O. M. Adedayo and M. S. Olivier. "Schema Reconstruction in Database Forensics". In: Advances in Digital Forensics X. Ed. by G. Peterson and S. Shenoi. Springer, 2014, pp. 101–116. Preprint. Authoritative version available at https://dx.doi.org/10.1007/978-3-662-44952-3_8

Chapter 1

# SCHEMA RECONSTRUCTION IN DATABASE FORENSICS

Oluwasola Mary Adedayo and Martin S Olivier

**Abstract** Although an increasing amount of research has been done in the database forensics field over the past few years, there are still various aspects of database forensics that need to be considered. One of the current challenges facing database forensics analysis deals with the fact that results obtained from a database during an analysis may actually be different from the raw data contained due to changes that have been made to the metadata. In this paper, we describe the different categories of changes that can be made to a database schema by an attacker and show that metadata changes can truly affect results from queries by using typical examples. Techniques that can be used to reconstruct the original schema of the relations in the database are also described.

**Keywords:** Database forensics, database reconstruction, relational algebra, inverse relational algebra, forensic analysis.

## 1. Introduction

Databases play an important role in many organizations and are a core component in many computing systems. However, the use of databases to store critical and sensitive information in various organizations has led to an increase in the rate at which they are exploited to facilitate computer crimes [5]. As such, database systems have become of interest during many forensic investigations as useful information relevant to an investigation can usually be found in it. Database forensics as a branch of digital forensics [15, 11] deals with the identification, preservation, analysis and presentation of evidence from databases [8]. Over the past few years, researchers [14, 1] have emphasized the need to incorporate database forensics analysis as part of traditional digital forensics due to

the amount of information that can often be retrieved from databases during an investigation. Although there is still much to be done in the formalization of different aspects of the database forensics process, the amount of research in database forensics is gradually increasing.

Considering the various work that has been done on database forensics, there are a number of ideas that are common to most of the research. The use of database logs as a source of information for forensic analysis seem to be one that stands out in many of the research works [12, 9, 10, 7, 5]. Database logs can be explored in various ways for the reconstruction of both queries executed on a database and the values in the database at an earlier time. In addition, databases have been recognized as a tool that can be used for the forensic analysis of the database itself since it provides the ability to execute complex queries for information retrieval from the database [14]. An important aspect of database forensics deals with the ability to revert operations performed on a database so that the information that exist on the database at an earlier time can be reconstructed [7, 5, 4]. An aspect of database forensics that has not yet been considered deals with the reconstruction of the database schema or the metadata. Although the schema is mentioned in much of the work that has been done [1, 7, 5, 12], there is currently no developed method for reconstructing the schema of a database.

In our earlier work [7, 5], we have presented an algorithm that can be used for the reconstruction of the data in a database at an earlier time. Various techniques to ensure the completeness of reconstructed information have also been considered [4]. This paper builds on the previous work by describing techniques for reconstructing the database schema. The paper shows how the schema of a relation that has been deleted or lost, can be reconstructed based on the information in the log file. Examples are given to illustrate the various techniques described.

## 2. Database Management Systems

One of the fundamental characteristics of a database system is that it does not only contain the data stored in it but also contains a complete definition and description of the database and the data contained. This description is known as the *metadata* or *database schema* and is stored in the system *catalog*. The metadata contains details about the structure, the type and the format of each file and data item, as well as the relationship between the data and any other constraints [3, 13]. The metadata is used by the DBMS to provide information to database users or application programs that need certain information about the database structure. Since the DBMS software is not written for one spe-

cific database application, the metadata provides the information needed by a DBMS in order to understand the structure, type and format of files and data that it will access.

However, since it is a known fact that the result obtained from database query is a function of both the metadata and the data stored in the database, this characteristic can be exploited to facilitate crimes. An example is where a criminal creates a view (stored as metadata) that gives him an easy access to some data. Although the raw data may still exist, it may not be retrievable once the metadata is deleted [14].

## 3. Dimensions of Database Forensics

In a recent work [6], we discussed the three dimensions involved in the database forensics problem space. Although database forensics research seems to be divergent, the dimensions involved in database forensics deals with the investigation of Modified databases, Damaged databases, and Compromised databases.

