Intro to Database Forensics

For Novalug

ZAK ZEBROWSKI

ZAKZ@ZAKZ.INFO

License

About Me

Zak Zebrowski

Author of https://github.com/japharl/web-identity

Forensic Database Engineer / Web Identity Specialist / Perl Guy / Raspberry Pi Kite Photography Guy

Course Outline

Introduction to databases

Introduction to forensics

Definitions

Converting Data to Unicode (which you should understand before)

Converting from (various database systems) to a common database system (Maria DB).

Questions

Notes

https://github.com/japharl/forensic-database-analysis

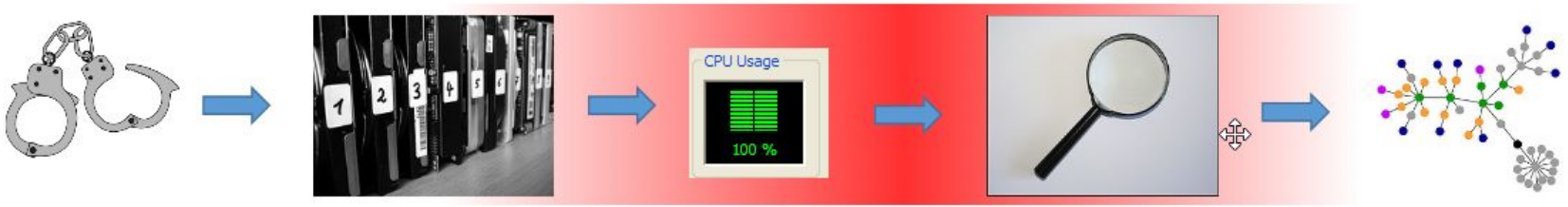
Note - these slides have been simplified as the original presentation was an all day course. :) Please see original in github location for full presentation.

Optional: Lab Install

1. Start a standard Ubuntu install.
   - Or use wsl ubuntu.
2. Git clone https://github.com/japharl/forensic-database-analysis.git
3. Open a command terminal and run as root .
4. bash –x /install.sh from the lab folder.
5. Note - wsl requires you to start the mysql service manually.

What is forensic database analysis?

- Forensic database analysis is a specialty of knowing a bunch of techniques to allow analysts to query databases, which were recovered in a forensic manner.
  - We receive evidence via appropriate legal authorities. Then, we mount and host a copy of the recovered database, convert that database to Maria DB (if it is not), while converting the data to Unicode, then write tools to query the database, based upon priority queries defined by analysts.



Introduction to Forensic Mindset

Like CSI related show.

But a lot more rules and regulations.

⚠️ Just because you have the idea from tv does not mean that you should do it.

⚠️ You need to ensure you have the appropriate legal authorities to work on the subject.

⚠️ Always document what the original sources are, who gave it to you, when, and track it as you go through your procedures as best you can so that when you are asked, if your case goes to court, the piece of media that the data came on and the legal restrictions surrounding that data.

Legal Considerations

- We always want to work on a copy of the data. Never work on the original evidential copy, unless there is an immediate need (life & death).
- Always note where data is from!
  - Different data sources have different restrictions, which may restrict how you use data in the court.
  - The court may not accept the evidence for various reasons.
    - An unknown data source
    - The source data was collected too long ago due to data retention rules.

Ask your appropriate authority if you have questions. IANAL.

Maria DB

I will use Maria DB / MySQL almost interchangeably throughout the course. For us they are "close enough" to being the same. This database is fast and has a lot of good features.

We will tend to use the "myisam" table type primarily. This *tends* to be faster than other table types. There are some gotchas with this table type (namely that it's not transactional and not fully recoverable) but are easier (and more prevalent) than other table types (in real life).

We will not use the full text index within Maria DB, as it does "automatic" things that we don't want, such as silently creating a stop word list. We will use sqlite3 full table index instead, which is an additional step, but is more accurate.

- Maria DB will be used for larger data collections, so that multiple machines can use it as it is being upgraded.
- Sqlite 3 is an additional database engine. This one is more of an "inline memory" database, often used in core os functions (on ios, macos, android, and sometimes in windows). See https://sqlite.org/index.html for general information. It's a great tool as well.

