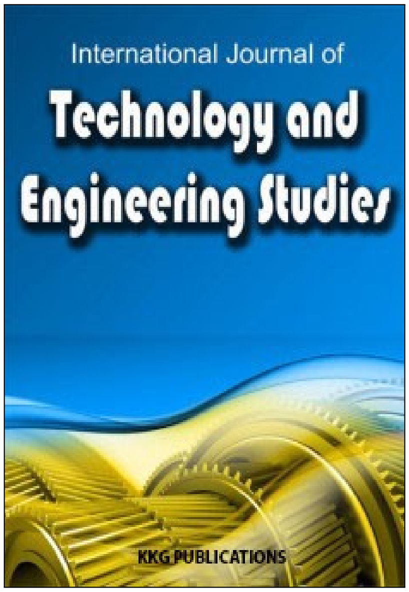

Key Knowledge Generation
Publication details, including instructions for author and subscription information:
http://kkgpublications.com/technology/

Method of Recovery of Deleted Records in a Postgre SQL Database

SEONGHWAN KIM [1], KWANGSIK CHUNG [2]

[1, 2] Department of Computer Science, Graduate School of Korea National Open University, Seoul, South Korea

Published online: 17 August 2017

To cite this article: S. Kim and K. Chung "Method of recovery of deleted records in a postgre SQL database," *International Journal of Technology and Engineering Studies,* vol. 3, no. 4, pp. 169-176, 2017.
DOI: https://dx.doi.org/10.20469/ijtes.3.40005-4

To link to this article: http://kkgpublications.com/wp-content/uploads/2017/3/IJTES-40005-4.pdf

KKG PUBLICATIONS

International Journal of Technology and Engineering Studies
vol. 3, no. 4, pp. 169-176, 2017

IJTES

METHOD OF RECOVERY OF DELETED RECORDS IN A POSTGRESQL DATABASE

SEONGHWAN KIM [1*], KWANGSIK CHUNG [2]

[1, 2] Dept. of Computer Science, Graduate School of Korea National Open University, Seoul, South Korea

Keywords:
PostgreSQL
Record
Delete
Recovery
Forensic
VACUUM
Hex editor

Received: 28 April 2017
Accepted: 04 June 2017
Published: 17 August 2017

Abstract. This paper recommends recovery methods for records deleted from PostgreSQL. This paper includes a description of PostgreSQL, which is the subject of this study. It also incorporates a layout of the pages and records relevant to data storage and the deletion tests and recovery algorithm used. When recovering deleted records, PostgreSQL data files, which contain a history of deleted records, are required. Also, a Hex editor is used, which can check the contents of data files. One can analyze each record's header information by opening PostgreSQL data files with a Hex editor. Then, the analyzed header information from the records is checked. If it is deemed that there are deleted records, they can be extracted for recovery. If VACUUM, a record cleanup program offered by PostgreSQL, has been used for deletion, the records cannot be restored. In this study, it was possible to recover the deleted records of the PostgreSQL database in the damages caused by cybercrime. Recovered records can be used as legal evidence, which can benefit companies in a court situation. The benefit of being in a court situation helps protect the interests of the enterprise.

INTRODUCTION

As enterprises depend significantly on IT systems, and as the amount of handled information increases, the use of databases grows and their range of usage also widens. With the increased use of databases, and a rise in IT-related corporate crimes, more and more database records are being permitted as legal evidence. However, database records can be deliberately or routinely deleted. Therefore, in order for database records to be allowed as legal evidence, these deleted records should be recovered.

Previous studies on the methods used to restore records deleted from databases have only been carried out in regard to Oracle and certain DBMS. Research on the recovery methods for records deleted from PostgreSQL, an open source database, has not yet been carried out. PostgreSQL is a very important database with 4th largest market share in the world, but there is no mention of PostgreSQL in other studies and no research has been done [5]. We have studied PostgreSQL's record recovery method, which was not existing. Hence, this paper recommends methods of recovery for records deleted in PostgreSQL. With respect to the methods used to recover deleted records, PostgreSQL data files, which feature a history of deleted records, are necessary. Also, a Hex editor, which checks the content of the data files, is used. PostgreSQL data files are opened with the Hex editor, and then each record's header information is analyzed. If it is deemed that the record was deleted, it is extracted and recovered.