A modified database is a database that has undergone changes due to normal business processes since an event of interest occurred. This category of databases is often of interest when a database is not directly involved in the crime being investigated, but is used to store information that may assist in solving the crime [7, 5, 6]. In contrast, a damaged database is a database in which the data contained in the database or other data files may have been modified, deleted or copied from their original locations into other places. These databases may or may no longer be operational depending on the extent of the damage done.

A compromised database is a database where some of the metadata or some software of the database management system (DBMS) have been modified by an attacker even though the database is still operational. Olivier [14] and Beyers et al. [1] pointed out that although a database itself may be the best tool for collecting data for forensic analysis of the database, the integrity of the results obtained cannot be guaranteed since the database might have been coerced into giving false information. Litchfield [12] also identified the problem while discussing steps to perform in a live response to attack on an Oracle database. The major part of the investigation of a compromised database deals with how to get a correct version of the metadata.

This paper gives an insight into this problem by exposing various ways in which a database can be compromised by changing the metadata. We address the problem by describing techniques that can be used for the reconstructing a database schema. The schema shows the design decisions about tables and their structures and as such can be used to get a

better understanding of the results obtained from database queries. The focus of this paper is to show typical examples of database compromise and also describe how the schema can be reconstructed by considering operations previously performed on the database. In an earlier work [7], we have presented an algorithm for the reconstruction of the data in a database. This paper considers the application of some of the previous described techniques for data reconstruction to the reconstruction of the database schema.

## 4. Concepts in Database Reconstruction

This section provides a brief overview of concepts used in the paper. It provides information about the relational model of database management systems, the relational algebra log and the inverse operators of the relational algebra.

**Relational Algebra and Relational Algebra Log** The relational model was developed by Codd [2] and works on the relational theory of mathematics. The model is composed of only one type of compound data known as a relation. Given a set of domains $D_1, D_2 \ldots D_n$ over which attributes $A = \{A_1, A_2, \ldots A_n\}$ are defined respectively, a relation $R$ (or $R(A)$) is a subset of the Cartesian product of the domains [2].

The relational algebra consists of basic operators used to manipulate relations and a relational assignment operator ($\leftarrow$). The basic operators transform either one or two relations into a new relation. The basic relational operators as defined by Codd [2] and the corresponding notation often used are shown in table 1 together with the inverses (explained below) of the operations. The notation, $R, S$ and $T$ used in table 1 are relations, while $A, B$ and $C$ are attributes of these relations. The notation, $p(attributes)$ is a logical predicate on one or more attributes representing a condition that must be satisfied by a row before the specified operation can be performed on it.

As introduced in our earlier work [7, 5], a relational algebra log (RA log) is a log of queries expressed as operations involving relational algebra operators instead of the traditional SQL notation and presents several advantages over the traditional SQL notation [7].

**Inverse Relational Algebra** The goal of the inverse operators of the relational algebra defined in our earlier work [7, 5] is to find the value of an attribute $A$ (for a tuple in relation $R$) at a specific time by finding the inverse of the most recent query performed on the current relation and sequentially retracing the queries backwards until the desired time is reached. Depending on whether or not the inverse usually generated

