Chapter XII
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Shi-Ming Huang
National Chung Cheng University, Taiwan

John Tait
Information Retrieval Faculty, Austria

Chun-Hao Su
National Chung Cheng University, Taiwan

Chih-Fong Tsai
National Central University, Taiwan

ABSTRACT

Data warehousing is a popular technology, which aims at improving decision-making ability. As the result of an increasingly competitive environment, many companies are adopting a “bottom-up” approach to construct a data warehouse, since it is more likely to be on time and within budget. However, multiple independent data marts/cubes can easily cause problematic data inconsistency for anomalous update transactions, which leads to biased decision-making. This research focuses on solving the data inconsistency problem and proposing a temporal-based data consistency mechanism (TDCM) to maintain data consistency. From a relative time perspective, we use an active rule (standard ECA rule) to monitor the user query event and use a metadata approach to record related information. This both builds relationships between the different data cubes, and allows a user to define a VIT (valid interval temporal) threshold to identify the validity of interval that is a threshold to maintain data consistency. Moreover, we propose a consistency update method to update inconsistent data cubes, which can ensure all pieces of information are temporally consistent.
INTRODUCTION

Background

Designing and constructing a data warehouse for an enterprise is a very complicated and iterative process since it involves aggregation of data from many different departments and extract, transform, load (ETL) processing (Bellatreche et al., 2001). Currently, there are two basic strategies to implementing a data warehouse, “top-down” and “bottom-up” (Shin, 2002), each with its own strengths, weaknesses, and using the appropriate uses.

Constructing a data warehouse system using the bottom-up approach will be more likely to be on time and within budget. But inconsistent and irreconcilable results may be transmitted from one data mart to the next due to independent data marts or data cubes (e.g. distinct updates time for each data cube) (Inmon, 1998). Thus, inconsistent data in the recognition of events may require a number of further considerations to be taken into account (Shin, 2002; Bruckner et. al, 2001; Song & Liu, 1995):

- **Data availability**: Typical update patterns for a traditional data warehouse on weekly or even monthly basis will delay discovery, so information is unavailable for knowledge workers or decision makers.
- **Data comparability**: In order to analyze from different perspectives, or even go a step further to look for more specific information, data comparability is an important issue.

Real-time updating in a data warehouse might be a solution which can enable data warehouses to react “just-in-time” and also provide the best consistency (Bruckner et al., 2001) (e.g. real-time data warehouse). But, not everyone needs or can benefit from a real-time data warehouse. In fact, it is highly possible that only a relatively small portion of the business community will realize a justifiable ROI (return on investment) from a real time data warehouse (Vandermay J., 2001). Real-time data warehouses are expensive to build, requiring a significantly higher level of support and significantly greater investment in infrastructure than a traditional data warehouse. In additional, real-time update will also require high time cost for response and huge storage space for aggregation.

As a result, it is desirable to find an alternative solution for data consistency in a data warehouse system (DWS) which can achieve near real-time outcome but does not require a high cost.

Motivation and Objective

Integrating active rules and data warehouse systems has been one of the most important treads in data warehousing (DM Review, 2001). Active rules have also been used in databases for several years (Paton & Daz, 1999; Roddick & Schrefl, 2000), and much research has been done in this field. It is possible to construct relations between different data cubes or even the data marts. However, anomalous updates could occur when each of the data marts has its own timestamp for obtaining the same data source. Therefore, problems with controlling data consistency in data marts/data cubes are raised.

There have been numerous studies discussing the maintenance of data cubes dealing with the space problem and retrieval efficiency, either by pre-computing a subset of the “possible group-bys” (Harinarayan et al., 1996; Gupta et al., 1997; Baralis et al., 1997), estimating the values of the group-bys using approximation (Gibbons & Matias, 1998; Acharya et al., 2000) or by using online aggregation techniques (Hellerstein et al., 1997; Gray et al., 1996). However, these solutions still focus on single data cube consistency, not on the overall data warehouse environment’s respective. Thus, each department in the enterprise will still face problems of temporal inconsistency over time.
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

In the paper, we seek to develop temporal-based data consistency by proposing a Temporal-based Data Consistency Mechanism (TDCM) as an alternative solution for data consistency in data warehouse systems (DWS). Through our TDCM, we can ensure that all related information retrieved from a DWS in on a consistent time basis. Thus, this mechanism can enhance data quality and potentially increase real-world competitiveness.