This paper analyzes database changes using generation and deletion tests for records; it also suggests recovery methods for deleted records based on the analysis, identifies exceptional cases, and investigates records' retrievability.

This paper includes a description of PostgreSQL, which is the subject of this study, and it also incorporates a layout of the pages and records relevant to data storage, and the deletion tests and recovery algorithm that were used.

LITERATURE REVIEW

DBMS, whose recovery methods for deleted records were studied, include Oracle, DB2, and the SQL Server, among other commercial DBMS, as well as MySQL (an open-source DBMS) [1], [2], [3], [4]. As for the relevant research, we will select two DBMSs among the commercial DBMSs, which have the top two highest market shares, and we will investigate their recovery methods.

In this paper, we will investigate the recovery methods for the records and tables in PostgreSQL. In addition, we will identify the threshold at which records and tables cannot be restored, and we will also discuss any pertinent test results.

*Corresponding author: Seonghwan Kim
†Email: auidy@naver.com

The Method of Recovery for Deleted Records in the Oracle Database

To restore records deleted from Oracle, the System tablespace and user tablespace created by the user are required. By figuring out the blocks of user tablespace, the record contents of every table can be identified, and one can also determine whether the records were deleted according to the flag value for that record. When deleting records, one flag value for each record of the data block changes. Flag values change from 0x2C to 0x3C, while other information remains fixed [1].

When dropping a table, two flag values for the deleted tables of the OBJ$ schema and C_OBJ$ schema in system tablespace change. The flag value for the deleted tables in the OBJ$ schema changes from 0x2C to 0x3C, and the flag value for the deleted tables in the C-OBJ$ schema changes from 0x6C to 0x7C.

Exceptionally, when the Shrink feature, which cleans up deleted space to optimize the database, is used, the deleted records and tables cannot be restored. Recovery of the tables is only possible when the records are not overwritten.

After extracting the deleted records and saving them as separate files, the restoration of deleted records ends. Table recovery was not mentioned in the relevant research.

The Method of Recovery for Deleted Records in DB2 Database

When deleting records from DB2, the flag value for EXTENT MAP inside the tablespace changes. When generating records, the flag values are given sequential values, and when they are deleted, the flag value changes to 0xFFFF while the records' other information remains fixed.

When dropping a table, the records are kept and they can be restored. However, the flag values that are related to deleting a table are not identified in the relevant research [2].

There are exceptional cases in which records cannot be recovered; this is when the last records are deleted and when new records are generated. In this case, it is impossible to recover the record, for the information on the last deleted record has been overwritten. In addition, when the database inside the DBMS is deleted with the DROP command, the records included in the deleted database cannot be recovered.

After extracting the deleted records and saving them as separate files, restoration work for the deleted records ends. There is no reference to the restoration work required for tables in the relevant research.

METHOD AND MATERIALS

PostgreSQL

PostgreSQL is an Object-Relational Database Management System (ORDBMS) that is based on POSTGRES 4.2, which was developed by UC Berkeley's Computer Science Department [6].

PostgreSQL supports representative functionalities, such as complex syntax, foreign key, trigger, updatable view, transaction integrity, Multi-Version Concurrency Control (MVCC), etc. and it follows SQL standards. Also, PostgreSQL can be extended by users in a number of ways. Functionalities that can be extended include data type, function, operator, aggregate function, index method, procedural language, etc.

When deleting records in PostgreSQL, the DELETE command is used. The format of the DELETE command is "DELETE FROM table name WHERE column name = 'column value'". When not using the WHERE condition and using the DELETE FROM table name together, all records in the table are deleted. Since there is no additional confirmation for deletion when deleting records, one must be careful. This research was carried out based on PostgreSQL 9.4.

Data Directory and File Structure

PostgreSQL defines the settings and data files used in DBMS as PGDATA. Generally, PGDATA means /var/lib/pgsql/data. PGDATA includes several subdirectory and control files, which are shown in TABLE 1. Also, settings files such as postgresql.conf, pg_hba.conf, and pg_ident.conf are included in PGDATA, but they can be saved in other places as well [7], [8], [11].