*Table 1.* Operators of the Relational Algebra and Their Inverses

| **Operators** | **Notation** | **Inverse Operator** |
|---|---|---|
| Cartesian product ($\times$) | $T \leftarrow R(A) \times S(B)$ | $\times^{-1}(T) = (R, S)$ where $R = \pi_A(T)$ and $S = \pi_B(T)$ |
| Union ($\cup$) | $T \leftarrow R \cup S$ | $\cup^{-1}(T) = (R^*, S)$ where $R^* = T - S$ provided that $S$ is known and vice versa |
| Intersection ($\cap$) | $T \leftarrow R \cap S$ | $\cap^{-1}(T) = (R^*, S^*)$ where $R^* = S^* = T$ |
| Difference ($-$) | $T \leftarrow R - S$ | $-^{-1}(T) = R^* = T$. If $R$ is known, then $S^* = R - T$ |
| Join ($\bowtie$) | $T \leftarrow R(A) \bowtie_{p(A,B)} S(B)$ | $\bowtie^{-1}(T) = (R^*, S^*)$ where $R^* = \bowtie_A (T)$ and $S^* = \pi_B(T)$ |
| Projection ($\pi$) | $T \leftarrow \pi_{A_1,A_2,A_3}(R)$ | $\pi^{-1}(T) = S^* = T$ |
| Selection ($\sigma$) | $T \leftarrow \sigma_{p(A)}(R)$ | $\sigma^{-1}_{p(A)}(T) = S^* = T$ |
| Division ($/$) | $T \leftarrow R[A, B/C]S$ | $/^{-1}(T) = (R^*, S^*)$ where $R^* = RM$ and $RM$ is the remainder of the division |

from an inverse operator is complete (that is, without missing information) or not, we categorize the inverse operators of relational algebra into two categories: a *complete inverse* or a *partial inverse*. However, regardless of this classification, there are often instances where a complete inverse can be found for operators categorized as a partial inverse operator. Given a relation $R$ generated from a inverse operation, the notation $R^*$ is used to denote that the inverse generated is incomplete. Table 1 gives a summary of the inverse operators of the relational algebra. For interested readers, a more detailed explanation of the inverses and when complete inverses exist can be found in previous work [7, 5].

## 5. Compromising a Database Schema

In this section, we describe some of the various ways in which a database schema can be compromised and illustrate the fact that this causes a database to give false answers to queries. The study was done using Postgres 9.2 running on a Windows 8 operating system.

### 5.1 Categories of Compromise

There are several activities that can be carried out on a database schema to compromise the database. Usually an attacker may compromise a database with the intention of damaging the database so that it is not useful for the intended purpose or users. An attacker may also compromise a database in an attempt to hide some data (stored legiti-

```
update pg_attribute set attnum = '5' where attrelid = '16432'
   and attname = 'purchasePrice';
update pg_attribute set attnum = '2' where attrelid = '16432'
   and attname = 'sellingPrice';
update pg_attribute set attnum = '3' where attrelid = '16432'
   and attname = 'purchasePrice';
```

*Figure 1.* Modifying Schema to Swap two Column Names

mately by a user or by the attacker) or to cover his tracks. In general, we can categorize the activities that can carried out by an attacker to achieve his objectives into the following groups:

- Localized changes to data,
- Changes to blocks of data, and
- Changes to links between blocks of data.

**Localized Changes to Data** Within a database schema, an attacker may make changes to specific columns of a relation. This may be done to hide the information contained in such columns or make the data unavailable to the database user. In practice, there are several ways in which this can be accomplished. A compromise such as simply swapping two columns can have severe effects on the operation of a database [14]. A typical example is where the "Purchase price" and the "Selling price" of a store's database is swapped by tampering with the schema. This obviously causes the store to sell their goods at a loss. Figure 1 shows a code segment that can be used to compromise a database by swapping the column names in the schema. This swap causes a `select` statement on one of the columns to return values from the second column.

Another possible way of manipulating the database schema by changing specific values in the schema in order to hide information is to change the identification of the attribute (within the schema) to something unknown to the schema. Although the raw data may still be present on the database, it becomes impossible to query the database for that information. It is possible that information initially thought to have been deleted or non-existent, turns out to be irretrievable because it now appears hidden from the DBMS through a modification of the schema. Figure 2 shows the command that can be used to hide the attribute `purchasePrice` from the authorized users by changing the identification used by the DBMS to access the column into a different one.