RELATED WORK

Active Rule and Data Warehouse Integration

Active rules have been used in databases for several years (Paton & Dazo, 1999). Most active database rules are defined by production rules and often an event-based rule language, in which a rule is triggered by an event such as the insertion, deletion or modification of data. The event-condition-action (ECA) model for active database is widely used, in which the general form of rules is as follows:

On event
If condition
Then action

The rule is triggered when the event occurs. Once the rule is triggered then, the condition is checked. If the condition is satisfied, the action is executed. Ariel (Hanson, 1996), STRIP (Adelberg et al., 1997), Ode (Arlein et al., 1995), and HiPAC (Paton & Dazo, 1999) are all systems of this type. The aim of an active database is to (1) perform automatic monitoring of conditions defined over the database state (2) take action (possibly subject to timing constraints) when the state of the underlying database changes (transaction-triggered processing).

Active rules have also been integrated into data warehouse architecture recently to provide further analysis, real-time reaction, or materialized views (Thalhammer et al., 2001; Huang et al., 2000; Adelberg, 1997). Also recently data warehouse vendors have concentrated on real-time reaction and response in actual applications, in their active data warehouse systems (Dorinne, 2001).

View Maintenance and Temporal Consistency

Materialized Data Consistency of View Maintenance

Many researchers have studied the view maintenance problem in general (Yang et al., 2000; Ling et al., 1999; Zhuge et al., 1995; Gupta & Mumick, 1995) and a survey of the view maintenance literature can be found in Gupta & Mumick, (1995; Ciferri, 2001), where views are defined as a subset of relational algebraic expressions.

Maintaining the consistency of materialized views in a data warehouse environment is much more complex than maintaining consistency in single database systems (Ciferri, 2001). Following the aforementioned literature, we separate view maintenance approaches into two parts: “Incremental Maintenance” and “Self-Maintenance”.

“Incremental Maintenance” is a popular approach to maintaining materialized view consistency (Saeki et al., 2002; Ling et al., 1999), and it is characterized by access through the base data sources. In contrast, the characteristic of “Self-Maintenance” is maintaining materialized view without access to the base data (Ciferri et al., 2001), because base data comes from sources that may be inaccessible. Furthermore, it may be very expensive or time-consuming to query the databases. Thus, to minimize or simply not to perform external access on those information sources during the maintenance process represents an important incremental view maintenance issue (Amo, 2000; Yang et al., 2000). Table 1, illustrates there two materialized view maintenance approaches.
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Temporal Consistency

The term “temporal consistency” comes from Real-Time DB systems (Heresar et. al, 1999), where the value of objects must correctly reflect the state of environment. Previous work (Ramamritham, 1993; Xiong et al., 1996; Tomic et al., 2000) has defined temporal consistency in real-time database systems (RTDBS) as follows. An RTDBS must maintain absolute consistency, that is, any changes in the real world should be promptly reflected in the database. If the age of a data object is within some specified threshold, called the absolute threshold “Ta”, the data object is absolutely consistent. An RTDBS must also maintain relative consistency, so that the data presents a consistent snapshot of the real world at a given time. Relative consistency requires that the set of data objects is considered to be relatively consistent if the dispersion of ages is smaller than a relative threshold “Tr”.

Temporal consistency was also defined using validity intervals in a real-time database to address the consistency issue between the real world state and the reflected value in the database (Ramamritham, 1993). Temporal data can be further classified into base data and derived data. Base data objects import the view of the outside environment. In contrast derived data object can be derived from possibly multiple base/derived data (e.g. data warehouse repository).

