Chapter 8

# SCHEMA RECONSTRUCTION IN DATABASE FORENSICS

Oluwasola Mary Adedayo and Martin Olivier

**Abstract** Although considerable research has been conducted in the area of database forensics over the past few years, several aspects of database forensics remain to be considered. One of the challenges facing database forensics is that the results of forensic analysis may be inconsistent with the raw data contained in a database because of changes made to the metadata. This paper describes the various types of changes that can be made to a database schema by an attacker and shows how metadata changes can affect query results. Techniques for reconstructing the original database schema are also described.

**Keywords:** Database forensics, database reconstruction, inverse relational algebra

## 1. Introduction

Databases are a core component of many computing systems and play an important role in many organizations. However, the use of databases to store critical and sensitive information has led to an increase in the rate at which they are exploited to facilitate computer crimes [5]. Database forensics is the branch of digital forensics [11, 15] that deals with the identification, preservation, analysis and presentation of evidence from databases [8]. Over the past few years, researchers [2, 14] have emphasized the need to incorporate database forensic analysis as part of traditional digital forensics due to the amount of information that is often retrieved from a database during an investigation. Although considerable research has been conducted in the area of database forensics, much work remains to be done with regard to formalizing the database forensic process.

The use of database logs as a source of information for forensic analysis has been the subject of many research efforts (see, e.g., [5, 7, 9,

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics X, IFIP AICT 433, pp. 101–116, 2014.

10, 12]). Database logs can be explored in a variety of ways in order to reconstruct the queries executed on a database and the values in the database at an earlier time. In addition, a database can serve as a tool to support the forensic analysis of the database itself because it permits the execution of complex queries for information retrieval [14]. An important aspect of database forensics deals with the ability to revert the operations performed on a database so that the information that existed in the database at an earlier time can be reconstructed [1, 5, 7]. One aspect of database forensics that has not yet been considered deals with the reconstruction of the database schema or metadata. Although the importance of the database schema has been discussed in the database forensic literature [2, 5, 7, 12], no method currently exists for reconstructing a database schema.

In earlier work [5, 7], we presented an algorithm for reconstructing the data that was present in a database at an earlier time. Several techniques to ensure the completeness of the reconstructed information have also been considered [1]. This paper builds on the previous work by describing techniques for reconstructing the database schema. It 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 provided to illustrate the reconstruction techniques.

## 2. Database Management Systems

One of the fundamental characteristics of a database system is that it contains the stored data as well as a complete definition and description of the database and the stored data. This description, which is referred to as "metadata" or the "database schema," is stored in the system catalog. The metadata contains details about the structure, type and format of each file and data item, along with the relationships between the data and other constraints [4, 13]. The metadata is used by the database management system (DBMS) to provide information about the database structure to database users and application programs. Because the DBMS software is not written for one specific database application, the metadata provides the information needed by the DBMS to understand the structure, type and format of files and data that it will access.

It well known that the result of a database query is a function of the metadata and the data stored in the database. This characteristic can be exploited to facilitate criminal activities. One example is when a criminal creates a view (stored as metadata) that provides easy access to data of interest. Although the raw data may still exist, it may not be retrievable after the metadata is deleted [14].

## 3. Dimensions of Database Forensics

In a recent work [6], we discussed three dimensions of database forensics: 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 after an event of interest occurred. This type of database 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 [5–7].

In contrast, a damaged database is a database whose data or files may have been modified, deleted or copied from their original locations to other places. Such a database may or may not be operational depending on the extent of the damage.

A compromised database is a database in which some metadata or DBMS software have been modified by an attacker although the database may still be operational. Olivier [14] and Beyers, *et al.* [2] have 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 because the database could have been modified into giving false information. Litchfield [12] also identified this problem when discussing the steps for performing a live response to an attack on an Oracle database. The major portion of the investigation of a compromised database focuses on obtaining the correct version of the metadata.

This paper provides insight into this problem by describing the various ways in which a database can be compromised by changing the metadata. Also, it shows how the problem can be addressed by reconstructing the database schema. The schema reveals the design decisions regarding the tables and their structures and, as such, can be used to obtain a better understanding of the results obtained from database queries. The paper describes typical examples of database compromise and shows how the schema can be reconstructed by considering operations that were previously performed on the database. In earlier work [7], we presented an algorithm for reconstructing database data. This paper considers the application of some of the previously-described data reconstruction techniques to the problem of reconstructing the database schema.

