Building of a Secure Data Warehouse by Enhancing the ETL Processes for Data Leakage

Preeti Patil
Associate Professor
Computer Department
MPSTME, NMIMS

Nitin Chavan
Assistant Professor
IT Department
MPSTME, NMIMS

Srikantha Rao
Ex-TIFR Scientist,
Director, TIMSCDR
Mumbai, India

S B Patil
Associate Professor
IT Department
MPSTME, NMIMS

1. ABSTRACT
Now days, many organizations outsource some of their important data to some outside trusted agents. This outsourced data coming from the data warehouse of the organization through ETL process may have some sensitive information. Though the agents are trusted one, the sensitive data may be given by them to some outside untrustworthy third party. This is known as data leakage. Such a data leakage may be the result of guilty agents or may be the third party has guessed or intruded the data by some other means. To find out whether the data has been given by agent or by some other means, the third party has guessed it, is very important issue. In this project, some data allocation techniques to trusted agents are presented that will improve the probability of identifying the guilty agents. The methods used do not modified the data. Also to find out the guilty agent is another sensitive topic. This paper proposes a system to find out the leakage of data and also the guilty agents. Few "realistic but fake" data can be injected to find out guilty agents.

General Terms
Experimentation, Legal Aspects, Reliability, Security, Verification

Keywords
Data Ware house, ETL, Perturbation, guilty agents, fake objects.

2. INTRODUCTION
Many organizations have their data warehouse. Through the ETL process, the data is extracted. There is a chance of intrusion of the data in to the transformation phase. Once the data gets loaded, one can use that data. Now days, the organizations have to outsource their sensitive data to some outside trusted third party. Such organizations are called as data distributors [1]. For example, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. The trusted third parties are called the agents. The goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data [1].

Perturbation is a technique in which the original data is modified and made somewhat less sensitive so that it will pretend itself as un-useful data to some intruder. Fingerprinting [4] and Watermarking [6] are such perturbation techniques. Instead of these techniques, simple methods like changing exact values by ranges can be done under perturbation. However, in some cases it is important not to alter the original distributor's data. For example, the values of any scientific research can not be modified by even a single margin. In such situations, one can not use perturbation techniques, so he has to go by the method in which data alteration is not required. This proposed system provides a solution under such situations.

The goal is to detect whether the data has been leaked by agent or has been intruded. Also it looks to find out that if data is leaked, which agent is more likely to be guilty. It proposes some data allocation strategies which increase the chance of finding the guilty agent. Also this method can use some fake objects for detection purpose.

3. LITERATURE SURVEY
Panagiotis Papadimitriou; Hector Garcia-Molina [1] has proposed this leakage detection system which can enable us to detect the guilty leaker without changing the integrity of the original data. They have used some allocation strategies to help find out guilty person.

Chuanxian Jiang; Xiaowei Chen; Zhi Li[2] has given the feasibility of embedded watermarks on relational databases in Discrete Wavelet transform domain. They have proposed that by By employing linear correlation detecting method, a watermark can be embedded into relational database in DWT domain.

Marecki; Mudhakar Srivatsa; Pradeep Varakantham[3] has formulated data leakage prevention problem as a Partially Observable Markov Decision Processes. They show how to embed digital watermarking in to the information and derive optimal information sharing strategies for the sender and optimal information leakage strategies for rational-malicious recipient.

Yingjiu Li, Member; Vipin Swarup; Sushil Jajodia[4] has presented a for fingerprinting relational data by extending watermarking scheme. The primary new capability provided by this technique is that, under reasonable assumptions, it can embed and detect arbitrary bit string marks in relation.

Mohamed Shehab; Elisa Bertino; Arif Ghafoor[5] has formulated the watermarking of relational databases as a constrained optimization problem and has discussed efficient techniques to handle the constraint. This watermarking technique is resilient to watermark synchronization error because it uses a partitioning approach.