TABLE 1
DATA DIRECTORY AND FILES

Item	Description
postmaster.pid	A lock file written the current process ID, data directory path, start
timestamp, port,	unix domain socket path, listen address, shared memory segment ID
postgresql.auto.conf	A file for storing config parameters that are set by alter system
postmaster.opts	A file for command line options the server was started with
PG_VERSION	A file written major version information
base	Directory including data directory
global	Directory including system table
pg_clog	Directory including status of transaction
pg_dynshmem	Directory including files that used by dynamic shared memory's subsystem
pg_logical	Directory including status data for logical decoding
pg_multixact	Directory including data of multi transaction status
pg_notify	Directory including data of listen/notify status
pg_replslot	Directory including data of replication slot
pg_serial	Directory including information about committed serializable transactions
pg_snapshots	Directory including experted snapshot
pg_stat	Directory including permanent file about statistics
pg_stat_tmp	Directory including temporary file for statistics
pg_subtrans	Directory including data of subtransaction status
pg_tblspc	Directory including symbolic link for tablespace
pg_twophase	Directory including status of prepared transaction
pg_xlog	Directory including write ahead log file

The database generated by a user is generated as a subdirectory 'base', which is a subdirectory of PGDATA. The directory name of the database generated by the user becomes OID (ObjectID), which is automatically generated in the DBMS. The table name generated by the user becomes OID, which is automatically generated in the DBMS. Temporary tables are generated in tAAA_BBB form; AAA serves as identification of the process that generated the file, and BBB becomes an OID that is automatically generated in the DBMS.

Page Layout

Every table is stored as an array of pages of a fixed size. The size of the page is 8 KB. In a table, since all pages are logically equivalent, the record can be saved in any page. Each page consists of 5 parts, and the layouts comprising the page are shown in TABLE 2 [9], [12].

TABLE 2
PAGE LAYOUT

Item	Description
Page Header	Include general information about page, and also had an unallocated space information. total 24 bytes.
Record identifier	Information about actual records location. 4 bytes per record.
Unallocated space	New record identifier is allocated from the start of this area, and actual new record is allocated from the end.
Record	The actual record itself.
Special space	For special space for specific data(optional)

The first 24 bytes of each page are the page header. The layouts comprising the page header are shown in TABLE 3.

TABLE 3
PAGE HEADER LAYOUT

Flag	Length	Description
pd_lsn	8 bytes	About next byte after last byte of xlog record for last change to this page
pd_checksum	2 bytes	Information of page checksum
pd_flags	2 bytes	Information of flag bits
pd_lower	2 bytes	About location. Offset for start of unallocated space
pd_upper	2 bytes	About location. Offset for end of unallocated space
pd_special	2 bytes	About location. Offset for start of special space
pd_pagesize_version	2 bytes	Information about page size and layout version
pd_prune_xid	4 bytes	For oldest unpruned XMAX on page.

The record identifiers are followed by the page header. Record identifiers require 4 bytes each. Record identifiers contain the start of the record, size, and certain attributes of the record. New record identifiers are allocated at the start of the unallocated space.

Unallocated space is followed by a record identifier. The unallocated space is where information is not stored.

The records are stored at the end of the unallocated space. The records themselves are where the contents entered by the user are stored.

The final section is the special section, which is generally not used.

Record Layout

All record structures are the same, and they contain fixed header sizes and other optional flag. The optional parts are t_infomask, t_hoff, and data entered by the user. Layouts comprising the header are shown in Table 4. The starting point of the data stored by the user is a byte that comes after the t_infomask2 value.

TABLE 4
RECORD HEADER LAYOUT

Flag	Length	Description
t_xmin	4 bytes	TransactionID stamp for insert
t_xmax	4 bytes	TransactionID stamp for delete
t_cid	4 bytes	CommandID stamp for insert and(or) delete
t_xvac	4 bytes	TransactionID for vacuum operation
t_ctid	6 bytes	current TID of this or newer row
t_infomask2	2 bytes	attributes number and various flag bits
t_infomask	2 bytes	various flag bits(optional)
t_hoff	1 byte	user data's offset(optional)

Change of Data file Triggered by Generating Record

There are four changes in page layout when record is generated. First, information of page header layout is changed. Second, record identifier is changed. Third, record data are added. Fourth, unallocated space is shrunken. Fig. 1 is figured about before the generating record.