In this research, we focus on maintaining data consistency within a temporal-based consistency approach when users send a query from DWS. Mumick (1997) proposes a method of maintaining an aggregate view (called summary-delta table method), and uses it to maintain summary tables in the data warehouse. Like many other incremental view maintenance techniques, we use a “delta” table to record insertion and deletion in the source data. We also combine active rule and temporal consistency concepts and adjusted these methods to construct a Temporal-based Data Consistency Mechanism (TDCM), through which we are able to simultaneously update related data cubes from different data marts.

### TEMPORAL-BASED DATA CONSISTENCY

#### Overview

The proposed TDCM is an alternative solution for data warehouse system (DWS), which uses active rules to maintain data consistency. Because temporal consistency is often ensured either by extended use of time stamps, or by validity status (Bruckner et al., 2001), we let knowledge workers or decision makers define a VIT (Valid Interval Temporal) as a threshold in this mechanism. This ensures that every piece of information captured

<table>
<thead>
<tr>
<th>Materialized View Maintenance Approaches</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Incremental Maintenance</strong>&lt;br&gt;(Saeki, 2002)&lt;br&gt;(Moro, 2001)&lt;br&gt;(Ling, 1999)</td>
<td>Maintain a materialized view in response to modifications to the base relations. Applicable to nearly all types of database updates. It is more efficient to apply this algorithm to the view than to re-compute the view from the database.</td>
</tr>
<tr>
<td><strong>Self-Maintenance</strong>&lt;br&gt;(Amo, 2000)&lt;br&gt;(Yang, 2000)&lt;br&gt;(Samtani, 1999)</td>
<td>Maintain the materialized views at the DW without access to the base relations. (e.g. by replicating all or parts of the base data at the DW or utilizing the key constraints information.)</td>
</tr>
</tbody>
</table>
from a DWS is delivered in a temporally correct manner. We define Temporal-based Data Consistency as following:

**Definition:** Temporal-based Data Consistency (TDC)

The set the dispersion of data object remains within a specified threshold, \( VIT \). The threshold \( VIT \) reflects the temporal requirements of the application. The dispersion of two data objects \( x_i \) and \( x_j \), denoted as \( T(x_i, x_j) \) is defined as \( T(x_i, x_j) = |t(x_i) - t(x_j)| \), where \( t(x_i) \) and \( t(x_j) \) are the timestamps of two objects, \( x_i \) and \( x_j \). Thus, the set \( S \) of data objects is said to have Temporal-based Data Consistency if:

\[
\forall x_i, x_j \in S, T(x_i, x_j) \leq VIT(S).
\]

**Temporal-based Data Consistency Mechanism**

**Data Warehouse Event and Active Rule Syntax**

In a data warehouse environment, multi-dimensional query events can be classified into dimension events and measurement events (Huang et al., 2000; Gray et al., 1996). Figure 1 illustrates the event classification in multi-dimensional query and data consistency.

Figure 2 shows our active rule syntax, which includes two parts: a rule body and a coupling model. The rule body describes the ECA base active rules, while the coupling model describes how the active rules can be integrated into the database query. The rule body is composed of three main components: a query predicate, an optional condition, and an action. The query predicate controls rule triggering; and the condi-

**Figure 2: Active rule syntax**

<table>
<thead>
<tr>
<th>Rule Body:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Rule ::= &lt;rule-name&gt;</td>
</tr>
<tr>
<td>On ::= &lt;query-predicate&gt;</td>
</tr>
<tr>
<td>[ if &lt;conditions&gt; := true ]</td>
</tr>
<tr>
<td>then</td>
</tr>
<tr>
<td>[evaluate query-commalist]</td>
</tr>
<tr>
<td>execute ::= &lt;action&gt;</td>
</tr>
<tr>
<td>query-predicate ::= event [,event [,event]]</td>
</tr>
<tr>
<td>event ::= drill down</td>
</tr>
<tr>
<td>pull</td>
</tr>
<tr>
<td>alter</td>
</tr>
<tr>
<td>condition ::= query-commalist</td>
</tr>
<tr>
<td>query-commalist ::= query [,query]*</td>
</tr>
<tr>
<td>query ::= table-expression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coupling Model:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query_coupling = Same</td>
</tr>
<tr>
<td>EC_coupling = Before</td>
</tr>
<tr>
<td>CA_coupling = Immediate</td>
</tr>
<tr>
<td>Execute_mode = Repeat</td>
</tr>
<tr>
<td>[precedes &lt;rule_names&gt;]</td>
</tr>
<tr>
<td>[follows &lt;rule_names&gt;]</td>
</tr>
</tbody>
</table>