Steps of Analysis

1. Documenting the source
2. Convert to Unicode
3. Convert to Mariadb
4. Make the data searchable.

Steps Caveat

One small gotcha with the steps…

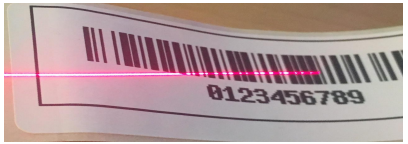
The steps are notional.

- You will need to be aware of Unicode / character set issues while you are importing data into a database. Sometimes you need to convert before a database is loaded (via a flag specifying a character set), sometimes after, and sometimes in the middle of things…
- Sometimes you will be required to mount data of forensically obtained hard drives. Sometimes you may receive a database "dump" of information (which may not have information about how a database was configured to work).
- There are multiple techniques to do certain things. Decide the best one for your work case, and expected queries, but be aware of the tradeoffs.

Step 1 - Documenting the data source

Start by noting where the drives come from.

Clearly document:

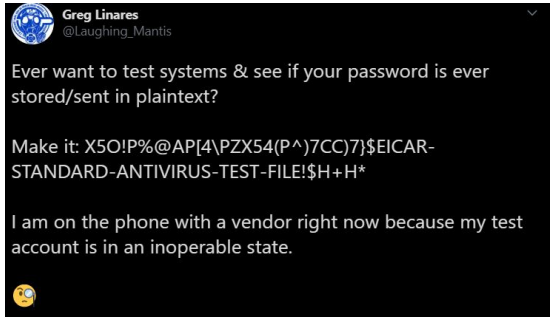


- When the drive arrived
- Why the drive arrived (what case / what is the need / arrived from...)
- Who gave you the drive (in case there are questions)
- What the drive (should) contain
  - You always verify (if possible) contains the correct data early in the process.

Comment on Antivirus

Depending on your data set, don't automatically apply antivirus.

- Users (who contribute to these criminality contained websites), *may* be contributing files that have viruses which may be critical to your case.
- During initial database load, ensure that you allow it to load regardless of whether it has viruses or not.
- After the data load protect the end user from viruses but allow them to access the virus infected files if they need it and are appropriately protected.
- Twitter
  source: https://twitter.com/Laughing_Mantis/status/1308212643723767809



Step 2 - Convert to Unicode

Let’s talk about character sets!

Computers have no knowledge of letters, only 1's and 0's.

Character sets are a convention to use a specific sequence as a character representation in your locale. Examples are on next slide. Node that each character set name can have many aliases that were used by different companies.

Unicode is an attempt at universal character set for multiple languages

- This includes, most of the individual character sets, plus additional languages that historically did not have code pages such as American Indian, and other uncommon languages around the world, and smiles (😁 🦞)...
- See: See history and more information here: https://en.wikipedia.org/wiki/UTF-8

Unicode allows databases to optimize for one character set and allows it to be more efficient.

- Especially when some languages have more than one character set for the given language.

Different Character Sets Example

Character Set Name (Some Aliases)	Character Text (Not decoded)	Interpreted Text
Latin1 (ISO-8859-1, IBM819, cp819)	Good Morning.	Good Morning.
Cp1251 (windows-1521, IBM-5347)	Äîáðîå óòðî	Доброе утро
Koi8r (ibm-878, windows-20866, cp878)	äÏÂÒÏÅ ÕÔÒÏ	Доброе утро
Koi8u (windows-21866, ibm1168)	äÏÂÒÏÅ ÕÔÒÏ	Доброе утро
Cp866 (windows-866, cp866, ibm866)	„®¡à®¥ ãâà®	Доброе утро
ISO 8859-5 (windows-28595, Cyrillic, ibm-915)	´ÞÑàÞÕ ãâàÞ	Доброе утро
Utf8	Доброе утро	Доброе утро

Corrupted Character Sets

How do character sets get corrupted?