Details such as the data type of the attributes of a relation can also be modified by changing specific values in the schema. This information

```
update pg_attribute set attnum = '5' where attrelid = '24587'
    and attname = 'purchasePrice';
```

*Figure 2.* Modifying Schema to Hide a Column

```
update pg_attribute set atttypid = '18' where attrelid = '16432' and
  attname = 'firstName'
```

*Figure 3.* Modifying Schema to Change Attribute's Datatype

```
update pg_class set relname = 's-unknown'||nextval('serial')
where relname in
select relname from pg_class where relnamespace=2200 and
  reltype!=0 and relkind ='r' and relname like 's%'
```

*Figure 4.* Changes to Blocks of Data in Schema

affects both the type of information that can be inserted into a column and the information that will be retrieved when the column is queried. Modifying an attribute's data type in the schema causes data (whether being inserted or retrieved) to be formatted according to this data type. The implication of this is that it becomes impossible to comprehend previously stored information due to changes in the data type.

Depending on the DBMS, various parts of a database schema, including the column names, data types and constraints can be compromised by changing specific values in the schema. An attacker with an understanding of the DBMS structure can execute a SQL query that affects the schema and the result obtained from subsequent queries on the database.

**Changes to Blocks of Data** Apart from changes to specific values, blocks of data such as complete tables or set of columns can be modified through the schema of a database. A typical instance is where an attack changes the name of all the columns with the same name but in different tables to a different name or changes the name of several tables to a different name that is not understandable to the user. For example, executing the code in figure 4, changes the name of all the tables with names starting with letter `s` into a different name. As such, it becomes impossible to retrieve the tables with the original name. In the same way as changing the name of several blocks of data, blocks of data can also be removed or even combined to compromise the database.

**Changes to Links Between Blocks of Data** Another category of changes that can be made to a database schema to compromise the database involves changes that affect the links between data. This type of changes often involves a modification of the foreign keys and/or the primary keys of the relations in the database. It can involve a localized change of specific information in the schema or changes to groups of data in the schema or a combination of both. A typical instance is where an attacker modifies the constraint specified on a table (within the schema) by removing the foreign key, so that it becomes impossible to link the information within two different tables. It is also possible to modify the links such that a table is linked to a wrong table.

Regardless of the category of changes that can be made by an attacker, an important aspect of database forensics deals with the ability to reconstruct both the data in the table and its schema. The following section describes techniques that can be used to reconstruct the schema by considering the operations previously performed on the database.

## 6. Schema Reconstruction

According to Elmasri [3], the widespread use and acceptance of relational DBMSs can be attributed to the simplicity and the mathematical foundation of the relational algebra. In this section, we explore these characteristics in the reconstruction of database schema. In section 1.6.1, we describe how the schema of a relation can be reconstructed by looking at the conditions associated with each of the operators of the relational algebra. In section 1.6.2, we investigate the application the concept of the inverse relational algebra previously used for the reconstruction of data [7, 5] in a database for reconstructing schemas. Section 1.6.3 describes how the schema can be reconstructed by checking the consistency of the information in the database.

### 6.1 Reconstruction from Previous Manipulations

Given a RA log, the attributes of the operations performed on a relation can be used to determine the schema of the relation. Many of the operators of the relational algebra that require two operands have the characteristics that the structure of the two operands are the same. The Union, Intersection and Difference operators fall into this category. Given a query expressed in relation algebra, $C \leftarrow A$ op $B$ where $A, B$ and $C$ are relations and `op` is a union ($\cup$), intersection ($\cap$) or difference operation ($-$), the three relations have exactly the same structure. Thus, if the structure of one of the relations is known, it can be used

as the structure for any of the other two relations and the data in the relation can be reconstructed [7, 5] based on this structure.

This is also true with the selection operator. Given select query, $B \leftarrow \sigma_{p(attr)}(A)$, where some tuples in $A$ are selected as $B$ given the condition expressed as $p(attr)$, both relations $A$ and $B$ have exactly the same structure. And the structure of one can be used for the other during reconstruction of data or to infer the structure of other relations.

An interesting aspect of applying this approach deals with the fact that it makes it possible to combine different operations and analyze what the likely schema for the resulting relation would be.

## 6.2 Reconstruction Using Inverse RA

A more formalized approach to reconstructing the schema of a relation is to apply the reconstruction algorithm for database forensics earlier described in previous works [7, 5]. Since a major part of the algorithm works based on the inverse relational algebra, we describe how the inverses can be used for schema reconstruction in this paper.