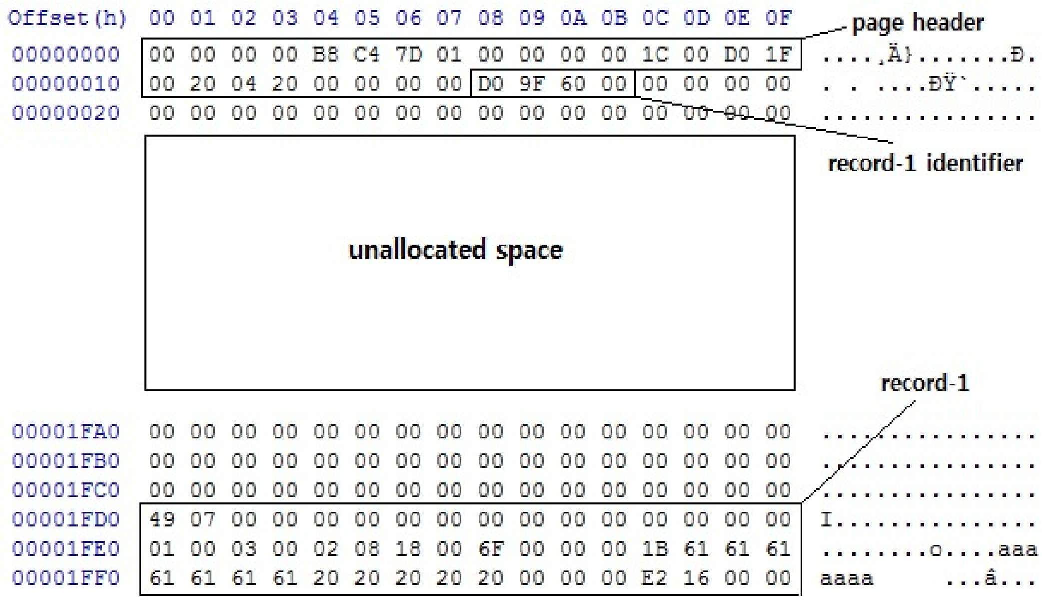

Fig. 1. Page layout

Fig. 2 is figured about after the generating record.