- Databases / scripts will silently handle incorrect settings, but the app will still work.
- (Criminal) admins only need to get settings right once.
- Browsers are forgiving but searching across multiple (intentionally) corrupted datasets is hard.
- Character sets can be set in multiple places, so it's hard to catch unless you are aware of what's happening.
  - In the (local app you are using)
  - In the database
    - For the connection between the application and the database
    - For the database (as a whole)
    - For a table (individually)
    - And you can call convert commands on any of the above.

If you are going to systematically exploit multiple databases, you need to get it right.

Corrupted Character Set Examples

Character Set in Database (utf8)	How Corrupted?	How to fix.
´ÞÑàÞÕ ãâàÞ	ISO 8859-5 not converted to utf8	Connect to database as utf8 Convert text from ISO to utf8. (May need to turn off utf8 flag) Store in utf8 database
Ð”Ð¾Ð±Ñ€Ð¾Ð¼ NƒN,Ñ€Ð¾	Utf8 inserted without the database flag set for utf8 information. The D character is a *good* indicator that this happened.	Connect to database without setting a utf8 connection. Select all rows and turn utf8 flag on in new database connection. Store in new database.

You don't need to know how to fix this now, just know that it happens.

Lab 1

Let’s take a validly formatted utf8 document, covert it to another character set, and examine it.

- This is just so you can see a non-utf8 character set in the “wild”.

```
cd /database_forensics/lab1

cat note.txt

perl utf8_to_other.pl cp1251

cat cp1251.txt
```

Open cp1251.txt and note.txt in Firefox.

- Start Firefox
- Drag both files individually to Firefox window.

Lab 1 Questions

Why did the terminal display note.txt correctly but not cp1251.txt ?

Why did the browser correctly display note.txt and cp1251.txt?

Lab 1 Answers

Why did the terminal display note.txt correctly but not cp1251.txt ?

- The terminal is only configured to a specific character set, namely utf8. You can set the terminal manually to a different setting, but then other character sets will not be viewable.

Why did the browser correctly display note.txt and cp1251.txt?

- The browser *has* to be able to view multiple character sets, often there are no header identifying (or perhaps incorrectly identifying) the character set on a web page, so it uses heuristics to determine the character set.

Caveat in character set processing: Utf8 vs utf16le or utf16be

There is a minor caveat with this Unicode goodness.

Each alphanumeric character for *most* languages are contained with 8 bits.

However, some languages (or language extensions) are actually *16* bits. Which decreases the amount of free space within a table structure for possible rows.

- In particular, you might want to store the larger size is support of smile characters.

So, depending on how you need to optimize your database, you may (or may not) want to interpret your Unicode data.

Detecting what character set data is in.

It can be straightforward to very hard. Let's consider a few cases:

- The data can all be in one dataset (say cp1251).
- The data can be in multiple character sets (say cp1251 and koi8r).
- The data can be in Unicode as multiple character sets (say a utf8 encoded string, with entries in cp1251 and koi8r) # Harder.
- The data can be in cp1251 data set but entered as koi8r. # Server misconfigurations are fun!
- The data can be in Unicode, but not handled as Unicode. # Sort of easier to approach.

How to approach?

Time, patience, and someone who is an expert in the language to assist to validate correctness*.

* Or if no one is available, run a processed text entry through machine translation. The translation does *not* need to be correct.

Lab 2

For each of the files, what character sets are each file in? Use the techniques below:

```
cd .. # assuming you are in lab1 directory
cd lab2
cat file1.txt
cat file2.txt
cat file3.txt
chardet file3.txt
firefox file1.txt # And use Firefox technique from before to identify the character set.
```

What are the character sets for each of those files?

Convert.pl & Encode::Guess

Convert.pl is a perl script I wrote, designed to copy a Maria DB database from one host to another (or the same) host, while converting character sets from native to “correct” character sets.

- “correct” is in quotes because there is still guessing involved.