**Figure 1. Event classification**
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Section specifies an additional predicate that must be true if a triggered rule is to automatically execute its action. Active rules are triggered by database state transitions—that is, by execution of operation blocks. After a given transition, those rules whose transition predicate holds with respect to the effect of the transition are triggered. The coupling models give database designers the flexibility of deciding how the rule query is integrated within the Multi-Dimensional Query (MDQ) (Gingras, & Lakshmanan, 1998).

There are five different execution attributes to determine the semantic of an active rule, as follows:

**Query_coupling**: treating the execution of a rule as a query in DWS, e.g., a rule query. If the Query_coupling is set to ‘same’, then the MDQ is committed only when the RQ (Rule Query) and DQ (Data Query) are both committed. If the Query_coupling is set to ‘separate’, then the MDQ commitment will depend only on the DQ. This suggests that the Query_coupling should be set to ‘Separate’ when the active rule does not have any effect on the DQ, in order to enhance the system performance by reducing query execution time.

**EC_coupling**: defining the execution sequence of the event and condition part for a relational active rule. The ‘before’ EC_coupling means that the rule condition is evaluated immediately before the DQ is executed. The ‘after’ EC_coupling means that the rule condition is evaluated after the DQ is in the prepare-to-commit state.

**CA_coupling**: presenting the execution sequence of the condition and action part for an active rule. The ‘immediate’ CA_coupling means that the rule action is executed immediately after the rule condition is evaluated and satisfied. The rule action executed after DQ is in the prepare-to-commit state, when CA_coupling is specified to ‘defer’.

**Execute_mode**: the triggered rule will automatically be deactivated after it is committed, when its Execute_mode is specified as ‘once’.

On the other hand, the rule is always active if its Execute_mode is specified to ‘repeat’.

**Precedes_follows**: The optional ‘precedes’ and ‘follows’ clauses are used to induce a partial ordering on the set of defined rules. If a rule r1 specifies a rule r2 in its ‘precedes’ list, or if r2 specifies r1 in its ‘follows’ list, then r1 is higher than r2 in the ordering.

Temporal-Based Data Consistency and Active Rule Integration

Active rules have been integrated into data warehouse architecture to maintain data consistency in the materialized views (Adelberg, 1997). Using temporal perspective, anomalies updated to obtain timely information by end-users’ queries will cause data inconsistencies in daily transactions. In this research, we focus on temporal-based data consistency as defined previously, according to which, the TDC Evaluation Protocol is described as in Figure 3.

The following example (Figure 4) is of integrated active rule and temporal-based data consistency evaluation protocol to maintain data consistency. When a user is browsing Data CubeA and using a drill-down OLAP operation (In “Months” level and Measurement<= “20”),

Figure 3. TDC Evaluation protocol

```plaintext
Temporal-based Data Consistency Evaluation Protocol
//Set the timestamp of object x_i, t(x_i)
//Set the timestamp of object x_j, t(x_j)
For each related object
IF t(x_i) = t(x_j) THEN
//Temporal-based Data Consistency
ELSE
IF |t(x_i) - t(x_j)| <= VIT (Valid Interval Temporal) THEN
//Temporal-based Data Consistency
ELSE
IF t(x_i) = t(x_j) THEN
Consistency Update x_i From t(x_i) to t(x_j)
ELSE
Consistency Update x_i From t(x_i) to t(x_j)
//Temporal-based Data Consistency
END IF
END IF
END IF
```
the active rule Analysis-Rule1 will be triggered for rule evaluation. If the user needs to retrieve more detail or related information from other data cubes, TDCM will be launched to maintain data consistency. Through the timestamp of each data cube and VIT threshold, we are able to decide which data cube needs to be updated.

