Proc. First International Conference on Data Warehousing and Knowledge Discovery (DaWak'99), Aug. 30 – Sep. 1, 1999, Florence, Italy.

Lazy Aggregates for Real-Time OLAP

Jukka Kiviniemi, Antoni Wolski, Antti Pesonen[+], Johannes Arminen

Technical Research Centre of Finland (VTT)
VTT Information Technology
P.O. Box 1201, 02044 VTT, Finland

{Jukka.Kiviniemi, Antoni.Wolski, Antti.Pesonen, Johannes.Arminen}@vtt.fi
http://www.vtt.fi/tte/projects/iolap

Abstract. In OLAP models, or data cubes, aggregates have to be recalculated when the underlying base data changes. This may cause performance problems in real-time OLAP systems, which continuously accommodate huge amounts of measurement data. To optimize the aggregate computations, a new consistency criterion called the tolerance invariant is proposed. Lazy aggregates are aggregates that are recalculated only when the tolerance invariant is violated, i.e., the error of the previously calculated aggregate exceeds the given tolerance. An industrial case study is presented. The prototype implementation is described, together with the performance results.

1 Introduction

Traditional OLAP (on-line analytical processing) [3] technology is developed mainly for commercial analysis needs. Data from various sources is compiled into multiattribute fact data tuples. They are used to compute multidimensional aggregates that make up the essence of OLAP.

An industrial process is often a subject of extensive analysis, too. However, the traditional OLAP technology is insufficient for such analysis. It lacks support for real-time operation and efficient time series analysis. Usually, the industrial analysis model, comprising of measurement data and derived data, must be up-to date instantly when the contents of the underlying data sources change. We claim that a certain level of temporal consistency [8] must be maintained, in an industrial analysis model.

In this paper, we present the requirements for Industrial OLAP (IOLAP) for industrial process analysis. We analyze problems encountered in a case study, having to do with handling time series data and maintaining consistency of the analysis structure in real-time. We introduce a method to decrease the amount of computation required to update the analysis model, by allowing the model to have some value-inaccuracy. We call this the lazy aggregation method. We present an example of a typical industrial application and summarize experimental results of applying the method.

[+]Currently on leave, at National Inst. of Standards and Technology, Gaithersburg, MD, U.S.A.

A theoretical abstraction of multidimensional analysis model, the data cube, was refined in [4]. The basic computation models for data cube were reviewed in [1]. A model for decomposition of data cube into a lattice was presented in [5]. Data cube maintenance and refreshing issues were addressed in [7]. We already introduced a time series database platform supporting active ECA rules [11].

The paper is organized as follows. In Section 2, we discuss IOLAP requirements. In Section 3, we introduce the concept of accuracy-based consistency enforcement and propose the lazy aggregate method. In Section 4, we focus on a case study and the prototype implementation. We conclude in Section 5.

2 IOLAP Model

Similarly to traditional OLAP models, the IOLAP data model involves base data and aggregate data. The base data reflects the current and past state of a process and is usually stored in a process database. An example of a process database management system implementation is given in [11].

Calculable data cubes may be represented as nodes in a combined aggregate lattice [5]. The lattice has *n+1* levels where *n* is the maximum number of dimensions. An example of a 4-level aggregate lattice is given in Figure 1.

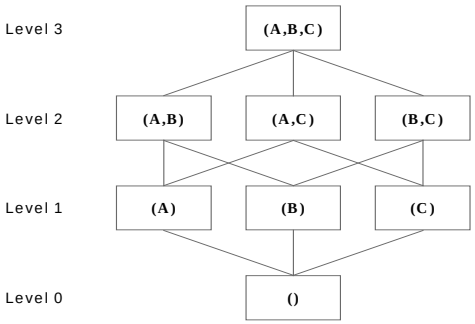

Figure 1. The aggregate lattice

The top level-3 ($l = n = 3$) aggregates are directly calculated from base data. They may be expressed as a tuple $< dim_A, dim_B, dim_C, fact >$ where *fact* is a value of an aggregate function over dimensions A, B, and C at a point (dim_A, dim_B, dim_C) of a 3-dimensional space. We chose to represent both the base and aggregate data using the relational model. For example, if the base data is stored in a relational table having the schema (id, dimA, dimB, dimC, value), the level 3 node for the AVG function is produced with the SQL query:

```
SELECT dimA, dimB, dimC, AVG(value)
FROM base_table
GROUP BY dimA, dimB, dimC;
```