The script takes for each table

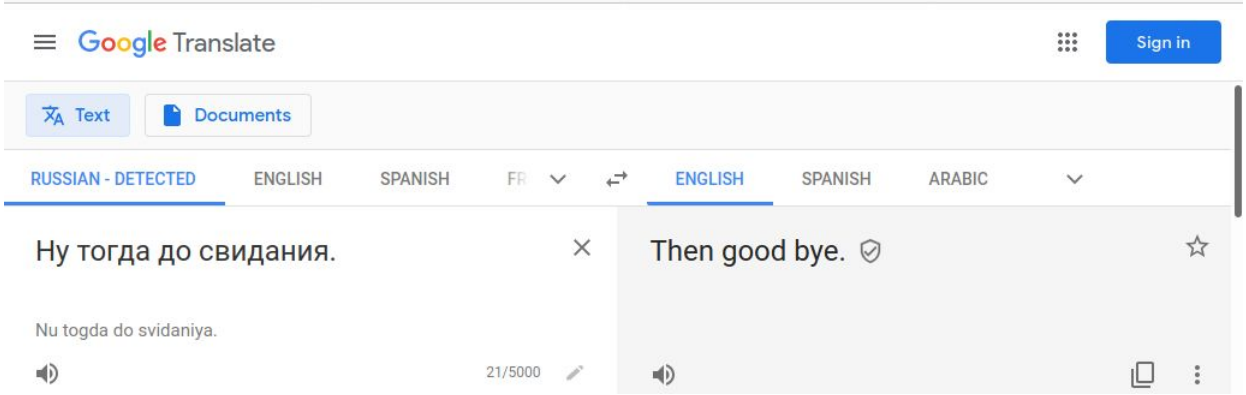
- For each row
  - For each column
    - Converts data converted from Encode::Guess (or a specific character set) to the correct character set encoding.
    - The magic is in *how* you connect and *what* you connect too, as the database and the database connection can have different character sets.
      - A database may only be in a certain character set. You can connect as that character set to the database and then convert it to Unicode.
      - Or you connect as Unicode and then translate the data in cp1251 to unicode, etc.

Located in /database_forensics/convert/

Is your new encoding correct?

Run through google translate (or ask a colleague) if you don't understand what is on the screen.

- If it's an incorrect character set, there will be no translation.
- If it's correct, there may be a complete or partial translation.



Step 3 - Converting from (native) to hosting the data in Maria DB

We'll now talk about converting from (native database type) to Maria DB.

Why not keep it native (original format)?

- You could. *But* there could be other considerations.
  - Can you boot from original media / copy of the original media?
  - Are you approved to run the OS that they were running (long term)?
  - Do you need to license the OS / database software that may not have been licensed originally?

Why Mari aDB? - Fast, Cheap & reasonably Optimized.

Why not Oracle™ or Microsoft SQL Server™? – Licenses are expensive.

Hardware comment

Strongly recommend solid state drives.

- Less susceptible to errors processing data.
- Faster.
- Reduce month & ½ timelines to 1 week for large data sets that need processing.

Recommend higher capacity memory servers

- More connections permitted to the database instances
- Faster insert times.

Caveats in the data conversion process

There will be pain points.

Remember, you have *no* control of the data.
- You can't ask for advice why some settings were used.

The data can come in fast and furious, or slow

The data can come in many formats

⚠️The data may have viruses.
- **Do not run antivirus on the data until it's been processed.**
- **You need to ensure the database gets loaded first.**
- **Then run antivirus on the rows of the database.**
- ⚠️**Be sure you are on an isolated network.**

You want to have a separate system to process the databases than the dev and production system.
- The "working" system can have various tables overwritten in the restoration process.

Due to (strange) errors that have happened on Windows, strongly recommend using Linux instead.

Table Engines

Maria DB has multiple engine types, that have different trade offs.

- Innodb – transactional tables, but slow to insert. **Default engine in recent (> 2010) databases.**
- MyISAM – non-transactional tables, slower to insert, but fast to select. **Default prior to 2010.**
- Aria – transactional tables, faster than innodb, but slower than myisam.
- CSV – csv files, but non-indexable.

We will use MyISAM table engine primarily, then use other table engines sparingly.

- It depends on what you need to do.
- AriaDB is nice for new projects where you do the collection.

Full text engine caveat

But mysiam / other MySQL table engines does not have a great full text search engine, which (unfortunately) are largely not configurable.