**Active Rule Activation Model**

This section discusses our active rule activation model by extending the model specified in (Paton & Daz, 1999; Huang et al., 2000) which shows how a set of rules is treated at runtime. The execution sequence of data query and triggered rules will influence the result and correctness of active rules. The coupling model provides more semantics for rule triggering and execution. Our temporal-based data consistency mechanism working process is shown in Figure 5.

- The **Signaling** phase includes to the appearance of an event occurrence caused by an event source.
- The **Triggering** phase takes the events produced and triggers the corresponding rules. The association of a rule with its event occurrence forms a rule instantiation.
- The **CE (Condition Evaluation):** The true phase evaluates the condition of the triggered rules which are satisfied.
- The **RE (Relation Evaluation):** The true phase evaluates the relations between different data objects that have existed or not.
- The **IE (Inconsistency):** The true phase detects a data inconsistency with related data object caused by a user anomaly updating a daily transaction. It will be considered inconsistent if the dispersion interval of two data objects is smaller then VIT threshold.
- The **Scheduling** phase indicates how the rule conflict set is processed. In this model,

---

**Figure 4. Active rule for data consistency within TDCM**

```
Define Rule Analysis-Rule1
//E (Event)
On dimensional drill down
//C (Condition)
If {Level = “Months” and
Dimensions = (“Product”, “Years”),
and Measurement = “TQuantity”, and Measurement<= “20”, and
Select Years, Months, Product, TQuantity
From CubeA
Where Product = “ALL” and MONTH>= “7” or MONTH <= “12”}
//A (Action)
then execute {
    // Temporal-based Data Consistency Evaluation Protocol
    // Set t1 and t2 are the timestamp of CubeA and CubeB
    IF | t1-t2 | <= 1 (Month) Then
        Retrieve “CubeB”
    ELSE
        IF t1 > t2 then
            Consistency_Update (CubeB to t1)
        ELSE
            Consistency_Update (CubeA to t2)
        END IF
    END IF
}
Coupling Model:
Query_coupling = Separate
EC_coupling = After
CA_coupling = Deferred
Execute_Mode = Once
```
rules are partially ordered. For any two rules, one rule can be specified as having higher priority than an other rule without ordering being required.

The semantics of the data warehouse active rule syntax determines how rule processing will take place at run-time once a set of rules has been defined. It also determines how rules will interact with the arbitrary data warehouse event and queries that are submitted by users and application programs. Even for relatively small rule sets, rule behavior can be complex and unpredictable, so precise execution semantics is essential (Huang et al, 2000). Figure 7 presents the detailed rule activation processing flow of our system. The detail rule activation process flow is as seen in Figure 6.

**Figure 6. The detail rule activation process flow.**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | Query coupling evaluation:  
|      | If *Query_coupling* is *Separate*, the system will submit the triggered rule to *QM* (query manager) as a new query. Otherwise, the system will proceed with the following steps. |
| 2    | Event-Condition coupling evaluation: *Before*  
|      |  2a. Reasoning rules, which its *EC_coupling* is equal to *Before*.  
|      |  2b. If the condition evaluation result is true, then the following two possible situations may happen.  
|      |     2b.1. The action part will be executed immediately if its *CA_coupling* is equal to *Immediate*.  
|      |     2b.2. The action part will be saved into a queue if its *CA_coupling* is equal to *Deferred*.  
|      |  2c. Repeating the steps 2a, 2b until no more rules are reasoned by step 2a. |
| 3    | Executing the data query. |
| 4    | Executing the queued rules, which are stored by step 2b.2. |
| 5    | Event-Condition evaluation: *After*:  
|      |  5a. Reasoning rules, which its *EC_coupling* is equal to *After*.  
|      |  5b. If the condition evaluation result is true, there are the following two possible situations.  
|      |     5b.1. The action part will be executed immediately if its *CA_coupling* is equal to *Immediate*.  
|      |     5b.2. The action part will be saved into a queue if its *CA_coupling* is equal to *Deferred*.  
|      |  5c. Repeating steps 5a, 5b until no more rules is reasoned by step 5a. |
| 6    | Executing queued rules, which are stored by step 5b.2. |
| 7    | Committing the query if and only if all sub-queries are committed. |
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Figure 7. Process flow of rule activation
Taxonomy for Situations of Temporally-Based Data Consistency Mechanism

