Fine Grain Provenance Using Temporal Databases

Dieter Gawlick
Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065, USA
dieter.gawlick@oracle.com

Venkatesh Radhakrishnan
Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065, USA
venkatesh.radhakrishnan@oracle.com

Abstract

Database applications often require rigorous provenance; e.g., it is important to know who is responsible for a change and at what time a change was done. Additionally, it is important to know which program, rule, or model was executed, and which data was used as input. The standard solution is to include provenance into the application logic. The consequence is that such a program extends significantly – a factor of 3 to 5 is often cited – and most importantly provenance by applications is normally badly designed and hard to use. This paper shows that temporal databases can be used to automate the provisioning process and that they enable a application design based on three key concepts: Facts, knowledge, and information.

1. Introduction

The ever increasing amount and complexity of data combined with an ever increasing amount and complexity of knowledge challenges the cognitive systems of humans.

One way to deal with this issue is the use of three key concepts: **Facts**, **knowledge**, and **information** [2]. Facts represent what we observe, information is what we extract, and knowledge is used to do the extraction. Facts are quantitative and are often expressed as numbers and dimensions; e.g., the blood pressure is 150 bpm. There are potentially (too) many facts for the human cognitive system; since they are indiscriminate.

Information is qualitative and is expressed using classifications; e.g., the blood pressure is *critical*. We can limit information to what is important; i.e., the extraction leads to a classification with a (high) positive or negative value. The human cogitative system prefers to deal with qualitative information; indeed humans almost always tend to use and communicate on the qualitative level.

Information tends to be fuzzy, which seems to be OK in most cases. In some cases, however, it is important to understand how information was derived; i.e., which facts and which knowledge was used to derive it. Furthermore, it may be of interest to understand the source and quality of the facts as well as the source and quality of the knowledge. Obviously, fine grain provenance can be used to deal with this issue.

The use of the term provenance as used in this paper is in line with the use in [3] (VisTrail) but not in line with the use in [4] (LIVE).

Modern databases with strong temporal support provide the infra-structure for fine grain provenance as integral part of their functionality.

The remainder of the paper is structured as follows: section 2 describes a use case, section 3 discusses temporal database, section 4 shows how temporal databases can be used to provide fine grain provenance, and section 5 is discussing the interrelation between facts, knowledge, information, and provenance. Finally, section 6 discusses the importance of snapshot isolation for provenance.

2. The Use Case: Patient Care

EMRs (Electronic Medical Records) are designed to capture facts that are relevant to understand the health of a person. Ideally, facts are stored as they are captured, giving the medical personnel an unobstructed view of the current and any previous state of a patient. All facts captured have to be kept and must be accessible at any time.

Since the amount of facts is overwhelming, doctors tend to classify these facts to derive useful information, e.g., the blood pressure is classified as *critical* if it is above or below a certain threshold. Information may be

derived from a single fact or a set of facts; e.g., a high probability of an impending heart attack can be derived from a set of facts of the blood chemistry. Doing so, doctors identity the fact or set of facts with the highest importance and turn these facts into relevant information. Obviously, this transformation from facts into information requires significant domain knowledge.

Since information looses the precision of facts, doctors have sometimes to review the transformation by reviewing the facts. Therefore, for any system that helps doctors to derive information automatically, fine grain provenance is a must.

These principles outlined for the patient care application can be used in many other domains; examples are smart grid management, account management, and program trading,

3. Temporal Databases

Temporal databases support two aspects of time, transactional and valid time.
The support for transactional time is designed to keep, remember, and access all versions of records without any support of the programs managing the data. In a transaction temporal database old versions of a record as well as deleted records will be kept and are accessible through simple (SQL) extensions. Such an extended query clause can be used to find values of one or more records in the past – at a specific time t or values in a period – from time t_1 to t_2.
Transaction time history is provided in the Oracle database by the Total Recall option. It provides the ability to retain transactional history of rows in a table. This history of a table can be accessed in SQL by adding decorators to a query to access it as of a time in the past (AS OF t) or to get versions between ranges of times (VERSIONS (t_1, t_2)) [1].
Here are two examples for transaction temporal support:

```
SELECT * FROM emp
        AS OF TIMESTAMP
        (systimestamp – interval '1' year);

SELECT * FROM emp
        VERSIONS BETWEEN timestamp
        to_timestamp('23-JAN-11') and maxvalue;
```

In contrast, applications are involved in managing valid time; i.e., applications have to provide a time interval, during which records are considered to be valid, a NULL upper limit means that a record is valid until changed.
Here are two examples for valid time support:

```
UPDATE emp
        portion of valid_time
        between to_timetamp
        ('23-JAN-10')
        and to_timestamp ('23-OCT-10')
        where empno=7934;

SELECT * from emp
        as of timestamp to_timestamp
        ('23-FEB-11') as of
        valid_time
        to_timestamp('23-JAN-11);
```

In the context of this paper we focus on one specific use of valid time: the correction of data or knowledge. Facts and to a lesser degree knowledge (and consequently information) is subject to later correction. In reviewing how information was arrived one has to know which facts (and knowledge) were used when the information was derived.