4. PROBLEM WITH EXISTING SYSTEM
Most of the systems described above use either fingerprinting or watermarking techniques as a tool to identify unauthorized copy of the sensitive information. The main problem with these

techniques is that they induce some marking bits into the data to provide the security. This induction of marking bits modifies the original data by some extent. In some application, where the constraints on integrity of data are not so strict, these techniques can be effectively used. But there are certain applications where the integrity of data is of main concern and can not be hampered even by a little extent. For example, the situations where scientific experiments are to be distributed to some outside agents for further processing, one can not induce any marking bits as it will affect the integrity of data which may then affect the further processing of the experiment or some patient records are to be given to some outside party for giving the treatment. Such records are also sensitive to the changes and can not be modified. In such situations, above given techniques can not be used. Thus there remains no way to prove the ownership of data. In such situations, this method can be used to find the guilty agents. ETLExtract, transform and load (ETL) is a process in database usage and especially in data warehousing that involves: - Extracting data from outside sources- Transforming it to fit operational needs (which can include quality levels)- Loading it into the end target (database or data warehouse)

4.1 Extraction

Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization/format. Common data source formats are relational databases and flat files, but may include non-relational database structures such as Information Management System (IMS) or other data structures such as Virtual Storage Access Method (VSAM) or Indexed Sequential Access Method (ISAM), or even fetching from outside sources such as through web spidering or screen-scraping. The streaming of the extracted data source and load on-the-fly to the destination database is another way of performing ETL when no intermediate data storage is required. In general, the goal of the extraction phase is to convert the data into a single format which is appropriate for transformation processing. An intrinsic part of the extraction involves the parsing of extracted data, resulting in a check if the data meets an expected pattern or structure. If not, the data may be rejected entirely or in part [7].

4.2 Transform

The transform stage applies a series of rules or functions to the extracted data from the source to derive the data for loading into the end target. Some data sources will require very little or even no manipulation of data. In other cases, one or more of the following transformation types may be required to meet the business and technical needs of the target database:

Selecting only certain columns to load (or selecting null columns not to load). For example, if the source data has three columns (also called attributes), for example roll_no, age, and salary, then the extraction may take only roll_no and salary. Similarly, the extraction mechanism may ignore all those records where salary is not present (salary = null).

Translating coded values (e.g., if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female), this calls for automated data cleansing; no manual cleansing occurs during ETL

Encoding free-form values (e.g., mapping "Male" to "1")Deriving a new calculated value (e.g., sale_amount = qty * unit_price)

Sorting

Joining data from multiple sources (e.g., lookup, merge)

Aggregation (for example, rollup — summarizing multiple rows of data — total sales for each store, and for each region, etc.)

Generating surrogate-key values

Transposing or pivoting (turning multiple columns into multiple rows or vice versa)

Splitting a column into multiple columns (e.g., putting a comma-separated list specified as a string in one column as individual values in different columns)

Disaggregation of repeating columns into a separate detail table (e.g., moving a series of addresses in one record into single addresses in a set of records in a linked address table)

Lookup and validate the relevant data from tables or referential files for slowly changing dimensions.

Applying any form of simple or complex data validation. If validation fails, it may result in a full, partial or no rejection of the data, and thus none, some or all the data is handed over to the next step, depending on the rule design and exception handling. Many of the above transformations may result in exceptions, for example, when a code translation parses an unknown code in the extracted data [7].

Load

The load phase loads the data into the end target, usually the data warehouse (DW). Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information, frequently updating extract data is done on daily, weekly or monthly basis. Other DW (or even other parts of the same DW) may add new data in a historicized form, for example, hourly. To understand this, consider a DW that is required to maintain sales records of the last year. Then, the DW will overwrite any data that is older than a year with newer data. However, the entry of data for any one year window will be made in a historicized manner. The timing and scope to replace or append are strategic design choices dependent on the time available and the business needs. More complex systems can maintain a history and audit trail of all changes to the data loaded in the DW.