We can identify several possible different situations for the Temporally-based Data Consistency Mechanism. We can have at least four distinct situations: 1. timestamp of all data cubes are the same; 2. timestamp of one data cube is expired; 3. timestamp of one data cube is new; 4. all of the timestamps of the data cubes are different from each other.

Consider that there are three data cubes (Data Cube1, Data Cube2, and Data Cube3) and three different times (t1, t2, and t3). Suppose t1 > t2 > t3 and the browsing sequence is Data Cube1—Data Cube2—Data Cube3. Thus, we can classify these events into several different situations:

In situation 1, the timestamp of all three data cubes are the same ("t1"). Thus, we do not have to update any data cube. According to our definition, they are temporally consistent.

In situation 2, the timestamp of Data Cube1 (t1) is not equal to Data Cube2 (t2), so Data Cube2 and Data Cube3 have temporal-based data consistency. As a result, when user browsing Data Cube1 and Data Cube2, our mechanism will update Data Cube1 (from t1 to t2). Thus our mechanism will update once for temporal-based data consistency.

In situation 3, the timestamp of Data Cube1 is equal to Data Cube2 (TDC), so Data Cube2 and Data Cube3 are inconsistent. As a result, when users are browsing Data Cube1 and Data Cube2, they do not have to update; but when users are browsing Data Cube2 and Data Cube3, our mechanism will update both Data Cube2 (from t2 to t3) and Data Cube1 (from t2 to t3). Thus, our mechanism will update twice for temporal-based data consistency.

Situation 1. (t1, t1, t1)

Situation 2. (t1, t2, t2)
Situation 3. \((t2, t2, t3)\)
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

In situation 4, the timestamps of all three data cubes are different, so when a user is browsing Data Cube1 and Data Cube2, our mechanism will update Data Cube1 (from \(t_1\) to \(t_2\)); when a user is browsing Data Cube2 and Data Cube3, our mechanism will update both Data Cube1 (from \(t_2\) to \(t_3\)) and Data Cube (from \(t_2\) to \(t_3\)). Thus, our mechanism will update 3 times for temporal-based data consistency.

**Summary**

In this section, we introduce a methodology to develop the TDCM. Through active rule and metadata repositories, we can provide consistent data to knowledge workers or decision makers when they query a data warehouse system. Using active rules to maintain temporal-based data consistency of stored facts does not guarantee a
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

timely, correct view of the modeled real world. But it does ensure that every piece of information captured by a data warehouse system is provided in a temporally consistent framework.

SYSTEM IMPLEMENTATION

In this section, a prototype system is implemented to demonstrate the feasibility to our mechanism. Our prototype system is based on a multi-tier environment. The client is an active cube browser system, which is coded by using JDK (Java Development Kit). The middle tier is the active data warehouse engine, which is written in Visual Basic. The cube database is designed in Microsoft SQL Server 2000. Figure 8 shows the architecture of the prototype system.

Active Data Warehouse Engine

Data Cube Manager

Our Data Cube Manager provides an easy method to generate a data cube. The algorithm of data cube creation we use was proposed by Gray et al. (1996). There are two kinds of data, which will be moved to our system. One is dimension data for the cube, and the other is fact data for the cube.

Creating a data cube requires generating the power set (set of all subsets) of the aggregation columns. Since the cube is an aggregation opera-
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

In the context of data warehousing, it makes sense to externalize it by overloading the SQL GROUP BY operator. In fact, the cube is a relational operator, with GROUP BY and ROLL UP as degenerate forms of the operator. Overloading the SQL GROUP BY can conveniently specify by overloading the SQL GROUP BY. If there are N dimensions and M measurements in the data cube, there will be $2^N - 1$ super-aggregate values. If the cardinality of the N attributes are $D_1, D_2, ..., D_N$ then the cardinality of the result of cube relation is $\Pi(D_i + 1)$. Figure 9 shows the fact data algorithm.