An outside search engine is generally preferred. Two recommendations:

- SQLite
  - Just works!
  - Generally approved software, as it can be proven that macos runs it under the hood for searching.
- Apache Lucy
  - A Perl based Lucene like full text index, which I am transitioning away from.
  - There's nothing inherently wrong with it, in fact, I've had great success with it. Allows you to configure everything and Unicode is correctly supported. However, (for unknown reasons) Apache Lucy was deprecated, so instead we are favoring SQLite
- Maria DB's FULLTEXT table(col)
  - Yeah, it's not ideal, but if you can't do external searches for full text indexes, you go with what you have.

🛑 General recommendation before we get too far!

Before you start loading files, run the script MySQL Tuner

Available from here : git clone https://github.com/major/MySQLTuner-perl

To run:

- cd ../lab3/MySQLTurner-perl/
- sudo perl mysqltuner.pl
- And output will be displayed to the screen, with recommendations for your current machine.
- It will say:
  - Variables to adjust:
  - Query_cache_size=0
- Etc. *and* consider advice from above that line.
- Do as it suggest (unless there is a good reason not to) for a more optimal server.
- cd /etc/mysql/mariadb.conf.d/
- vi 50-server.cnf
- Then: service mariadb restart
  - To apply the changes.

Also, while talking of /etc/mysql/mariadb.conf…

General recommendation of modifying utf8mb4 character set default to utf8 in the various configuration files.

- Why?
  - UTF8mb4 takes more space to store characters than utf8.
  - Only really makes a difference if you care about smile characters. If you don't, there is no reason to keep it.
- Change the files in /etc/mysql/mariadb.conf/*

Mini lab 3 & Questions?

We’re going to just run the mysql performance tuner to see what it says.

cd .. # assuming you are in lab2 directory

cd .MySQLTuner-perl

perl mysqltuner.pl

Then browse to /etc/mysql and look at some files there.

Moving data from (native) to Maria DB, relative pain

MySQL database “dump” files

Perl DBI

CSV / XLSX

“Cold” Mysql MyISAM tables

ODBC

XML / JSON / YAML

Access

Encrypted Microsoft SQL Server 2005

Non-standard database systems

Oracle

Alternatives

- Assuming $$ on projects.
- When all else fails

Mysql / Maria DB dump files

Dump is a general search term, for a complete copy of a database, with table definitions, stored procedures, views and data.

Generally, you can get away with doing a:

- mysql < data.sql

But there are some caveats.

- If the file contains InnoDB engine type (due to the load time), change it to MyISAM manually in a text editor like vi.
  - ⚠️Why manually and not a regexp? There is no prohibition in a database from the term InnoDB in a database table. So, if you do a full regexp replace, you therefore may be unintentionally changing the meaning of the ***data*** within the database!
- The data may be in the incorrect character type. You may need to specify a character set on the command line to manually get it to load correctly. Mysql will give you an error if you try to insert without correct character type set.
  - mysql --default-character-set=cp1251 < ./data.sql
- You may get duplicate key inserts. Depending on the row, consider deleting the row. 😱
- If it *still* doesn't load correctly, look at using the convert.pl script to manually deal with character sets.

Continued…

What happens if I can’t connect to the database?

Sometimes the database dump files rewrites data in the “mysql” database, which contains the permissions files.

In /etc/mysql/mariadb/50-server.conf , add the text skip-grant-tables

⚠️ This is obviously a security issue, revert when possible and restart the server.

You should be able to connect as root with no password to any database on the server.

Perl DBI

Perl is a computer language I use, a lot, to help with databases.

- See day 0 presentation if you need a perl refresher. While the point of this class is not to cover the basics of perl, if you've never seen it, it's a place to start.

Perl is *especially* valuable for dealing with multiple databases and character sets / *corrupted* character sets.

DBI is *sort* of an ODBC like environment, which is a general way to connect to a database system (if you have the necessary driver installed correctly).

In addition to mysql, there are drivers for csv (files), SQLite™, postgres™, DB2™, RAM (memory), JDBC, LDAP, ODBC, etc.

Code on next slide to see how to export / import a table.