The lower level aggregates can be calculated from precalculated upper-level aggregates. We say that the aggregates at level *l* are calculated from the source data at level

l+1(l = 0, ..., n-1) and the level *n* is aggregated from the base (fact) data. The zero-level node is a single value aggregated over the whole base data set. The lattice can be also called an *N-cube* meaning a collection of *N* cubes where $N = 2^n$.

The lattice may be materialized to a various extent. In two extreme cases, the aggregate data may be totally calculated at query time or they may be totally materialized in advance. There has been some research effort aimed at maintaining lattice consistency in a most efficient way, once the lattice is being refreshed [5, 7].

We take a different approach in this work. In IOLAP, the goal is to be able to perform real-time analysis using the latest process data. The temporal consistency of the analysis data is expected to be within sub-second range. We thus assume the lattice is fully materialized at all times, and we strive to avoid "insignificant" recalculations, i.e. such aggregate recalculations that would not be required from the accuracy point of view.

We propose a new consistency criterion that is based on numerical accuracy and leads to the concept of lazy aggregation. A user sets an allowable tolerance for an aggregate calculation error. Consequently, an aggregate value need not be recalculated if the existing value represents the current process within the given tolerance.

3 Lazy Aggregates

3.1 Definitions

Definition 1: Error band. Error band ϕ_v of variable *v* is defined as a maximum deviation of the measured value v_i from a specific reference norm.

$$\phi_v = \max_i (abs(\varepsilon(v_i))), \quad (1)$$

where *i* spans all measured occurrences of *v*, and $\varepsilon(v_i)$ is an actual measurement error of v_i. Error band is typically expressed as an absolute percentage of full scale, e.g. 10%, and it is normally associated with a measurement equipment.

Definition 2: Tolerance. Tolerance α_v of a variable *v* is an acceptable degree of variation of *v*. It is also usually expressed as an absolute percentage of full scale.

The above two concepts seem to be similar but there is a semantic difference between them: the error band stems from the physical characteristics of the measuring equipment, and the tolerance is an externally given requirement. Note, that we expect the measurement system to maintain, at any time the relation

$$\phi_v \leq \alpha_v \text{ for every variable v.} \quad (2)$$

Sometimes the error band values are not known, and the tolerance values are used instead. For simplicity of notation, we also assume that $\phi_v = \alpha_v$, for base data. For the purpose of evaluation of the error band of the aggregate values, we introduce the concept of the *base aggregate error band.*

Definition 3: Base aggregate error band (BEB). Base aggregate error band BEB_a of an aggregate *a* is a maximum deviation of the aggregate value from a real exact value. It is expressed as an absolute percentage of full scale. It is defined as a function:

$$BEB_a = f(S, A), \quad (3)$$

where $S = \{s_j \mid j = 1, ..., m\}$ is the set of available source values, $A = \{\alpha_j \mid j = 1, ..., m\}$ is the set of the corresponding tolerance values, and *m* is the size of the source value set. There is a binary (correspondence) relation $< s_j, \alpha_j > \subset R$.

As we deal with the effect of propagating errors from function arguments to the function result, various approaches to error propagation may be used in order to establish an appropriate BEB function for each aggregate. For example, if we treat measurement values as value intervals, the interval arithmetic [6] may be applied.