**Active Rule Manager**

The active rule manager is specified the rules of data cube, and the grammar of active rules in our system follows the standard ECA (Event, Condition, and Action) rule. We designed an Active Rule Wizard, which is included with a friendly user interface and takes the designer through four easy steps to construct an active rule. Figure 10 shows the active rule construction process.

**Two Metadata Repositories in the Implementation**

The Metadata Repository and the Metadata Manager are responsible for storing schema (Meta-Model) and providing metadata management. Thus there are two metadata repositories in our system, as follows:

---

![Figure 10. Active rule construction process](image-url)
• Star schema metadata

The most popular design technique used to implement a data warehouse is the star schema. The star schema structure takes advantage of typical decision support queries by using one central fact table for the subject area and many dimension tables containing de-normalized descriptions of the facts. After the fact table is created, OLAP tools can be used to pre-aggregate commonly accessed information. Figure 11 displays the OMT model of star schema metadata.

• Active rule schema metadata

Many useful semantics are included in the proposed active rule schema metadata. The active rule schema is in two parts: a rule body table and a coupling model table. The rule body table is used to describe the ECA base active rules schema and the coupling model table is used to describe how the active rules can be integrated into the MDQ. Figure 12 presents an OMT model of an active rule schema.

Active Data Cube Browser

The active cube browser provides an OLAP function for user queries. When users browse cubes,
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Figure 13. Active cube browser

several OLAP events (e.g. Drill-Down, Roll-Up, Slice, Dice…) will be triggered. The data warehouse engine detects the event and employs an active rule mechanism to go one step further to analyze and return a warning to the active cube browser. At the same time, our mechanism will also follow a consistency rule to automatically maintain temporal consistency.

Moreover, in order to represent the query result in an easily understood manner, we use the adopting JFreeChart API from an open-source organization called JFreeChart. Our active cube browser provides several charts (e.g. Pie Chart, 3D Pie Chart, Horizontal Bar Chart, Vertical Bar Chart…) for clearer comparison analysis. Figure 13 shows the Data Warehouse Engine detection when a dimension drill down event occurs, the rule of DiscussCube is triggered to browse another cube and be shown in the Active Frame.

SYSTEM EVALUATION

System Simulation

Previous studies (Song & Liu, 1995) have considered only a general measure of temporal consistency, called %Inconsistency, which indicates the percentage of transactions which are either absolutely or relatively inconsistent. In this simulation, we use the number of inconsistencies to measure temporal consistency in top-down and bottom-up architecture.

Definition: Number of Inconsistencies

The number of all possible data inconsistencies due to user anomalies in updating transactions.
We used the following parameters in this simulation for purposes stated:

- The Number of Related Data Cubes (|N|): To decide how many related data cubes are used in a simulation run.
- Valid Interval Temporal (|VIT|): The threshold value specifies the temporal-based data consistency of data required by the DWS. The time interval of each two data objects with greater then VIT is considered out-of-date. In our simulation, we give VIT the same value of 1. We expected the number of inconsistency would be smaller as the value of VIT increased.
- The Number of update transactions in a period (|U|): The user anomalies update transactions to retrieve the newest data from the DWS. In our simulation, we use a randomizer to decide which data cubes will be updated.
- Simulation Run Periods (|P|): Total run period in our simulation.
- Simulation Times (|T|): The total execution time of our simulation.

In each series of experiments, we started to simulate the number of inconsistencies with a transitional Bottom-up and Top-Down data warehouse architecture. In the Bottom-up data warehouse architecture, given |N| is 10, |VIT| is 1, |U| 3, and |P| is 10. The objective of our TDCM is to avoid possible inconsistent situations under the Bottom-up architecture.

In the Top-down data warehouse architecture, we gave the same parameters for the simulation program. The only difference between Bottom-up and Top-down is that Top-down architecture has a reload period (set reload period is 3) which can centrally refresh the data warehouse after a specified period.