## 4. Database Reconstruction

This section provides a brief overview of the main concepts underlying the paper. In particular, it describes the relational model, the relational algebra, the relational algebra log and the inverse operators of the relational algebra.

*Table 1.* Relational algebra operators 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 |

**Relational Algebra and Relational Algebra Log.** The relational model developed by Codd [3] is based 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 the attributes $A = \{A_1, A_2, \ldots A_n\}$ are defined, a relation $R$ (or $R(A)$) is a subset of the Cartesian product of the domains [3].

The relational algebra consists of basic operators for manipulating relations and a relational assignment operator ($\leftarrow$). The basic operators transform one or two relations into a new relation. The basic relational operators as defined by Codd [3] are listed in Table 1 along with the inverse operators (explained below). In the table, $R$, $S$ and $T$ denote relations, while $A$, $B$ and $C$ denote attributes of relations. Note that $p(attributes)$ 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 [5, 7], a relational algebra log (RA log) is a log of queries expressed as operations involving relational algebra operators instead of the traditional SQL notation. It presents several advantages over the traditional SQL notation [7].

**Inverse Relational Algebra.** The inverse operators of the relational algebra defined in our earlier work [5, 7] are used to obtain the value of an attribute $A$ (corresponding to a tuple in relation $R$) at a specific time by computing the inverse of the most recent query performed on the current relation and sequentially retracing the queries backward until the desired time is reached. Depending on whether or not the result generated by an inverse operator is complete (i.e., without any missing information), the inverse operators of the relational algebra are categorized as a "complete inverse" or a "partial inverse." Note that, regardless of this classification, there are often instances where a complete inverse can be found for an operator that is categorized as a partial inverse operator. Given a relation $R$ generated by a inverse operation, the notation $R^*$ is used to denote that the inverse generated is incomplete. Table 1 lists the inverse operators of the relational algebra. Interested readers are referred to [5, 7] for a detailed description of the inverses and the conditions under which complete inverses exist.

## 5. Compromising a Database Schema

This section describes several ways in which a database schema may be compromised. Also, it demonstrates how a compromise can cause a database to give incorrect answers to queries. The study was performed using Postgres 9.2 running on a Windows 8 operating system.

Several actions can be performed on a database schema to compromise the database. Usually, an attacker may simply damage a database to prevent it from being used for its intended purpose. An attacker may also compromise a database in an attempt to hide some data stored by a user or by the attacker. In general, an attacker could make three types of changes:

- Localized changes to data
- Changes to blocks of data
- Changes to links between blocks of data

**Localized Changes to Data.** An attacker can make changes to specific columns of a relation within a database schema. The changes may be made to hide the information contained in the columns or to make the data unavailable to database users. This can be accomplished in several ways. A compromise involving the swapping of two columns can have serious effects on database operations [14]. An example is swapping the "Purchase Price" and "Selling Price" columns in a store database, which would cause the store to sell its goods at a lower price. Figure 1

```
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 a schema to swap two column names.

shows a code segment that can be used to compromise a database by swapping the column names in the database schema. The swap causes a `select` statement on one of the columns to return values from the second column.

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

*Figure 2.* Modifying a schema to hide a column.

A database schema could also be manipulated by changing the identification of an attribute (within the schema) to something unknown to the schema. Although the raw data may still be present in the database, it becomes impossible to query the database. Information initially thought to have been deleted or non-existent could, in fact, turn out to be non-retrievable because it is hidden from the DBMS by schema modification. Figure 2 shows a command that can be used to hide the attribute `purchasePrice` from authorized users by changing the identification used by the DBMS to access the column.

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

*Figure 3.* Modifying a schema to change the datatype of an attribute.

Details such as the datatypes of the attributes of a relation can be modified by changing specific values in the schema (Figure 3). Such a change affects the type of information that can be inserted into a column as well as the information that is retrieved when the column is queried. Modifying the datatype of an attribute in the schema causes data (whether it is being inserted or retrieved) to be formatted according to the new datatype; this makes it impossible to comprehend previously-stored information.

Depending on the DBMS, several parts of a database schema, including the column names, datatypes and constraints can be compromised by changing specific values in the schema. An attacker with an under-

```
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 a schema.

standing of the DBMS structure can execute an SQL query that affects the schema and the results obtained from subsequent database queries.