Perl Code

```
#!/usr/bin/perl

use strict;

use warnings;

use DBI;

my $dbh_from =
DBI->connect(“dbi:SQLite:dbname=toimport.db
”,””,””,{RaiseError=>1}) or die “$!”;

my $dbh_to =
DBI->connect(“dbi:mysql:database=test”,unde
f,undef,{RaiseError=>1}) or die “$!”;

my $sql0=qq{select row,value from table};

my $sth0 = $dbh_from->prepare($sql0);

$sth0->execute();

my $v,$r;

$sth0->bind_columns(undef,\$r,\$v);

my $sql1=qq{insert into table(row,value) values
(?,?)};

my $sth1= $dbh_to->prepare($sql1);

while ($sth0->fetch()){

  $sth1->execute($r,$v);

}
```

CSV / XLS / XLSX

Perl also has a library for these data types. While they do have a bunch of options, they can be straight forward to import and export valid data.

CSV -> Recommended module: Text::CSV_XS

- I actually *don't* recommend the DBD driver for CSV. Text::CSV_XS is faster and can handle Microsoft exports better.

XLSX / XLS -> Spreadsheet::XLSX can read in spreadsheets.

Common issues:

- Spreadsheets are a *bit* complicated normally, possible multiple pages, etc. Manually review the spreadsheet before import.
- Microsoft (in some versions) doesn't handle character sets correctly, need to verify export data looks correct, and possibly reprocess natively.

ODBC Discussion

ODBC (which is a windows standard) is a way of connecting / importing / exporting data, which is like JDBC (for java) and DBI (for perl).

Once you get an application running on Windows, which has an ODBC adapter, you can configure ODBC to export the data and put it into another format or database (that may be easier to use, such as csv, and then import it into mysql.)

- Access, SQL Server, Maria DB, and other databases all have ODBC types.

⚠️Note that there are two different flavors of ODBC drivers, and settings within window. Win32 and Win64. Both are executables with the same name, odbcad32.exe. The 64 bit version is located in the system32 directory, but the 32 bit version is in syswow64 direction. 😐 You will need to use the correct one for the correct driver and use a code system compiled the same way to access the data.

- Eg. Access 32 bit needs a odbc driver that's 32 bit, and the perl code to extract data from ODBC needs to be a 32 bit flavor.
  - (Strawberry Perl (http://strawberryperl.com/) is the preferred windows perl version, offered in 32 and 64 bit varieties.)

Cold MyISAM Tables

Cold – in this case – means from raw file dump of /var/lib/mysql/database_name/ contents, which are table.myd table.myi and possibly other files.

Do this:

- service mysql stop
- cp –R source/ /var/lib/mysql/source
- chown –R mysql:mysql /var/lib/mysql/source
- chmod –R 777 /var/lib/mysql/source
- service mysql start

Then

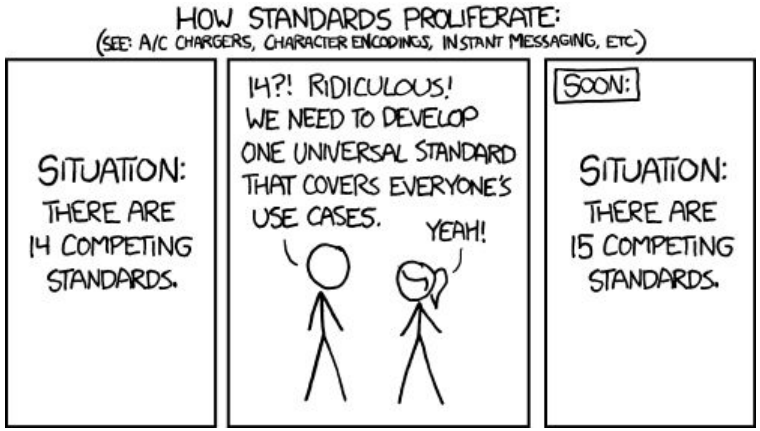
- Note errors in /var/log/mysql/*
  - These may be permission or corrupted tables or similar. Note & fix as possible.
- Repair tables as necessary. # Discussed soon.
- Possibly run convert.pl to fix character set issues.