**Transposing Schemas into Relations** Since the inverse operators of the relational algebra requires a relation as its operand, it is necessary that the schema of a relation can also be expressed as a relation in order to use the inverse operators in schema reconstruction. Many DBMSs provide the ability to retrieve the structure of a relation on the database as a table. In Postgres for example, the command shown in figure 5 can be used to retrieve the structure of a relation into a table where each attribute of the relation becomes a tuple in the retrieved table. Figure 6 shows examples of schema retrieved as a table.

As a consequence of retrieving the schema of a relation as another relation, we also need to find the *transpose* of the operations performed on the original relations so that the transposed operation can be applied to the retrieved schemas and used for schema reconstruction. Below, we consider each of the relational algebra operators and identify the operation that can be considered as a transposition of each operation.

- Cartesian product ($\times$): Given a query $T \leftarrow R \times S$, we know that the resulting relation $T$ has all the attributes of both $R$ and $S$. That is, if we can retrieve the schema of relations $R$ and $S$, then the schema of $T$ would be the union of both schemas. Thus, the transpose of a Cartesian product operation is a union operation.
- Union ($\cup$): Given a query $T \leftarrow R \cup S$, the schema of $T$ is the same as that of $R$ and $S$. If the schema of each of the operands could be retrieved, then the schema of $T$ would be an intersection

```
select ordinal_position, column_name, data_type, is_nullable,
    descrip.description AS comment
from information_schema.columns columns
left join pg_class class on (columns.table_name = class.relname)
left join pg_description descrip on (class.oid = descrip.objoid)
left join pg_attribute attrib on  (class.oid = attrib.attrelid
    and columns.column_name = attrib.attname
    and attrib.attnum = descrip.objsubid)
where table_name= 'actor'  /* The table name*/
group by ordinal_position, column_name, data_type,
     is_nullable, table_schema, descrip.description;
```

*Figure 5.* Retrieving Schema as a Table

of the two schemas. Thus, the transpose of a union operation is an intersection operation.

- Intersection ($\cap$): Based on the same reason as for the union operation, the transpose of an intersection operation, $T \leftarrow R \cap S$, is also an intersection operation.

- Difference ($-$): In a difference operation expressed as the query, $T \leftarrow R - S$, we know that the schema of $T$ is the same as that of $R$ and $S$. Similar to the union and intersection operation, the transpose of a difference operation is an intersection operation.

- Join ($\bowtie$): A join operation can be compared to a Cartesian product, except for the fact that the Cartesian product is performed under certain specified condition and as such may not include all the tuples that might be in a Cartesian product. However, this has no effect on the attributes of the operands or the resulting relation. As such, given a join query, $T \leftarrow R \bowtie_{p(A,B)} S$, $T$ has all the attributes of both $R$ and $S$. Thus, if we can retrieve the schema of both $R$ and $S$, then the schema of $T$ would be the union of both schemas. This implies that the transpose of a join operation is a union operation.

- Projection ($\pi$): A projection operation selects some of the attributes of a relation as another relation. Given a query, $T \leftarrow \pi_{A_1,A_2,A_3}(R)$, the attributes of $T$ is a subset of that of $R$. Thus, the transpose of the projection operation is a selection operation.

- Selection ($\sigma$): The output of a selection operation, $T \leftarrow \sigma_{p(A)}(R)$, has exactly the same set of attributes as that of the operand, since a selection only affects the tuples of a relation. Thus, the transpose

*Table 2.* Transpose of the Relational Algebra Operators

| **Operators** | **Transposed Operators** |
|---|---|
| Cartesian product ($\times$) | Union ($\cup$) |
| Union ($\cup$) | Intersection ($\cap$) |
| Intersection ($\cap$) | Intersection ($\cap$) |
| Difference ($-$) | Intersection ($\cap$) |
| Join ($\bowtie$) | Union ($\cup$) |
| Projection ($\pi$) | Selection ($\sigma$) |
| Selection ($\sigma$) | Selection ($\sigma$) |
| Division ($/$) | Difference ($-$) |

of a selection operation is a selection (without any conditions) of all the tuples in the schema of the operand.

