersected directly with bit-to-bit operation in Fastbit.

Considering the picture, we can name temporal predicate types as T, spatial predicate types as S, S1 being HSB-index, S2 being SB-index, T1 being the temporal bitmap using binning technique, and T2 being the B+-tree organization for temporal data. If we combine those abbreviated names, we have the eight combinations that should be evaluated in performance tests: T1_S1 (Temporal predicate is processed first using bitmapped temporal organization, Spatial predicate is processed second using HSB-index), T1_S2, T2_S1, T2_S2, S1_T1, S1_T2, S2_T1, and S2_T2.

Following the other indexes, the implementation of each alternative should use as development platform the C++ language under a Linux machine, PostgreSQL as database, and Fasbit as bitmap indexing solution.

3.2. Validation

Tests of the implemented alternatives should be performed so to exercise data with different query window sizes as well as different temporal spans. The GQM methodology should be used to define the needed performance tests and all the data that needs to be gathered while performing them. The plan is to use data generated by data warehouse benchmarks, such as SSB (Star Schema Benchmark) (O'NEIL, P. E. et al., 2009), Spatial SSB (Spatial Star Schema Benchmark) (Nascimento et al., 2011), and Spadawan Benchmark (Siqueira et al., 2010) and adapt the data generated to include spatio-temporal characteristics. The gathered results would then be evaluated so conclusions on which index strategy works best for each situation simulated by the tests can be taken.

4. Performed Activities

4.1. Current activities

So far, we can list as current activities: a) literature review so to get involved in the activities already performed and up to the challenge for the new activities to be performed; and b) proposal discussion taking in account the alternatives that can be taken.

4.2. Future activities

Future activities comprise: a) reproduce the workbench to run the three other indexes (SB-index and HSB-index to be used as steps in the new index, STB-index to be used as comparison); b) implement the options proposed in the solution; c) plan and execute tests with results collection; and d) compare results and conclude on the best alternatives fitting each situation.

5. Final Considerations

Nowadays, indexing DWs became a major concern with all the promised data volumes and even more in the face of the new non-conventional data demands. STDWs are important on their side as they can map fast changing phenomena as well as many geographical happenings worth studying after data collection. The inclusion of the

mentioned indexes in a working database environment (like Postgresql) can be considered as future work as well as moving the whole spatio-temporal with indexing idea to a No-SQL database where data growth restrictions are lower.

References

Jensen, C., Clifford, J., Elmasri, R., Gadia, S. K., Hayes, P. J., & Jajodia, S. (1994). A Consensus Glossary of Temporal Database Concepts. ACM SIGMOD Record, 23(1), 52-64.

Kimball, R.; Ross, M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2nd ed. New York, NY: Wiley, 2002. 464 p.

O'Neil, P. E.; O'Neil, E. J.; Chen, X. Star Schema Benchmark. Disponível em: <http://www.cs.umb.edu/~poneil/StarSchemaB.PDF>. Acesso em: Jul. 2013

Nascimento, S. M.; Tsuruda, R. M.; Siqueira, T. L. L.; Times, V. C. T.; Ciferri, R. R.; Ciferri, C. D. A. The Spatial Star Schema Benchmark. In: Brazilian Symposium ON Geoinformatics (GEOINFO), 12nd, 2011, Campos do Jordão, SP. Proceedings... São José dos Campos, SP: MCT/INPE, 2011. p.73-84.

Roddick, J. (1995). A Survey of Schema Versioning Issues for Database Systems. Information and Software Technology, 37(7), 383-393.

Siqueira, T. L. L. SB-index: um índice espacial baseado em bitmap para data warehouse geográfico. 2009. 118 f. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, Universidade Federal de São Carlos, São Carlos. 2009

Siqueira, T. L. L.; Ciferri, R. R.; Times, V. C.; Ciferri, C. D. A. Benchmarking spatial data warehouses. In: International Conference On Data Wareahousing and Knowledge Discovery (DaWaK), 12th, 2010, Bilbao, Spain. Proceedings... Berlin / Heidelberg: Springer, 2010. p. 40-51.

Siqueira, T. L. L.; Ciferri, C. D. A.; Times, V. C.; Ciferri, R. R. The SB-index and the HSB-index: efficient indices for spatial data warehouses. Geoinformatica, Hingham, MA, v. 16, n. 1, p. 165-205, jan. 2012.

Tsuruda, R. M. STB-index: Um Índice Baseado em Bitmap para Data Warehouse Espaço-Temporal. 2013. 81 f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação, Universidade Federal de São Carlos, São Carlos. 2013

Wu, K.; Stockinger, K.; Shoshani, A. Breaking the Curse of Cardinality on Bitmap Indexes. International Conference on Scientific and Statistical Database Management, 20., 2008, Hong Kong. **Proceedings…** Berlin/Heidelberg: Springer-Verlag, 2008. p.348-365

Wu, K.; Otoo, E. J.; Arie, S. Optimizing Bitmap Indices With Efficient Compression. ACM Transactions on Database Systems, v.31, n.1, p.1-38, 2006

Wu, M.-C.; Buchmann, A. P. Research Issues in Data Warehousing. In: International Conference on Data Engineering, 14., 1998, Orlando. **Proceedings…** Washington: IEEE Computer Society, 1998. p.220-230