BEB functions for some aggregates are trivial. For example, for the Maximum aggregate:

$$BEB_{MAX} = \alpha_j \text{ such as } MAX(S) = v_j . \quad (4)$$

In the case of the Average aggregate, the following approximation may be sufficient:

$$BEB_{AVG} = AVG(A). \quad (5)$$

We use BEB to characterize each lattice node element in terms of its maximum accuracy, i.e. the accuracy achieved when the node element value (aggregate value) is recalculated form the source data having some tolerance. We now proceed to characterize the error induced by *not* recalculating the aggregate.

We introduce the concept of an *actual error band* to reflect the instantaneous error band of an aggregate. It is recalculated each time there is a change to source data.

Definition 4: Actual aggregate error band (AEB). Actual aggregate error band AEB of a node element is defined as

$$AEB_a = BEB_a + abs(\sum_i \delta_i) \quad (6)$$

where δ_i is the error delta calculated for each i-th change (transaction) affecting a source data variable:

$$\delta_i = f_a(v_i, v\text{-}old_i , P_v) \quad (7)$$

where v_i and $v\text{-}old_i$ are the new and old values of the source data and P_v is the power (cardinality) of the set of v_i values being aggregated. The function is aggregate-specific. The value of *i* is set to zero each time the aggregate value is recalculated. Evaluation of AEB is triggered by any replacement of $v\text{-}old_i$ with v_i.

As a consistency criterion, we propose to maintain, in a lattice of aggregates, the following invariant:

Definition 5: Tolerance invariant (consistency criterion). For each *j-th* element at the *p-th* lattice node, the following holds:

$$\forall_{p,j} AEB_j^p \leq \alpha_p \quad \textbf{(8)}$$

where α_p is a tolerance value associated with the aggregate at the node *p*.

3.2 Computational Model

The idea of the lazy aggregates is to delay the aggregate recalculation until the actual error band of the aggregate value exceeds the given tolerance. We propose to maintain tolerance invariant by way of a standard ECA rule (trigger) mechanism [9]. An ECA rule is associated with each aggregate at levels 1, ..., n. The aggregate recalculation at any level takes place in the following steps.

1. Event detection: a change of data source at level *l* is detected.
2. Condition evaluation: the tolerance invariant at level *l-1* is checked
3. Action execution (conditional): the aggregate at level *l-1* is recalculated

As a result, the change in the base data is propagated downward the lattice in an optimized way. A possible scenario of aggregate recalculation is shown in Figure 2.

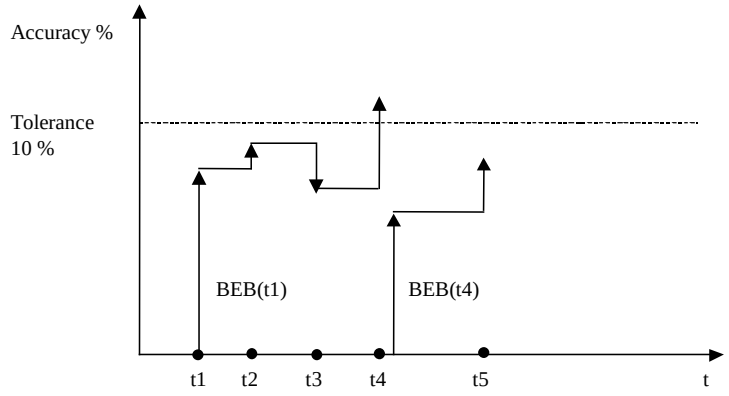

Figure 2. An example of lazy aggregate computation

In the example, at time *t1*, the aggregated power consumption is calculated and the base error band (BEB) is evaluated. At *t2* and *t3*, the new measurement values are arrived and the evaluation of the tolerance invariant is fired. The corresponding error deltas are calculated but no aggregate recalculation is triggered as the resulting actual error band (AEB) is still within the tolerance. At *t4*, the accumulated AEB exceeds the tolerance value, which leads to aggregate recalculation and re-evaluation of BEB. At *t5*, again, the tolerance invariant holds, and no aggregate recalculation is needed.