As the load phase interacts with a database, the constraints defined in the database schema — as well as in triggers activated upon data load — apply (for example, uniqueness, referential integrity, mandatory fields), which also contribute to the overall data quality performance of the ETL process.

For example, a financial institution might have information on a customer in several departments and each department might have that customer's information listed in a different way. The membership department might list the customer by name, whereas the accounting department might list the customer by number. ETL can bundle all this data and consolidate it into a uniform presentation, such as for storing in a database or data warehouse.Another way that companies use ETL is to move information to another application permanently. For instance, the new application might use another database vendor and most likely a very different database schema. ETL can be used to transform the data into a format suitable for the new application to use.

An example of this would be an Expense and Cost Recovery System (ECRS) such as used by accountancies, consultancies and lawyers. The data usually ends up in the time and billing system, although some businesses may also utilize the raw data for employee productivity reports to Human Resources (personnel dept.) or equipment usage reports to Facilities Management.

4.3 Real-life ETL cycle

It consists of the following execution steps:

1. Cycle initiation
2. Build reference data
3. Extract (from sources)
4. Validate
5. Transform (clean, apply business rules, check for data integrity, create aggregates or disaggregates)
6. Stage (load into staging tables, if used)
7. Audit reports (for example, on compliance with business rules. Also, in case of failure, helps to diagnose/repair)
8. Publish (to target tables)
9. Archive
10. Clean up

4.4 Legal Electronic Data Exchange Standard

The Legal Electronic Data Exchange Standard is a set of file format specifications intended to standardize bill/invoice data transmitted electronically ("e-billed") from a law firm to a corporate client. It is abbreviated LEDES and is usually pronounced as "leeds".

LEDES was developed by the LEDES Oversight Committee (LOC), which was formed by the PricewaterhouseCoopers Law Firm and Law Department Services Group. Members of the committee include law firms, corporate legal departments, electronic billing vendors and time and billing software vendors. The LOC was incorporated as a California non-profit mutual benefit corporation in 2000.

The file format has several variations:

- **LEDES 1998**, the first "LEDES" format, created in 1998, but no longer in use.
- **LEDES 1998B**, a pipe-delimited plain text file. The standard was adopted in 1998, and it is by far the more commonly used LEDES format. It lacks some flexibility, having a rigid structure. Another disadvantage of LEDES 1998B is that invoice-level data is repeated on every line item even though it is only needed once, as it does not vary per line. Many clients attempt to impose nonstandard customizations, thus defeating the purpose of having a standard. Nonetheless, law firms prefer it for its simplicity and familiarity.
- **LEDES 2000**, adopted in 2000, is an XML format. Adoption of this newer standard has been slow. One advantage of LEDES 2000 is that although the structure is very well defined, the specification defines "extend" segments, allowing the insertion of client-specific fields without breaking the format or violating the standard.
- **LEDES 1998B-INTL** (international), a pipe-delimited plain text file, based on the LEDES 1998B standard. This format was designed to accommodate legal bills generated outside of the United States. It includes all of the fields in the LEDES 1998B format, plus additional ones. The format was proposed in 2004 by the Legal IT Innovators Group (LITIG). The LEDES Oversight Committee (LOC) ratified the format in 2006.
- **XML E-Billing version 2**, ratified in 2006 but yet to see adoption, is intended to improve upon LEDES 2000.
- **XML E-Billing version 2.1** improves upon version 2.
- **LEDES Budget Standard** was ratified in 2006. This XML format is intended to facilitate the exchange of budget data between law firms and clients.
- **Timekeeper Attribute Standard** is a proposed XML format intended to be used to transmit timekeeper and rate data to from law firms to clients.

5. PROPOSED SYSTEM

5.1 Problem Definition

Suppose there is one organization which needs to outsource some of their work to some other outside party or organization. Such outside organizations are called as Agents. Now this outsourced data is found at some unauthorized place for example, the data is displayed on some web site or as a result of legal discovery process, or with some unauthorized person. Now there exits two possibilities:

1. The data has been leaked by one or some of the trusted agents.

2. The unauthorized person might have intruded the data using some unfair means. The goal is to maximize the chances of detecting the guilty agent, if he has leaked the data.