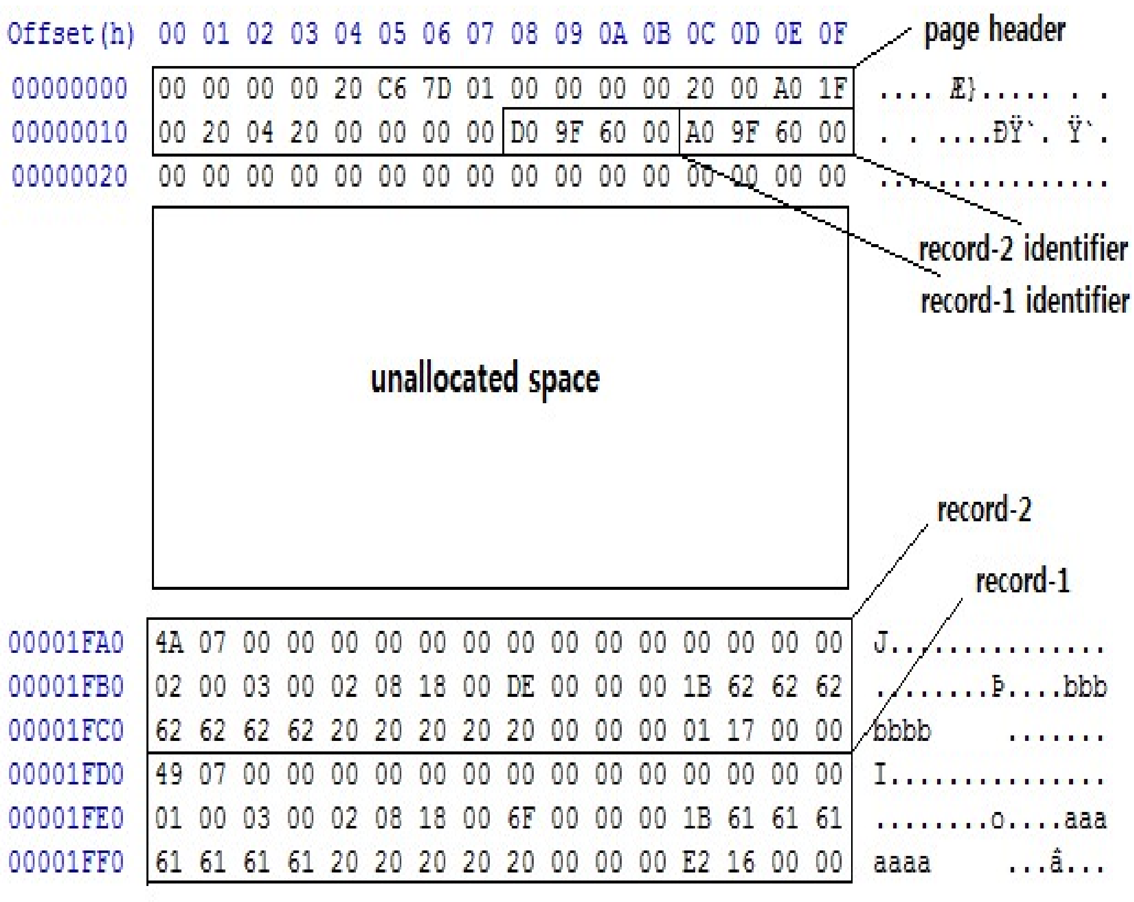

Fig. 2. Page layout after generating record

Change of Data file Triggered by Record Deletion

There are two changes in record header layout when record is deleted. First, t_xmax flag is changed. Before deletion, the value of t_xmax is '0x00 00 00 00'; afterwards, an unspecified value is allocated. Second, t_ctid flag is changed. According to the deletion test, the value of 4th byte from the t_ctid value changes from '0x00' to '0x20'; the value of 6th byte from the t_ctid value changes from unspecified value to unspecified value. Only, the value of 4th byte from the t_ctid value is changed regularly. Fig. 3 is figured about after the deleting record.

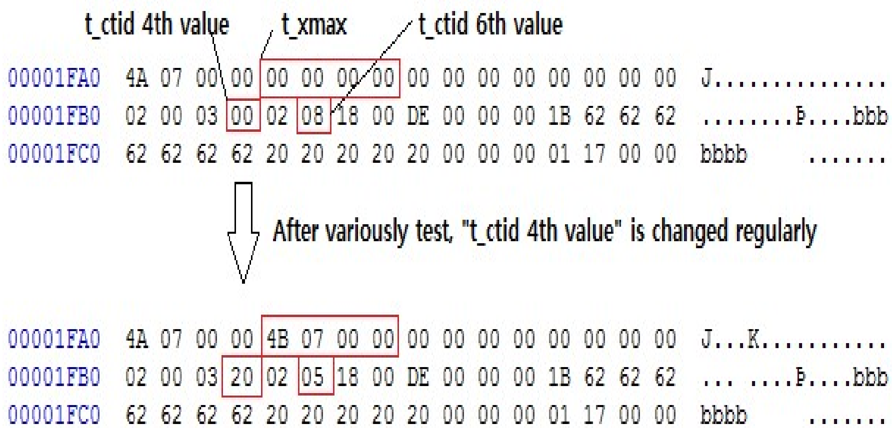

Fig. 3. After the deleting record

Instances, where the record header information changed according to performing the deletion test, were above three results. For all continuous tests, the 4th byte from the t_ctid value after the change remained fixed.

When distinguishing the deleted records, if the 4th byte from the t_ctid value was '0x00', it was not deleted, and if it was '0x20', it was found to be deleted.

Record Generation of after Deletion

When a new record is generated after the existing one is deleted, the contents entered by the user of the deleted record are not overwritten, and they remain without any changes. Deletion of the existing record and generation of a new record are shown in Fig. 4.