4 Implementation and Results

4.1 Case Study

ABB Industry Oy is a leading manufacturer of electric drive systems for heavy industry. A typical product is a paper machine drive system [2]. It consists of few tens of high-power electric motors (drives), together with the associated frequency converters, and a control and supervision system. In the case study, the requirement is to be able to survey the overall drive operation in real-time. Drives are combined into drive sections which, in turn, are grouped according to machine parts such as wire, press and dryer. Several paper machines may be surveyed at a factory or at different locations, at the same time. Possible dimensions to be used in motor behavior analysis are thus: *section*, *machine part*, *machine*, *location*, *power rang*e, *type* and *manufacturing year*. Some of the dimensions may be considered different granularities of a single dimension. For example, machine part, machine and location may be granularities of the geographical dimension.

The variables measured at each motor are, typically, temperature, power and torque. The measured values constitute the base data of the case study IOLAP model. The data is collected at typical rate of one measurement record (a tuple of all measurement values) per second per motor. For a 100-motor installation, the update rate of 100/s is attained, for any variable type. If we required that the lattice node values are recalculated each time a source value changes, the required aggregate recalculation rate would be $(100 \bullet 2^n)/s = 1600/s$. Such a rate would not be feasible on a low-cost PC equipment. We apply the lazy aggregation method to reduce the recalculation rate significantly.

4.2 Prototype Implementation

We have implemented a prototype of a general-purpose N-cube server called Rubic [10]. Rubic is based on the existing RapidBase active time series database system [12]. The aggregates in the data cube lattice are organized as relational database tables. Triggers are associated with all time series and aggregates.

All the functionality of checking the tolerance invariant and computing the error band values is implemented in a detached Rubic Aggregate Engine process. When any value in time series or an aggregate node is changed, an appropriate trigger is fired and the aggregate engine handles the action. The action execution starts with the tolerance invariant checking and, depending on the result, either the actual error band is updated or the lattice node value is recalculated.

The analysis data is generated with a process data generator, which feeds new measurements values for each motor periodically. The aggregated results are analyzed using some general-purpose reporting tool connected to RapidBase via the ODBC driver.

4.3 Performance Results

The test scenario consists of 100 motors and the power of each motor is measured. The measurements are assumed to have a random walk behavior and each motor is handled independently.

The motors are analyzed on the basis of four dimensions (*type*, *power range*, *year of manufacturing*, and *machine part*). Thus, the used aggregate lattice contains 2^4=16 tables, which are all materialized. The experiment is performed separately for two aggregate functions, AVG(power) and SUM(power). These functions were selected to represent distributive and algebraic aggregate function groups [4], respectively. We do not address holistic aggregate functions in our experiment. The arrival rate of the measurement values for each motor is 1/s. The tolerance values are varied within a range of 2% to 20%.

During the test run, the number of lattice recalculations is measured. The lattice recalculation is performed, when an updated error band exceeds the given tolerance. The result is expressed as a lattice recalculation percentage that is calculated with the following formula (for a given period of time):

$$RECALC\% = \frac{no.\_of\_lattice\_element\_recalculations}{2^{no.\_of\_\dim.} * no.\_of\_input\_transactions} \quad (9)$$

The test was run under Windows NT 4 in a 333 MHz Pentium Pro PC with main memory of 128Mb. The experimental results are presented in Figure 3.