**Changes to Blocks of Data.** Blocks of data such as complete tables and sets of columns can be modified via the database schema. Two simple examples are an attack that changes identically-named columns in multiple tables to a different name and an attack that changes the names of several tables to a different name that is not understandable to users. The code in Figure 4 changes the names of all tables with names starting with the letter `s` to a different name. This makes it impossible to retrieve the tables using the original name. In a similar manner to changing the names of blocks of data, blocks of data could be removed or even combined to compromise the database.

**Changes to Links Between Blocks of Data.** Another method of compromising a database involves changes that affect the links between data. This is accomplished by modifying the foreign keys and/or primary keys of database relations. It can involve localized changes to information in the database schema or changes to groups of data in the schema or a combination of both. An example attack is the modification of a table constraint (within the schema) by removing the foreign key, making it impossible to link information in another table. It is also possible to modify the links such that the table is linked to a wrong table.

Regardless of the types of changes made by an attacker, an important aspect of database forensics is the ability to reconstruct the data in a table and its schema. The next section describes techniques that can be used to reconstruct the schema by considering the operations that were previously performed on the database.

## 6. Schema Reconstruction

According to Elmasri and Navathe [4], the widespread use of relational DBMSs is due to their simplicity and the mathematical foundation of the relational algebra. This section leverages these characteristics to reconstruct the database schema. Section 6.1 describes how the schema of

a relation can be reconstructed by examining the conditions associated with each relational algebra operator. Section 6.2 investigates the application of inverse relational algebra operators used for reconstructing database data [5, 7] to reconstruct a database schema. Finally, Section 6.3 describes how a schema can be reconstructed by checking the consistency of database information.

## 6.1 Reconstruction from Manipulations

Given an RA log, the attributes of the operations performed on a relation can be used to determine the schema. Many of the relational algebra operators with two operands require the structures of the operands to be the same. The Union, Intersection and Difference operators fall in this category. Given a relational algebra query, $C \leftarrow A \texttt{ op } B$, where $A, B$ and $C$ are relations and `op` is a Union ($\cup$), Intersection ($\cap$) or Difference operation ($-$), then the three relations $A, B$ and $C$ have the same structure. Thus, if the structure of one of the relations is known, the same structure can be used for the other two relations and the data in the relations can be reconstructed [5, 7] based on this structure.

This is also true with the Selection operator. Given a Select query, $B \leftarrow \sigma_{p(attr)}(A)$, where some tuples in $A$ are selected as $B$ based on the condition $p(attr)$, then the relations $A$ and $B$ have exactly the same structure and the known structure of one relation can be used as the structure of the other relation in order to reconstruct data or to infer the structure of other relations.

An interesting aspect of this approach is that it is possible to combine different operations and analyze the likely schema for the resulting relations.

## 6.2 Reconstruction Using Inverse RA

A formal approach for reconstructing the schema of a relation is to apply the reconstruction algorithm for database forensics described in previous work [5, 7]. Since the algorithm uses inverse relational algebra operators, we proceed to describe how the inverses can be used for schema reconstruction.