Fig. 4. Generated after the existing one is deleted

Deletion of a Database and Table

When deleting a database and table, the physical directory and the files targeted for deletion are deleted. Therefore, the records included in the targets for deletion cannot be restored. When deleting a table, the size of the physical file becomes 0 bytes; then, when restarting PostgreSQL, the physical file of the deleted table is deleted.

Database Optimization Test

Even if the records are deleted, a history of the records is kept. However, if VACUUM, which is a cleaning program that is offered as a basic program by PostgreSQL, is used, then the entire history of deleted records is also deleted [10].

RESULTS

When records are deleted, the value of t_xmax and t_ctid inside the record header changes. The value of t_xmax changes from '0x00 00 00 00' of 4 bytes to another value, which varies according to the value of the transaction. As for t_ctid value, the value of fourth bytes among values comprised of 6 bytes changes from '0x00' to '0x20'.

Recovery Algorithm for Deleted Records

Copy the deleted table file to the system to be recovered. Connect to PostgreSQL and check the structure of the table to be restored. The table file to be restored is divided into pages and copied to a separate file. Each page copied into a separate file is divided into records. Check the t_ctid flag of the record to separately collect records with the fourth byte value '0x20'. Combine the collected records with the table structure and convert the values of each column into ASCII form to complete recovery.

The deleted record recovery algorithm of PostgreSQL is shown in Fig. 5.

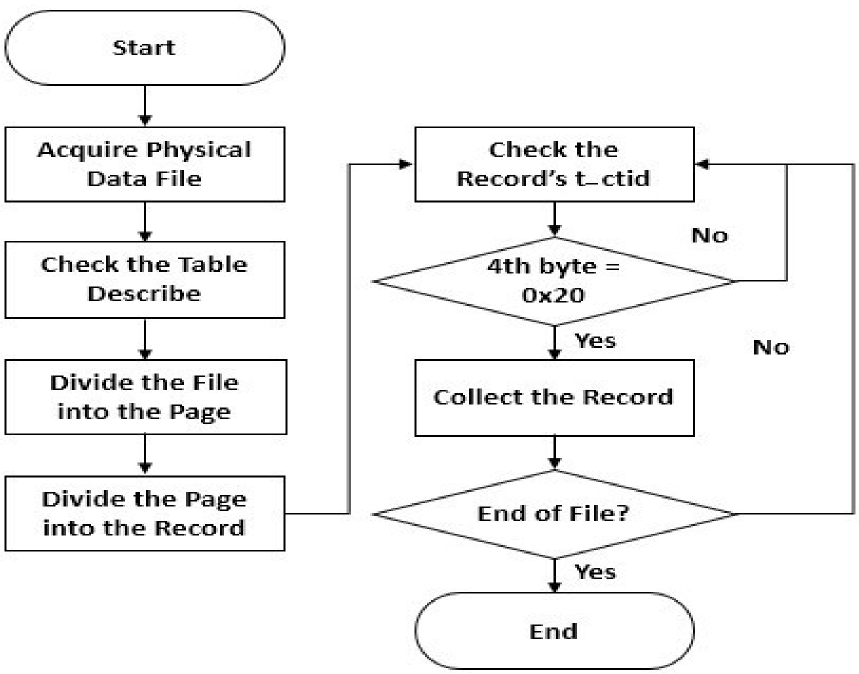

Fig. 5. Recovery algorithm for deleted records

DISCUSSION

The deleted record recovery method requires a PostgreSQL data file in which the structure of the table where the deleted record existed and the information of the deleted record exist [13,14]. Connect to PostgreSQL server, execute SQL command, and acquire table structure through it. To analyze the data file, copy the entire PostgreSQL data file to a computer that will perform the data recovery operation using a storage medium such as USB. The hex editor is used to check the contents of the data file with the records to be restored.

Open the PostgreSQL data file with a hex editor and check the header information of the record. If the value of the fourth byte of the t_ctid flag of the confirmed header information of the deleted record is changed to '0x20', the record is determined as the deleted record. When the record is deleted, since the value of the fourth byte of the t_ctid flag is changed consistently to '0x20', the t_ctid flag is defined as a criterion for determining the deleted record.