5.2 Notations

Suppose O= $\{o_1,o_2,..,o_n\}$ be the objects to be distributed to U= $\{u_1,u_2,\ldots,u_n\}$ agents. There exist two types of requests that can come from U. These are:

1. Sample Request- Any object can be provided to agent.

2. Explicit Request- Objects are to be provided based on some condition.

5.3 Modules

5.3.1 Database

A database has to be maintained to keep the record of registration of the agents that are given the data for processing. It is also going to maintain the sensitive data that has been given to agents.

5.3.2 Agent Maintenance

This module includes-

- Registration: Here details of agents are registered and it collects the information about them like what are the sensitive data they want.
- History: Here the agent history is maintained like what all the details are given by distributor previously It maintains entire details of the agent. To detect the guilty agent, it checks the history.

5.3.3 Detecting guilty Agents

Suppose that after giving objects to agents the distributor discovers that a set S has leaked. This means that some third party called the target has been caught in possession of S. For example, this target may be displaying S on its web site, or perhaps as part of a legal discovery process, the target turned over S to the distributor. Since the agents U1…..Un have some of the data, it is reasonable to suspect them leaking the data. However, the agents can argue that they are innocent, and that the S data was obtained by the target through other means. For example, say one of the objects in S represents a customer X. Perhaps X is also a customer of some other company, and that company provided the data to the *target*. Or perhaps X can be reconstructed from various publicly available sources on the web. The goal is to estimate the likelihood that the leaked data came from the agents as opposed to other sources. Intuitively, the more data in S, the harder it is for the agents to argue they did not leak anything. Similarly, the "rarer" the objects, the harder it is to argue that the target obtained them through other means. Not only to estimate the likelihood the agents leaked data, it also helps to find out if one of them in particular was more likely to be the leaker. For instance, if one of the S objects was only given to agent U1, while the other objects were given to all agents, suspect U1 more. It says an agent Ui is guilty and if it contributes one or more objects to the target.

5.3.4 Data Allocation

The main focus of this method is the data allocation problem: how can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Figure 6, there are four instances of this problem are addressed, depending on the type of data requests made by agents and whether "fake objects" are allowed. The two types of requests that are to be handled were defined in Section 4.2: sample and explicit. Fake objects are objects generated by the distributor that are not in set T. The objects are designed to look like real objects, and are distributed to agents together with the T objects, in order to increase the chances of detecting agents that leak data. Figure 1 represents four problem instances with the names EF, EF` , SF and SF` , where E stands for explicit requests, S for sample requests, F for the use of fake objects, and F` for the case where fake objects are not allowed. Note that for simplicity it is assumed that in the E problem instances, all agents make explicit requests, while in the S instances, all agents make sample requests. The results can be extended to handle mixed cases, with some explicit and some sample requests.

Here is a small example provided to illustrate how mixed requests can be handled. Assume that there are two agents with requests R1 = EXPLICIT(T, cond1) and R2 = SAMPLE(T′, 1) where T′ = EXPLICIT(T, cond2). Further, say cond1 is "state=CA" (objects have a state field). If agent U2 has the same condition cond2 = cond1, we can create an equivalent problem with sample data requests on set T_. That is, our problem will be how to distribute the CA objects to two agents, with R1 = SAMPLE(T′, |T′|) and R2 = SAMPLE(T′, 1). If instead U2 uses condition "state=NY," we can solve two different problems for sets T′ and T − T′. In each problem there will be only one agent. Finally, if the conditions partially overlap, R1 ∩ T′≠ Ø, but R1 ≠ T′ one can solve three different problems for sets R1 − T′, R1 ∩ T′ and T′ − R1.

6.3.5 Addition of fake object

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. The idea of perturbing data to detect leakage is not new. However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In this case, perturbing the set of distributor objects by adding fake elements is done.