As time proceeds, the number of inconsistencies will increase with Top-down or Bottom-up architectures, a problem our TDCM seeks to resolve. With detailed investigation, we show the number of inconsistencies will increase as the related number of data cubes |N| increases.

**Number of Update Comparison**

According to our definition of temporal-based data consistency, we use a consistent update for each of two related data objects that are considered out-of-date. As we described in chapter 1, real-time updates have no temporal consistency problems, so the real-time update approach has the best performance in temporal consistency. However its enormous cost limits its applicability as an optimum solution. In this section, we compare the real-time update approach and the proposed TDCM approach to measure the number of update transaction.

**Definition: Number of Update Transaction**

*All possible consistency updating transaction of data objects permute with different timestamps.*

We used the following parameters in this simulation for the purposes stated:

- The Number of Related Data Cubes (|N|): To decided how many related data cubes in a simulation run.
- Valid Interval Temporal (|VIT|): The threshold value specifies the temporally-based data consistency of data required by the DWS. The time interval of each two data objects with greater then VIT is considered out-of-date. In this simulation, we give VIT the same value of 0 for the worst case situation.

Considering the worst case, \( L = \{X_1, X_2, X_3, \ldots, X_n\} \) is a set of data objects and \( T_{\text{now}} \) is the current time. \( T = \{t_1, t_2, t_3, \ldots, t_n\} \) (\( t_n > t_{n-1} > t_{n-2} \ldots > t_1 \)) is a set of timestamps where the user browsing sequence will be followed by a sequence, such as:
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

X₁ → X₂ → X₃... → Xₙ. Our program simulates all possible situations to calculate the number of update transactions in the real-time and TDCM approaches.

To contrast the real-time update and TDCM approach, we use an easily compared and analyzed metric %Update Number (specific weight) to illustrate the results. The simulation results are shown in Figure 14:

Figure 14 shows that under the worst case situation, if data cube relationship is in a specified range (less than 7), our TDCM approach is better than the real-time update. Considering the other situations, including 1 to m relations or given a VIT threshold greater than 0, we expected the number of update transaction will be decreased. Figure 15 shows the simulation result under VIT=1 situation.

Because we use a system simulation to evaluate our effectiveness, we not only compare the number of inconsistencies in the Top-down and Bottom-up architectures, but also calculate the number of update transactions for real-time update and for our TDCM approach. We also found the point to reach temporal-based data consistency is on VIT threshold setting. A useful and suitable VIT can not only maintain temporal-based data consistency easily but also greatly reduce the update time cost.

CONCLUSION

In this research, we have defined temporal-based data consistency in a data warehouse system and established a TDCM to maintain data consis-
Using Active Rules to Maintain Data Consistency in Data Warehouse Systems

Table 2. Summary compare result

<table>
<thead>
<tr>
<th></th>
<th>DWS with TDCM</th>
<th>Traditional DWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Cost</td>
<td>Less</td>
<td>More</td>
</tr>
<tr>
<td>Data Quality</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Data Consistency</td>
<td>Temporally-based</td>
<td>Inconsistent</td>
</tr>
<tr>
<td>Availability</td>
<td>Available</td>
<td>Partially Available</td>
</tr>
<tr>
<td>Comparability</td>
<td>Comparable</td>
<td>Partially Comparable</td>
</tr>
</tbody>
</table>

At the present time our implementation is not sufficiently efficient to perform effectively on real scale problems with rapidly changing data and complex constraints between data. Finding ways of improving efficiency is therefore a major focus of our current work.

Our current results apply only to a single data warehouse situation. Although such situations are common in practice, future practical applications on the internet will involve access to multiple heterogeneous data warehouses and data sources exhibiting more complex consistency problems. This will also be an objective of our research in future.

REFERENCE


Hanson, E.N. (1996). The design and implementation of the ariel active database rule system. *IEEE Transaction on Knowledge and Data Engineering*, 8(1),157-172.


Using Active Rules to Maintain Data Consistency in Data Warehouse Systems


