MySQL Record Carving

Esan Wit[1] Leendert van Duijn[1]
Supervisor: Kevin Jonkers[2]

[1]SNE University of Amsterdam

[2]Fox-IT

February 5, 2014

Introduction

Why:

- No easy option available
- Recover deleted records
- Recover damaged or corrupted records
- Mailing lists said that it was very difficult to recover any sensible data
    - Challenge accepted

Existing Work

- Databases leave data behind on deletion/updates, Stahlberg et al.
- Recovery from SQLite, Pooters et al.
  - Template matching
- Recovery from alternative sources, Frühwirt et al.
  - Logs, replication files, temporary tables
- Percona LLC has a tool for data recovery from InnoDB tables
  - Template matching
  - Tricky setup
  - Finds many false positives
- Proving database integrity via index properties, Kieseberg et al.

Goal

To recover deleted records from MySQL.

- What data remains after deletion of a record?
- What methods exist for recovering, parts of, this data?
- Can this be extended to cover more general damage or different databases?
- How do the differences between database systems relate to record recovery?

Basic layout

Record 1		Record 2	Record
3	Record 4	Record 5	
Deleted record		Record 7	
...			

Table: Generic layout of records

- MyISAM whole file
- InnoDB leaf pages only

Deleting a record

```
00 00 00 00 03 00 63 01  08 fe 02 00 00 00 12 4c  |......c........L|
65 65 6e 64 65 72 74 20  76 61 6e 20 44 75 69 6a  |eendert van Duij|
6e 18 4c 65 65 6e 64 65  72 74 2e 76 61 6e 44 75  |n.Leendert.vanDu|
69 6a 6e 40 6f 73 33 2e  6e 6c 28 35 62 61 61 36  |ijn@os3.nl(5baa6|
```

Original

```
00 00 00 00 00 00 00 68  ff ff ff ff ff ff ff ff  |.......h........|
ff ff ff ff ff ff ff ff  76 61 6e 20 44 75 69 6a  |........van Duij|
6e 18 4c 65 65 6e 64 65  72 74 2e 76 61 6e 44 75  |n.Leendert.vanDu|
69 6a 6e 40 6f 73 33 2e  6e 6c 28 35 62 61 61 36  |ijn@os3.nl(5baa6|
```

After deletion

Fragment of MyISAM data file

Deleting a record cont.

```
64 52 e6 55 ff 00 00 00  00 80 28 18 12 00 00 00  |dR.U......(.....|
18 00 75 00 00 00 02 00  00 00 00 0b 09 87 00 00  |..u.............|
01 3b 01 1c 4c 65 65 6e  64 65 72 74 20 76 61 6e  |.;..Leendert van|
20 44 75 69 6a 6e 4c 65  65 6e 64 65 72 74 2e 76  | DuijnLeendert.v|
```

Original

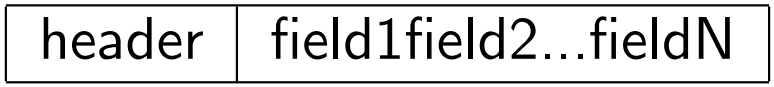
```
64 52 e6 55 ff 00 00 00  00 80 28 18 12 00 20 00  |dR.U......(... .|
18 00 00 00 00 00 02 00  00 00 00 0b 0a 08 00 00  |................|
01 3c 01 10 4c 65 65 6e  64 65 72 74 20 76 61 6e  |.<..Leendert van|
20 44 75 69 6a 6e 4c 65  65 6e 64 65 72 74 2e 76  | DuijnLeendert.v|
```

After deletion

Fragment of InnoDB data file

Record format

All records have a similar structure:



Header content and length depend on various properties:

- Record state
- Engine, row format
- Null fields
- Variable length fields

Template matching

- Attempt to parse data
- Be careful and strict
- Any misalignment will ‘corrupt’ output
- Some types lack visible boundaries

Template matching

- Sliding window template matching
- Data type validation
- Extensible data validation

Basic algorithm

```
for location in candidates
   for template in templates
      record = parse(location, template)
      if(success(record))
         score = validate(record)
         if(score > threshold)
            results.add(record)
            candidates.add(suggestion(record.length))
```

Early validators

An early validator can do elimination, when no full record is available yet e.g.

- Enum is invalid (i.e. it's value is out of bounds)
- A string contains invalid characters according to its character set
- A record(header) is (not) marked as DELETED
- A record(header) is marked as RESERVED

Correct template design and strict early validators can limit false positives, while some match any position found.

Row validators

A validator scores a row, based on field content e.g.

- String field `Name` contains common english letters
- String field `Email` looks like a valid email address
- Timestamp field `Last login` is within a realistic range
- Field `Last login` > `Registration date`

Results and observations

Simple test on InnoDB (98306 matches attempted, 12 matched)

	Present	Deleted	False pos.	False neg.	Total
Validated	1	6	0	0	7
Expected	1	6	0	0	7

InnoDB world (507922 matches attempted, 9193 matched)

	Present	Deleted	False pos.	False neg.	Total
Validated	4042	100	88	25	4142
Expected	3979	100	0	0	4079

- Using validators we were able to reduce false positives.
- Template and validator quality determines overall effectiveness

Features of PoC

- Currently supported engines
  - InnoDB Antelope file format, Compact row format.
  - InnoDB Antelope file format, Redundant row format.
  - MyISAM Fixed row format
  - MyISAM Dynamic row format
- Early field level validation
- Row level validation
- Many different field types
  - datetime, decimal, varchar, set, etc.

Conclusion

- After executing a DELETE data remains partially recoverable
- Template matching can be used to recover this data
- Validation can be a powerful tool in eliminating false positives

Future work

- Other database systems
- Generic templates
- Split records
- Validators
- Machine learning

Questions?

Sources


Oracle, *MySQL 5.6 Reference Manual*, `https://dev.mysql.com/doc/refman/5.6/en/`, 2014-01-25 (revision: 37524)


Percona LLC, *Percona Data Recovery Tool for InnoDB*, `http://www.percona.com/software/mysql-innodb-data-recovery-tools`


Ivo Pooters and Pascal Arends and Steffen Moorrees, *Extracting SQLite records - Carving, parsing and matching*, 2011,


Peter Frühwirt and Peter Kieseberg and Sebastian Schrittwieser and Markus Huber and Edgar Weippl, *InnoDB database forensics: Enhanced reconstruction of data manipulation queries from redo logs* , Information Security Technical Report, 2013


Stahlberg, Patrick and Miklau, Gerome and Levine, Brian Neil, *Threats to Privacy in the Forensic Analysis of Database Systems*, ACM, 2007


Kieseberg, P. and Schrittwieser, S. and Mulazzani, M. and Huber, M. and Weippl, E., *Trees Cannot Lie: Using Data Structures for Forensics Purposes*, 2011