**Transposing Schemas into Relations.** Since the inverse operators of the relational algebra require a relation as an operand, the schema of a relation must 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 in the database as a table. In Postgres, for example, the command shown in Figure 5 can be used to

```
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 a schema as a table.

retrieve the structure of a relation into a table where each attribute of the relation is a tuple in the retrieved table.

As a consequence of retrieving the schema of a relation as another relation, it is also necessary to find the "transpose" of the operations performed on the original relations so that the transposed operations can be applied to the retrieved schemas and used for schema reconstruction. We now consider each relational algebra operator and identify the operation that can be considered to be its transposition:

- **Cartesian Product ($\times$):** Given a query $T \leftarrow R \times S$, the resulting relation $T$ has all the attributes of both $R$ and $S$. Thus, if the schemas of relations $R$ and $S$ are retrieved, then the schema of $T$ would be the union of the two schemas. Consequently, 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 those of $R$ and $S$. If the schemas of the two operands could be retrieved, then the schema of $T$ would be the intersection of the two schemas. Thus, the transpose of a Union operation is an Intersection operation.
- **Intersection ($\cap$):** Using the same reasoning as for the Union operation, the transpose of an Intersection operation, $T \leftarrow R \cap S$, is also an Intersection operation.
- **Difference ($-$):** Given a Difference operation expressed as the query, $T \leftarrow R - S$, the schema of $T$ is the same as those of $R$ and $S$. As in the case of the Union and Intersection operations, the transpose of the Difference operation is an Intersection operation.
- **Join ($\bowtie$):** A Join operation is similar to a Cartesian Product, except that the Cartesian Product is performed under a specified

*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 ($-$) |

condition and, as such, the result of the Join 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 on the resulting relation. Given a Join query, $T \leftarrow R \bowtie_{p(A,B)} S$, it is clear that $T$ has all the attributes of both $R$ and $S$. Thus, if the schema of both $R$ and $S$ can be retrieved, then the schema of $T$ would be the union of the two 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 to produce another relation. Given a query, $T \leftarrow \pi_{A_1,A_2,A_3}(R)$, it is clear that the attributes of $T$ correspond to a subset of the attributes of $R$. Thus, the transpose of a 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; this is because a Selection only affects the tuples of a relation. Thus, the transpose 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 an attribute $C$ of relation $S$, it is clear that the attribute of $T$ will include all the attributes of $A$ except the attribute (also in $S$) that was used in the Division operation. 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 Division operation is a Difference operation.

Table 2 summarizes the transposed operations that can be applied to reconstruct the schema of a relation. The value of a particular attribute of a relation at time $t$ is obtained by computing the inverse of the most

recent query performed on the current relation and going backward in sequence until the desired time $t$ is reached. The same technique can be applied to schema reconstruction, except that the transposed operations are used to handle the schemas involved in the relational algebra operations.

**Applying the Inverse RA and Transpose Operators.** This section describes how the transposed operators described above can be used together with the inverse operators to reconstruct table schemas. According to Table 2, there are four relational algebra operators that must be considered: (i) Union ($\cup$); (ii) Intersection ($\cap$); (iii) Selection ($\sigma$); and (iv) Difference ($-$). This is because the relational algebra operators can be transposed using these four operators.

The inverse of a Union operation $T \leftarrow R \cup S$ can only be determined if one of the expected outputs (i.e., $R$ or $S$) is known. If relation $S$ is known, then $\cup^{-1}(T) = (R^*, S)$ where $R^* = T - S$. On the other hand, if $R$ is known, then $\cup^{-1}(T) = (R, S^*)$. A complete Inverse Union exists only when $R$ and $S$ have no tuples in common [5, 7]. Since the transpose of the Cartesian Product and Join operators is a Union operation, this definition can be applied to reconstruct the schemas of relations involved in either operation. Given an operation $C \leftarrow A$ `op` $B$, where `op` is the Cartesian Product or 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 schemas of relations $C$ ($S_C$) and $B$ ($S_B$) are retrieved using the code in Figure 5 and stored as the tables in Figure 6, then the schema of relation $A$ can be reconstructed 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 tuple inverses containing all the tuples in $T$, i.e., $\cap^{-1}(T) = (R^*, S^*)$ where $R^* = S^* = T$. Complete inverses exist when $R$ and $S$ are known to be the same [5, 7]. The Inverse Intersection operator can be applied to reconstruct the schemas of relations involved in a Union, Intersection or Difference operation because their transpose is an Intersection operation. That is, given an operation $C \leftarrow A$ `op` $B$, where `op` is the Union, Intersection or Difference operator, if the schema of $C$ can be retrieved as a relation $T$, then the schemas of $A$ and $B$ can be reconstructed using the definition of the Inverse Intersection operator. A unifying characteristic of the Union, Intersection and Difference 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 the schemas of the

| 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 $B$.

| 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.

other relations can be reconstructed completely 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 (i.e., $S$) satisfy the predicate $p(A)$ [5, 7]. The Inverse Selection operator can be applied to reconstruct the schemas of relations involved in a Projection or Selection operation because 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 because a Projection is usually a Selection of some of the columns of the operand. In the case of a Selection operation, the reconstructed schema is complete because the Selection operator picks some of the tuples of its operand with all their attributes.