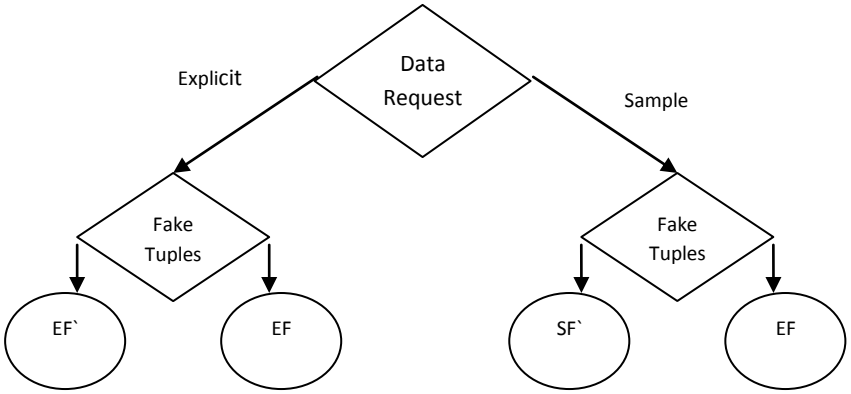


Fig 1: Leakage Problem Instances

In some applications, fake objects may cause fewer problems than perturbing real objects. For example, say the distributed data objects are medical records and the agents are hospitals. In this case, even small modifications to the records of actual patients may be undesirable.

However, the addition of some fake medical records may be acceptable, since no patient matches these records, and hence no one will ever be treated based on fake records. A trace file is maintained to identify the guilty agent. Trace file are a type of fake objects that help to identify improper use of data.

6. EXPERIMENTATIONS

We have considered the Department of IT of MPSTME for experimentations. We have considered some fictitious medical records for experimentations. We chose four people (we will call them agents) of the department to distribute the data to. We considered that Mr. X has HIV positive. To increase the chances of detection, we add different stages of disease as fake objects.

Now we provide this information to the agents.

Agent 1 is given the information that Mr. X is having HIV Positive and is in stage 1.

Agent 2 is given the information that Mr. X is having HIV Positive and is in stage 2.

Agent 3 is given the information that Mr. X is having HIV Positive and is in stage 3.

Agent 4 is given the information that Mr. X is having HIV Positive and is in stage 4.

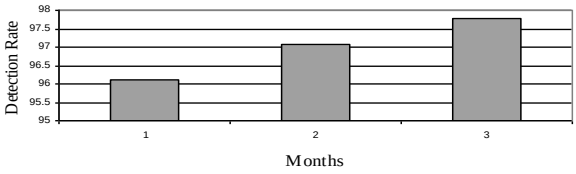
Now these agents were told to give the data to some outside party who should not receive it. The outside parties were told to display that information somewhere on internet or on their account. Based on the fake object, we tried to find out the guilty agent who has leaked information using above explained model. We found out following values.

Table1. Detection Rate of guilty agents

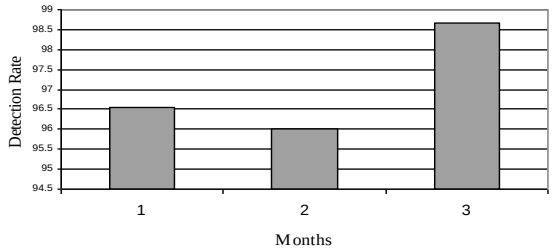
Agent	January	February	March
Agent1	96.11% detection	97.08% detection	97.78% detection
Agent2	96.56% detection	96.02% detection	98.67% detection
Agent3	97.09% detection	98.41% detection	96.8% detection
Agent4	98.11% detection	96.31% detection	97.43% detection

From the table 1, it can be seen that the detection rate is not full. This happens because the data that may be leaked may not be displayed as it is. Suppose that agent 1 is given the data that Mr. X is having HIV Positive at stage 1, wherein stage 1 is the fake information. Then Agent 1 has leaked that information to outside third party. But that third person has displayed that Mr. X is having HIV Positive only. As stage of the disease is not displayed, one can not surely say that Agent 1 has leaked the information as the information that Mr. X is HIV Positive is with other agents also. This is the drawback of this methodology.