Back to processing: XML / JSON / YAML

XML / JSON /YAML are standard data formats that occasionally need to be treated as a database.

The “best” way to handle these, are to write programs against the libraries which correctly handle data standards.

It’s not easy. But once you get it working right once, then you can apply updates in a reasonable straight forward way.



https://xkcd.com/927/

Microsoft Access

Microsoft Access is a special case.

The ODBC driver works, but, if you have Unicode in the database, the utf8 flag is *not* set. This means, the strings that are returned are *not* utf8.

- Export the data to csv.
- Or, use other library techniques to convert non-Unicode to Unicode.

Weirdly, there are other considerations as well:

- Auto incremented integers may not export correctly.
- Floats may not export correctly either.

Cold InnoDB

Cold InnoDB files are also a special case.

InnoDB are transactional tables, which are the default in recent versions of Maria DB.

When you get these files, these are non-trivial to recover.

InnoDB will only work with the *exact sub-build* of Maria DB or mysql that was used originally. 😐

- Sometimes, occasionally, they will be compatible between sub-builds, but often not.

There are special innodb-recovery flags available to set in server.conf which you can adjust, which should allow for data selection out of tables. (See next slide).

At that point where you have set the recovery flag, just select data out of the database, eg:

```
mysqldump –uroot myData > myData.sql
```

Cold InnoDB Continued

To test a version (on a non-production Linux instance):

- `service mysql stop`
- `cp ./ib_* /var/lib/mysql/`
- `chmod –R 777 /var/lib/mysql/*`
- `chown –R mysql:mysql /var/lib/mysql`
- `service mysql restart`
- `tail /var/log/mysql/error.log`
- `tail /var/log/syslog`

- If it doesn't start, consider adding to /etc/mysql/maridb.conf/50-server.conf
  - `Innodb_force_recovery=7`
    - ⚠ **Danger** – this is sort of saying "damn the torpedo's full speed ahead!". An overkill for servers you want to have running in the future, but ok, for our needs.
      - https://en.wikipedia.org/wiki/Battle_of_Mobile_Bay#%22Damn_the_torpedoes%22

Encrypted Microsoft SQL Server 2005

Not easy. This is a very hard.

SQL Server 2005 is a common in communities.

One would initially think that they could use the “redo” logs to reload the database tables correctly, and then select information out of the restored tables. (Where the redo logs were exported via encase or similar.)

Unfortunately, this will *not* work for information in encrypted tables. The encryption key is stored in the *boot sector* of the server.

The solution is to completely restore the server to hard drive via dd, and then boot that server.

Change the administrator password via a windows utility.

Boot the server, and let the server identify the hardware for the “new” system. (This may take a while).

- Attempt to ensure that the hardware is reliable.

Extract the data contained in the server decrypted via ODBC to a Maria DB server.

Non-standard database systems

Not easy. This is a very hard.

Try to export to CSV or EXCEL or any other data system that you do have a method to import with.

Alternative to previous approaches

Commercial solutions

- Allow you to do a lot of magical things with the databases, however, you may have limited ability to update known bad records or corrupted records.
- There are many different providers at different price points. Lower price points generally require you have more knowledge about database queries. Higher priced ones generally handle corrupted data better or has ways to fix it for you.

However generally they often do not do what we want.

- They don't handle the character sets we have the data in
- They aren't context aware that custom tools can be
- They aren't geared for finding the needle in the haystack, rather, looking at combining multiple haystacks into a single haystack.
  - Which, when companies merge, are a valid use case for a commercial tool.

Step 4 - Make it searchable

Let’s talk about data processing & search prioritization for analysts.

- Keep in mind that analysts will not see any of the previous work. Nor will they see this work. They will only see what is available in the actual application they receive that you will write separately from this.
- If you have a new huge data set, what do you want to prioritize to get data to the analysts?

Generalization caveats & Recommendations

Dealing with deleted data.

Murphy caveats.

Sanity Checking.

What to prioritize?