The Inverse Difference operation can be applied to reconstruct the schema of the operands of a Division operation because 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 $R^* = T$ because $T \subseteq R$. 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 $R$ (i.e., $S \subseteq R$) 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 $S \subseteq R$ in known, then a complete relation $S$ is found from the inverse as $S = R - T$ [5, 7]. This implies that, if the schema of $C$ ($S_C$) in the Division operation $C = A/B$ is known, then the schema of $A$ contains all the tuples in $S_C$. The schema may be incomplete because relation $A$ may contain other attributes that were not included in the Division operation. The reconstructed schema for $A$ is given by $S_C \cup S_B$ and is complete if the schema of $B$ ($S_B$) is known and it is also known 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 a reconstructed schema is generated in the form of a table, the structure of the corresponding table is simply a matter of transposing the reconstructed schema so that tuples in the reconstructed table become the columns of the required relation. The technique described above can be applied to a sequence of operations in order to arrive at the particular relation whose schema is required. This is equivalent to working with the database reconstruction algorithm described by Fasan and Olivier [5, 7], and using the transposed operations and the schemas expressed as tables instead of the actual operators and operands involved in relational algebra operations.

### 6.3 Reconstruction Through Consistencies

Another approach that can be used to reconstruct a database schema is to check the consistency of the information contained in the database. Typically, a DBMS provides the ability to represent complex relationships between data. The descriptions of these relationships are stored in the schema and imply that certain conditions hold between related data. Identifying instances that do not satisfy these conditions could point to actions performed by an attacker.

This approach is particularly useful for reconstructing the constraints associated with a relation. For example, if a referential integrity constraint [4] exists from a relation $R$ to a relation $S$, then the relevant attributes in the two relations would be expected to have the same datatypes. This information can be used to determine the datatypes of the attributes if the schemas of $R$ and/or $S$ need to be reconstructed. The referential integrity constraint also implies that the referenced column is the primary key of one of the two relations.

Another example is the application of the entity integrity constraint [4], which specifies that a primary key value cannot be a null value. Thus, 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.

Database constraints and attribute characteristics can also help identify characteristics relating to the schema of a relation. While it may be possible to retrieve some of the constraints in a schema using the two techniques described above, it may be necessary to use additional techniques such as checking for data consistencies when reconstructing the constraints relating to a relation.

## 7. Conclusions

A database schema can be compromised by attackers in several ways, causing queries to yield results that may be different from the actual data stored in the database. The techniques described for reconstructing relation schemas are useful in forensic investigations of compromised databases. These techniques leverage the inverses of relational algebra operators that were originally defined to reconstruct data that may have existed in a database at an earlier point in time.

Future research will focus on measuring the effectiveness of the schema reconstruction approaches in various error scenarios. In addition, it will investigate the problem of schema reconstruction from the big data and NoSQL perspectives.

## Acknowledgement

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

## References

[1] O. Adedayo and M. Olivier, On the completeness of reconstructed data for database forensics, *Proceedings of the Fourth International Conference on Digital Forensics and Cyber Crime*, pp. 220–238, 2013.

[2] H. Beyers, M. Olivier and G. Hancke, Assembling metadata for database forensics, in *Advances in Digital Forensics VII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 89–99, 2011.

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

[4] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, Boston, Massachusetts, 2011.

[5] O. Fasan and M. Olivier, Correctness proof for database reconstruction algorithm, *Digital Investigation*, vol. 9(2), pp. 138–150, 2012.

[6] O. Fasan and M. Olivier, On dimensions of reconstruction in database forensics, *Proceedings of the Seventh International Workshop on Digital Forensics and Incident Analysis*, pp. 97–106, 2012.

[7] O. Fasan and M. Olivier, Reconstruction in database forensics, in *Advances in Digital Forensics VIII*, G. Peterson and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 273–287, 2012.

[8] K. Fowler, *SQL Server Forensic Analysis*, Addison-Wesley, Boston, Massachusetts, 2009.

[9] P. Fruhwirt, M. Huber, M. Mulazzani and E. Weippl, InnoDB database forensics, *Proceedings of the Twenty-Fourth IEEE International Conference on Advanced Information Networking and Applications*, pp. 1028–1036, 2010.

[10] P. Fruhwirt, P. Kieseberg, S. Schrittwieser, M. Huber and E. Weippl, InnoDB database forensics: Reconstructing data manipulation queries from redo logs, *Proceedings of the Seventh International Conference on Availability, Reliability and Security*, pp. 625–633, 2012.

[11] S. Garfinkel, Digital forensics research: The next 10 years, *Digital Investigation*, vol. 7(S), pp. S64–S73, 2010.

[12] D. Litchfield, Oracle Forensics, Parts 1–6, NGSSoftware Insight Security Research Publication, Next Generation Security Software, Manchester, United Kingdom, 2007–2008.

[13] S. Nebiker and S. Bleisch, *Introduction to Database Systems*, Geographic Information Technology Training Alliance, Zurich, Switzerland, 2010.

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

[15] G. Palmer, A Road Map for Digital Forensic Research, Report from the First Digital Forensic Research Workshop, DFRWS Technical Report, DTR-T001-01 Final, Utica, New York, 2001.