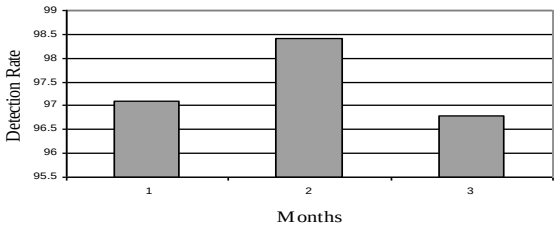
Figure 2 shows the experimentation results of the detection of the data leakage to un-trusted third party. This is based on the activities carried out from January 2011 to March 2011 by four agents. The comparison graphs shows that the data leakage detection rate using this method is close to 100%.



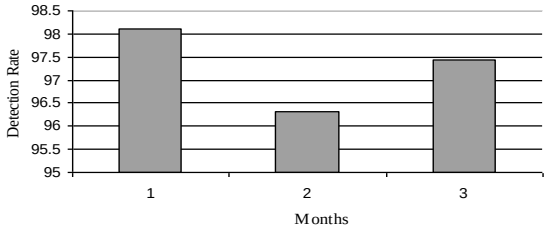
Agent 1



Agent 2



Agent 3



Agent 4

Fig 2. Leakage Detection Rate of Agents

7. CONCLUSION

Watermarking is indeed a very good method of ownership right protection. But watermark when inserted modifies the original data to some extent. It is same with each and every watermarking technique with some differences. Thus the watermarking scheme will be of no use if the data to be distributed is very sensitive and can not be modified at all.

The proposed method for data leakage can be very useful in the situations where the data can not be modified even by a very little value. It is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means.

The data distribution strategies can improve the distributor's chances of identifying a leaker. In some cases "realistic but fake" data records can be injected to improve the chances of detecting leakage and identifying the guilty party.

8. FUTURE SCOPE

In the above section we saw the drawback of the system. This is because the un-trusted third party may display some part of the leaked information and not the whole information. So it will be difficult to detect the guilty agent. Thus into this method, as a future work, we can add some data distribution techniques that will help detect the guilty person. Also we can develop an automated module that will generate the fake objects related to the data that is to be distributed. These fake objects will look important to agent as well as third party.

9. REFERENCES

[1] Panagiotis Papadimitriou, Hector Garcia-Molina. 2010. Data Leakage Detection, IEEE Transaction On Knowledge And Data Engineering.

[2] Chuanxian Jiang, Xiaowei Chen, Zhi Li. 2009. Watermarking Relational Databases for Ownership Protection Based on DWT. Fifth International Conference on Information Assurance and Security.

[3] Janusz Marecki, Mudhakar Srivatsa, Pradeep Varakantham. 2007. A Decision Theoretic Approach to Data Leakage Prevention. IEEE International Conference on Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust.

[4] Yingjiu Li, Vipin Swarup, Sushil Jajodia. 2005. Fingerprinting Relational Databases: Schemes and Specialties. IEEE Transaction on Dependable And Secure Computing.

[5] Mohamed Shehab; Elisa Bertino, Arif Ghafoor. 2008. Watermarking Relational Databases Using Optimization-Based Techniques. IEEE Transactions on Knowledge and Data Engineering.

[6] Yuer Wang, Zhongjie Zhu, Feng Liang, Gangyi Jiang. 2008. Watermarking Relational Data Based on Adaptive Mechanism. Proceedings of IEEE International Conference on Information and Automation.

[7] Simitsis, A.; Vassiliadis, P.; Sellis, T. "Optimizing ETL processes in data warehouses", Data Engineering 21st International Conference, 2005. ICDE 2005. Proceedings. Publication Year: 2005, Page(s): 564 – 575.