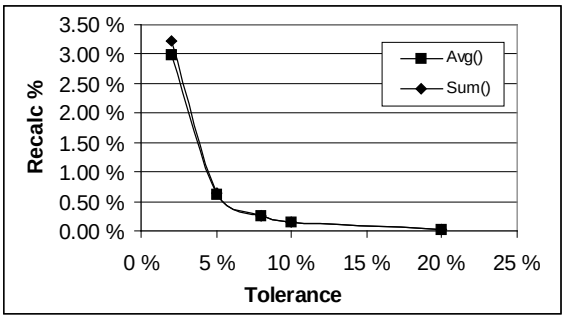

Figure 3. Experimental results

The experimental results begin with a tolerance value of 2%. The lower values were not attainable due to performance limitations. It can be clearly seen that, by using affordable tolerance level, say 5%, it is possible to reduce the number of lattice recalculations drastically. Furthermore, regardless of the aggregate function type, the performance is alike. However, neither function is demanding in terms of complexity nor are their BEB and AEG functions. More complex algebraic aggregate function may cause indeterminate performance degradation when using the lazy aggregate method.

5 Conclusions

We discussed the requirements for using OLAP in industrial analysis, and we found the OLAP concept useful for the purpose. However, the traditional OLAP approach must be enhanced to fulfill industrial analysis needs. Typical industrial data is time-based and it has a fast arrival rate. The accuracy of the data at various levels of the OLAP structure (data cube) varies in time as the underlying base data changes. We took the advantage of this phenomenon to utilize the accuracy of computations in the performance optimization scheme called lazy aggregates. Lazy aggregates are the aggregates that are calculated only if the accuracy of the previously calculated values is not within given tolerance. We defined the concepts of error band and the consistency criterion in the form of the tolerance invariant. We propose, how the lazy aggregate method can be implemented using an active main memory database. We provide a case study based on a paper industry application. We also provide a prototype implementation of a general-purpose N-cube server called Rubic. We analyze our lazy aggregate approach using Rubic and we find that, by using the lazy aggregate concept, it is possible to perform complex industrial analysis on a standard PC platform in the presence of fast data acquisition.

References

1. S. Agarwal, R. Agrawal, P. M. Deshpandre, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi. On the Computation of Multidimensional Aggregates. *Proc. of the 22nd VLDB Conference*, Bombay, India, 1996, pp. 506-521.
2. PPS 200: The papermaker's Drive. ABB Industry Oy, 1998, Helsinki, Finland.
3. F. Codd, et al. Providing OLAP to User-Analysts: An IT Mandate. TR, E.F. Codd & Associates, 1993 (http://www.arborsoft.com/essbase/wht_ppr/ coddTOC.html)
4. J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Proc. of 12th Internat. Conf. on Data Engineering*, New Orleans, Louisiana, U.S.A., 1996, pp. 152-159.
5. V. Harinarayan, A. Rajaman, J. D. Ullman. Implementing Data Cubes Efficiently. *SIGMOD Record*, Vol. 25, No. 2, June 1996, pp. 205-216.
6. R. Moore. Interval Analysis, Prentice Hall, 1966.
7. I. S. Mumick, D. Quass, B. S. Mumick. Maintenance of Data Cubes and Summary Tables in a Warehouse. *Proc. of the 1997 SIGMOD. Conf.*, Tucson. AZ, U.S.A., pp. 100-111.
8. K. Ramamritham. Real-Time Databases. *Distributed and Parallel Databases*, 1(2), April 1993, pp. 199-226.
9. J. Widom, S. Ceri (eds.). Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, 1996.
10. A. Wolski, J. Arminen, J. Kiviniemi, A. Pesonen. Design of Rubic, Version 1.0. Research Report TTE1-1-99, VTT Information Technology, January 1999. (http://www.vtt.fi/-tte/projects/industrialdb/publs/rubic-design.pdf)
11. A. Wolski, J. Karvonen, A. Puolakka. The RAPID Case Study: Requirements for and the Design of a Fast-response Database System. *Proceedings of the First Workshop on real-Time Databases* (RTDB'96), Newport Beach, CA, USA, 1996, pp. 32-39 (http:-//www.vtt.fi/tte/projects/industrialdb/publs/case.pdf)
12. RapidBase Home Page, VTT Information Technology, 1999. (http://www.vtt.fi/tte/projects/rapid/)