4. Supporting Fine Grain Provenance with Temporal Databases

Temporal databases have all the elements to provide fine grain provenance without any additional support by applications. All versions of records are kept; each version of a record includes the transaction time which is associated with its creation. Since each transaction has its unique time, each change can be uniquely associated to transactions.
For each transaction in turn, the database system knows the client along with any security information such as sign in, log off time, and any credential. Last but not least, there is a log of any (SQL) access with the time when the access was, unless snap shot isolation is used (see section 6).
Versions queries as shown in section 3 can be used to get the versions information for each row. In addition, a function dbms_flashback_archive.get_sys_context can be used to obtain the security information about each transaction – this is of course specific to the Oracle database.

```
SELECT
    dbms_flashback_archive.get_sys_context
    (:xid, 'USERENV', 'CLIENT_INFO')
    FROM dual;
```

5. Managing Facts, Knowledge, and Information

The introduction stated the use of three key concepts: Facts, knowledge, and information. In this section the implementation of these concepts will be discussed.

Facts – Facts can be represented as structured, semi-structured, and un-structured data. Therefore, the use of SQL and XML has to be complemented by additional data types, such as spatial, textual, and multi-media. Medical applications require DICOM. The underlying system has to be able to support all these technologies and provide temporal support.

Knowledge – Knowledge can be captured in the form of (continuous) queries, rules, models, and processes. The underlying system has to be able to manage knowledge in any of these forms. If there is any change, the underlying system has to be able to capture the time of the change and provide temporal support.

Knowledge is used to derive information in two modes: Real time and retroactive mode. In real time mode existing knowledge is applied to incoming facts as soon as they become available - along with all the relevant facts that already exist. In retroactive mode knowledge is applied to existing facts; i.e., the retroactive mode is used to review facts after the knowledge has changed or new knowledge becomes available. In retroactive mode, the system has to simulate the real time mode; i.e., it has to go through the existing facts in (transaction) temporal order.

Knowledge is not always absolute. One doctor may consider a value as being serious while another doctor may consider the same value as being critical. Consequently, knowledge has to be customizable. This customization – and any update thereof - has to be reflected in the database and will also be reflected in the provenance support.

Information – Information is derived by applying knowledge to facts. In its simplest form, information is represented by a predicate and a classification value. In the medical example, there are a few values associated with each predicate, e.g., normal, guarded, serious and critical [2]. Additionally, there is typically an order in the classification values – from good to bad – and it is therefore possible to look at changes. This allows derivative expressions such as improving or deteriorating assuming the information is kept in a temporal database. Looking at the length of time intervals one can look at the speed of the change.

A more complex classification involves several facts; this is typically called a diagnosis. A good example is the use of the blood chemistry to predict the probability of a cardiac arrest several hours ahead of the event. In this case a model is used to calculate the probability; the model also associates a classification value (normal to critical), and creates an information element if the classification value is 'bad enough.'

Frequently, the same set of classification values is associated with many information predicates. This provides a much desired uniformity thus avoiding the complication of dealing with a variety of dimensions and ranges of values as is required for facts. This allows very generic queries and rules such as: which – if any – vitals or blood values of a patient are critical or even more generic: what is important to know about a specific patient.

Extracting information from facts – Whenever there are new facts the knowledge base has to be analyzed to find out if these new facts lead to the real time creation, modification, or deletion of information. Each fact has a time when it became available (visible) – this is the transaction temporal time. All facts and knowledge that are available at that time will be used to derive new information. The information will be created in the context of a transaction; therefore a transaction time is associated with that information. Therefore, each information element has two times, one time representing the state of the facts and the knowledge, the other time represents the time when the information element became available (visible). Since information elements are temporal they will go through changes.

Understanding the facts and knowledge behind information – Whenever information is investigated, the system will be able to show when the information was available, which knowledge with which customization was used, and which facts were used. Furthermore, it will be able to show when the facts were created or updated and by whom as well as when the knowledge that was used was created or updated and by whom. I.e., the support for provenance is provided assuming the information extraction is done in the context of a temporal database.

6. Consistency

Consistency is a key aspect of database technology. In non-temporal databases consistency is normally supported by 2-phase locking. 2-phase locking dictates that no lock can be released until all locks are acquired and that locks prevent changes to any acquired record by another transaction.

Unfortunately, 2–phase locking reduces parallelism to a point that it is not practical for many applications.

An alternative is snapshot isolation. With the support of snapshot isolation, the database is able to present data at a point in time while allowing changes to the same data. This technology is the preferred technology for reports and for deriving information from facts.

The Oracle database supports snapshot isolation.

7. Conclusion

We have described how temporal databases can be used to implement fine grain provenance. Commercial implementations such as Total Recall [1] can be used as a basis. Total Recall maintains the transaction history and provides the ability to query seamlessly and consistent the history of facts, knowledge, and information.

8. References

[1] Oracle Total Recall White Paper: http://www.oracle.com/technetwork/database/focus-areas/storage/total-recall-whitepaper-171749.pdf

[2] D. Gawlick, A. Ghoneimy, Z. Liu: How to Build a Modern Patient Care Application. International Conference on Health Informatics (2011) 427-432.

[3] VisTrails: http//w.sci.utah.edu/software/1-nsf/40-vistrails

[4] A. Das Sarma, M. Theobald, and J. Widom. LIVE: A Lineage-Supported Versioned DBMS. Proceedings of the 22nd International Conference on Scientific and Statistical Database Management, Heidelberg, Germany, June 2010.