- Division ($/$): Given a query, $T \leftarrow R[A, B/C]S$, where a relation $R$ is divided by a attribute $C$ of the relation $S$, we know that the attribute of $T$ will include all the attributes of $A$ except the attribute (also in $S$) that was used in the division. That is, if the schema of the two relations $R$ and $S$ are known, then the schema of $T$ would be the schema of $R$ minus the schema of $S$. Thus, the transpose of a divide operation is a difference operation.

Table 2 gives a summary of the *transposed* operations that can be applied in the reconstruction of the schema of a relation. As mentioned in the definition of the inverse operators of the relational algebra, the objective of the inverse operators is to find the value of a particular attribute of a relation at a specific time $t$ by finding the inverse of the most recent query performed on the current relation and sequentially going backwards until the desired time is reached. The same technique can be applied in schema reconstruction, with the addition that the transposed operations are used to handle the schemas involved in any of the relational algebra operations.

**Applying the Inverse RA and Transpose Operators for Reconstruction** In this section, we describe how the transposed operators described above can be used together with the inverse functions defined for the operator in order to reconstruct table schemas. From table 2, there are four relational algebra operators that we need to consider: The union ($\cup$), Intersection ($\cap$), Selection ($\sigma$) and Difference ($-$) operators since the relational algebra operators can be transposed as one of them.

The inverse of a union operation $T \leftarrow R \cup S$ can only be determined if one of the expected outputs (that is $R$ or $S$) is known. If relation $S$ is known, then $\cup^{-1}(T) = (R^*, S)$ where $R^* = T - S$. On the other

| **ordinal_position integer** | **column_name character varying** | **data_type character varying** |
|---|---|---|
| 1 | id | integer |
| 2 | lastname | character varying |
| 3 | firstname | character varying |
| 4 | lastupdate | timestamp |
| 5 | emp_id | integer |
| 6 | dept | character varying |
| 7 | address | character varying |

(a) Schema of Relation $C$ ($S_C$)

| **ordinal_position integer** | **column_name character varying** | **data_type character varying** |
|---|---|---|
| 1 | id | integer |
| 2 | lastname | character varying |
| 3 | firstname | character varying |
| 4 | lastupdate | timestamp |

(b) Schema of Relation $B$ ($S_B$)

*Figure 6.* Retrieved Schemas of Relations $C$ and relation $B$

hand, if $R$ is known, then $\cup^{-1}(T) = (R, S^*)$. A complete inverse union is found only when both $R$ and $S$ have no tuples in common [7, 5]. Since the transpose of the Cartesian product and the join operators is a union operation, this definition can be applied in the reconstruction of the schemas of relations involved in either operations. Given an operation $C \leftarrow A$ `op` $B$, where `op` is a Cartesian product or a join operator, if the schema of $C$ can be retrieved as relation $T$, then the schema of $A$ can be reconstructed using the inverse union operator, provided that the schema of $B$ is known, or vice versa. As an example, if the schema of relation $C$ ($S_C$) and relation $B$ ($S_B$) are retrieved using the code in figure 5 and stored as the tables in figure 6, then we can reconstruct the schema of relation $A$ as $(S_C - S_B)$, which yields the result in figure 7.

The inverse of an intersection operation $T \leftarrow R \cap S$, generates partial tuples inverses containing all the tuples in $T$, that is, $\cap^{-1}(T) = (R^*, S^*)$ where $R^* = S^* = T$. Complete inverses can be found when $R$ and $S$ are known to be the same [7, 5]. The inverse intersection operator can be applied for the reconstruction of the schemas of relations involved in a union, intersection or difference operation since their transpose is an intersection operation. That is, given an operation $C \leftarrow A$ `op` $B$, where `op` is a union, intersection or difference operator, if the schema of $C$ can be retrieved as a relation $T$, then the schema of $A$ and $B$ can be reconstructed using the definition of the inverse intersection operator. A unifying characteristic of the union, intersection, and difference

| **ordinal_position integer** | **column_name character varying** | **data_type character varying** |
|---|---|---|
| 5 | emp_id | integer |
| 6 | dept | character varying |
| 7 | address | character varying |