The record information determined as the deleted record is stored as a separate file. By comparing the record information stored in a separate file with the structure of the table, it is determined which column value the record value to be restored is.

For the readability of the record information, the record information expressed in hexadecimal notation is converted into the ASCII form and the recovery operation of the deleted record is completed. If you use VACUUM, a cleanup program for deleted records provided by PostgreSQL, it is impossible to recover records because deleted record information is deleted.

CONCLUSION AND RECOMMENDATIONS

As IT usage in enterprises increases, the usage of databases increases accordingly. The increased use of databases raises the possibility that database records will need to be used as evidence in crime situations. For a database record to be utilized as legal evidence, records associated with the crime should be presented to the court. However, database records can be deleted both routinely and deliberately, which makes it impossible to present them to the court. Therefore, in order to submit deleted records to the court as evidence, it is necessary to restore the deleted records. A number of studies have been carried out on this topic, but they have only targeted record recovery methods for certain DBMSs. Studies of record recovery methods for PostgreSQL have not yet been performed, despite of the world's fourth largest market share. This paper proposes a record recovery method for PostgreSQL. We proposed a method to recover deleted records when PostgreSQL records are illegally deleted or deleted normally but need to be recovered. In this study, it was possible to recover the deleted records of the PostgreSQL database in the damages caused by cybercrime. Recovered records can be used as legal evidence. This can be beneficial to companies in court situations. The advantage of being in a court situation helps protect and keep the property of the enterprise. And the results of this study can be used to recover records even if normally deleted records are needed. However, there are limits to the possibility of recovering deleted records in some cases already mentioned.

Future work will be dedicated to studying for another DBMS that has not yet been studied. And we will also build a recovery automation solution if the environment is given.

Declaration of Conflicting Interests

This is the original work formulated and executed by the authors. It is not being processed or published elsewhere and there are no identifiable conflicts of interest.

REFERENCES

[1] J. H. Choi, D. W. Jeong and S. Lee, "The method of recovery for deleted record in oracle database," *Journal of the Korea Institute of Information Security and Cryptology,* vol. 23, no. 5, pp. 947-955, 2013.

[2] K. Lee, D. Jeong, C. Kang and S. Lee, "The method of recovery for deleted record in the DB2 database," *Journal of Digital Forensics,* vol. 8, no. 2, pp. 1-14, 2014.

[3] S. M. Jang, "Deleted records recovery research for MySQL innoDB," Unpublished dissertation, Graduate School of Information Security, Korea University, Seoul, South Korea, 2014.

[4] S. Y. Park, "A research for record recovery method in database," Graduate School of Information Security, Korea University, Seoul, South Korea, 2013.

[5] M. Stonebraker and L. A. Rowe, "The design of Postgres," in *Proceedings of the International Conference on Management of Data,* Washington, D.C., WA, pp. 340-355, May 28-30, 1986.

[6] PostgreSQL. (2017). *PostgreSQL 10 RC 1* [Online]. Available: https://www.postgresql.org/

[7] Federico Campoli. (2015). *PostgreSQL database administration* [Online]. Available: https://goo.gl/7fHzbA

[8] J. S. Katz and J. Mlodgenski. (2013). *A tour of postgreSQL data types* [Online]. Available: https://goo.gl/egHAqk

[9] B. Momjian. (2017). *PostgreSQL internals through pictures* [Online]. Available: https://goo.gl/fcG6cr

[10] B. Momjian. (2017). *Mastering postgreSQL administration* [Online]. Available: https://goo.gl/Ax6GC2

[11] S. Christensen. (2010). *Postgres administration for sysadmins* [Online]. Available: https://goo.gl/h8zvr5

[12] EnterpriseDB. (n.d.). *Introduction to postgreSQL administration* [Online]. Available: https://goo.gl/YAHw8H

[13] Kim, J., Park, A. and Lee, S. "Recovery method of deleted records and tables from ESE database," *Digital Investigation*, vol. 18, pp. 118-124, 2016.

[14] Liu, X., Fu, X. and Sun, G. "Recovery of deleted record for SQLite3 database". In *8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Vol. 2, pp. 183-187, 2016.

— This article does not have any appendix. —