(a) Reconstructed Schema of Relation $A$ ($S_A$)

| **emp_id** | **dept** | **address** |
|---|---|---|
| ⋮ | ⋮ | ⋮ |
| ⋮ | ⋮ | ⋮ |

(b) Structure of Relation $A$

*Figure 7.* Reconstructed Schema of Relation $A$ Using Inverse Union Operation

operators is that their operands must have the same schema. Thus, if the schema of relation $C$ (or that of $A$ or $B$) is retrieved as a table, then we can completely reconstruct the schema of the other relations by applying the inverse intersection operator.

Given a selection operation $R \leftarrow \sigma_{p(A)}(S)$, the inverse selection is given by, $\sigma^{-1}_{p(A)}(R) = S^*$, where $S^* = R$. That is, all the tuples in $R$ are also in $S$. The inverse selection yields a complete inverse if all the tuples in the operand of the selection operator (that is, $S$) satisfied the condition specified as $p(A)$ [7, 5]. The inverse selection operation can be applied in the reconstruction of the schemas of relations involved in a projection or a selection operation since the transpose of both operations is a selection operation. That is, given a projection operation $C \leftarrow \pi_{A_1,A_2}(B)$ or a selection operation $C \leftarrow \sigma_{p(A)}(B)$, if the schema of relation $C$ is known (say $S_C$), then the schema of $B$ contains all the tuples in $S_C$. In the case of a projection operation the reconstructed schema may be incomplete since a projection is usually a selection of some columns of the operand. In the case of a selection operation, the reconstructed schema is complete since a selection picks some of the tuples of its operand with all their attributes.

The inverse difference operation can be applied for the reconstruction of the schema of the operands of a division operation since its transpose is a difference operation. Given a difference operation $T \leftarrow R - S$, the left operand is easily determined by the inverse difference operator as $R^* = T$ since $T \subseteq R$. A complete $R$ can be determined only if the relation $S$ is known and all the tuples in $S$ are also known to be in $R$ (that is, $S \subseteq R$) so that $R = T \cup S$. The relation $S^*$ with partial tuples can also be determined if $R$ is known, in which case, $S^* = R - T$. If we know that $S \subseteq R$, then a complete relation $S$ is found from the inverse

as $S = R - T$ [7, 5]. This implies that if the schema of $C$ ($S_C$) in the divide operation $C = A/B$ is known, then we know that the schema of $A$ contains all the tuples in $S_C$. The schema may be incomplete since the relation $A$ may contain other attributes that were not included in the divide operation. The reconstructed schema for $A$ is given by $S_C \cup S_B$ and is complete if we know the schema of $B$ ($S_B$) and also know that all the columns in $B$ are also in $A$. On the other hand, the schema of relation $B$ can be reconstructed as $S_A - S_C$ if the schema of $A$ ($S_A$) is known. Also if all the columns in $B$ are also in $A$, then the reconstructed schema for relation $B$ is complete.

Although reconstructed schemas are generated as a table, the structure of the corresponding table is simply a matter of transposing the reconstructed schema so that tuples in the reconstructed table becomes the columns of the required relation. The technique described above can be applied to sequence of operations in order to arrive at the particular relation whose schema is required. This is equivalent to working with database reconstruction algorithm described by Fasan and Olivier [7, 5], and using the transposed operations and the schemas expressed as tables instead of the actual operator and operands involved in an operation.

## 6.3 Reconstruction Through Consistencies

Another approach that can be used for the reconstruction of a database schema involves checking the consistency of the information contained in the database. Typically, a DBMS provides the ability to represent complex relationships between data. The description of these relationships are stored in the schema and imply that certain conditions hold between related data. An understanding of the conditions expected to hold in any relationship can be used to identify instances that fail to satisfy these conditions and may point to actions performed by an attacker.

This approach is particularly useful for the reconstruction of the constraints associated with a relation. For example, if there is a referential integrity constraint [3] from a relation $R$ to a relation $S$, then the attributes concerned in both relations are expected to be of the same data type. This knowledge can be used to determine the data type of the attribute involved if either of the schemas of $R$ or $S$ needs to be reconstructed. The referential integrity constraint also points to the fact that the referenced column is the primary key of one of the two relations. Another example of the application of database consistencies for reconstruction is the application of the entity integrity constraint [3], which specified that no primary key value can be a NULL value. This shows

that an attribute that can take a null value is not the primary key of the relation whose schema is to be reconstructed and vice versa.

A combination of the different constraints that can be specified on a database as well as the characteristics of various attributes may also be useful for identifying the characteristics relating to the schema of a relation. We note that even though it may be possible to retrieve some of the constraints in a schema using the two techniques earlier described, it may be necessary to use additional techniques such as checking for the data consistencies in reconstructing the constraints relating to a relation.

## 7. Conclusion and Future Work

This paper aimed to describe techniques that can be used for the reconstruction of the schema of the relations in a database. The paper builds on the inverse function of the operators of the relational algebra, previously defined for the reconstruction of the data that exist in a database at an earlier time of interest. By using typical examples, we first show that a database schema can indeed be compromised, causing queries to yield results that may be different from the actual data. The paper then describes techniques that can be employed in reconstructing the structure of the relations in a database based on the operation perform on them. Examples are given to illustrate the techniques.

Future work will include measuring the effectiveness of the different approaches for schema reconstruction in different error scenario. In addition, it may also be interesting to consider the problem of schema reconstruction from a big data or NoSQL perspective.

## Acknowledgement

This research was supported by the Organization for Women in Science for the Developing World (OWSD).

## References

[1] Hector Beyers, Martin Olivier, and Gerhard Hancke. Assembling metadata for database forensics. In *Advances in Digital Forensics VII*, pages 89 – 99. Springer Berlin Heidelberg, 2011.

[2] E. F. Codd. *The Relational Model for Database Management, Version 2.* Addison-Wesley, 1990.

[3] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems.* Addison-Wesley Publishing Company, 6th ed., 2010.

[4] Oluwasola Mary Fasan and Martin Olivier. On the completeness of reconstructed data for database forensics. In *Proceedings of the 4th*

*Inter. Conf. on Digital Forensics and Cyber Crime*, USA, 2012.

[5] Oluwasola Mary Fasan and Martin S. Olivier. Correctness proof for database reconstruction algorithm. *Digital Investigation*, 9(2):138–150, 2012.

[6] Oluwasola Mary Fasan and Martin S. Olivier. On dimensions of reconstruction in database forensics. In *Proceedings of the 7th International Workshop on Digital Forensics and Incident Analysis (WDFIA 2012)*, pages 97 – 106. Plymouth University, 2012.

[7] Oluwasola Mary Fasan and Martin S. Olivier. Reconstruction in database forensics. In *IFIP International Conference on Digital Forensics*, pages 273–287, January 2012.

[8] Kevvie Fowler. *SQL Server Forensic Analysis*. Addison Wesley Professional, 2008.

[9] P. Frühwirt, M. Huber, M. Mulazzan, and E. R. Weippl. InnoDB database forensics. In *24th IEEE Inter. Conf. on Advanced Information Networking and Applications*, pages 1028–1036, 2010.

[10] P. Frühwirt, P. Kieseberg, S. Schrittwieser, M. Huber, and E. Weippl. InnoDB database forensics: Reconstructing data manipulation queries from redo logs. In *7th International Conference on Availability, Reliability and Security*, pages 625–633, 2012.

[11] Simson L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7, Supplement(0):S64 – S73, 2010.

[12] David Litchfield. Oracle forensics part 1 - part 6, 2007. NGSSoftware Insight Security Research Publication.

[13] S. Nebiker and S. Bleisch. *Introduction to Database Systems*. Geographic Infor. Tech. Training Alliance (GITTA), June 2010.

[14] Martin S. Olivier. On metadata context in database forensics. *Digital Investigation*, 5(3-4):115–123, 2009.

[15] Gary Palmer. A road map for digital forensic research. Technical report, First Digital Forensic Research Workshop, New York, 2001.