- Large tables – they generally signify something important.
- Small tables – see if you can glimpse it and understand the meaning… particularly code or other similar tables. But maybe not focus as much time on it.
- Tables that answer questions
  - Eg. With the current data can you answer questions?
- Everything – if you just can't make sense of it. (Special case!)
- *Listen* to the analysts to see what their needs are.

- Also - Deleted rows
  - Because mysql table types initially didn't contain ways to cascade delete information, it is often quite possible to recover "deleted" users from conversations.

Prioritization Continued

Extract any priority fields only and import them into a new database.

- Store or just note where the secondary information is in a full text index of some sort.

Index the data as the users have requested for what their priority queries are.

- Listen to your analsyts!

Don’t be afraid to create computed indexes…

- Soundex, lower case only, etc.

Generalizing

⚠️ Be careful to say no to the temptation to over generalize data from the start.

- You may generalize the wrong thing.
- You may generalize on the details that are **not** needed.

If it's a one off, just write display code, then search code (or use a generic search described later).

If it's not a one off, process in the most generic way possible.

- Have a database of databases to describe the data you need to process.
- Also have a database to store results.
- Given that database of databases, for each database process each "type" of databases uniformly, and store that processed data in the database of results.
  - Then, if possible, try supporting multiple versions of a database type via try / catch methodology.
- Next, index and further process the stored results database and write common tools against that.

Dealing with deleted data

One of the interesting things about (certain database systems like mysql), rows are not automagically cascaded deleted through the system. The systems are still used though because they are easy to set up!

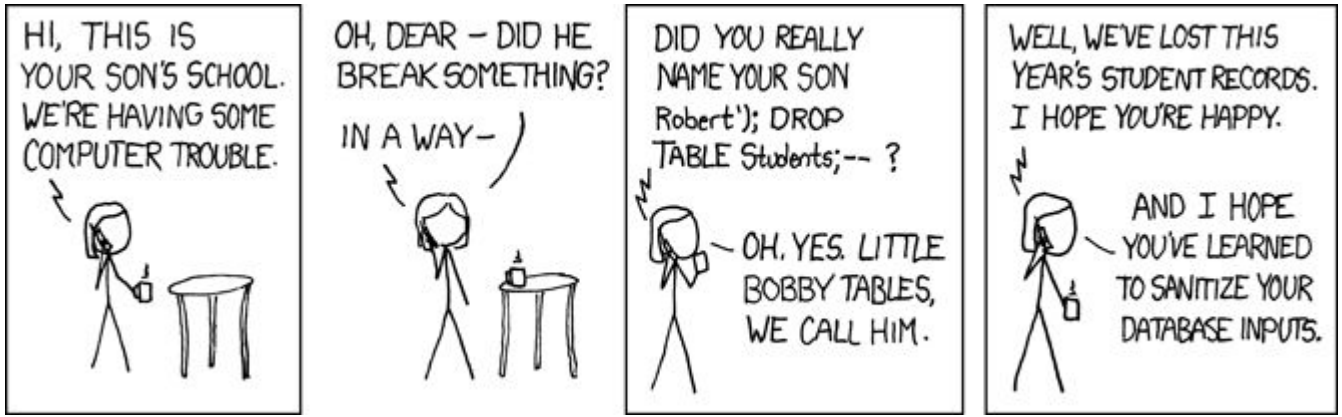
So, imagine you have a bulletin board system, where multiple users can write posts.

Let's say a user is no longer using the system so the user is deleted from the users table.

Will there be posts still available in the posts table that the user wrote?

- Possibly! You can create a (deleted user) record as a place holder for a user who posted messages but was deleted from the system.

(Or, see also)


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Aside: Processing data – Dealing with Murphy

Try to make sure people are on the same page before the crisis hits, so you know which systems are critical to avoid powering off the wrong system over the weekend.

- Tape down any loose wires as needed.

Assume file systems will fail (running out of inodes, bad hdd…)

Move “one off” systems to a production facility as soon as feasible.

Assume hard drives will fail – have backups.

Assume cpus will fail – have backups.

Assume power will fail – have (working) UPS’s.

Assume code will break – have dev, staging, and production copies of data & apps.

- Don’t shortchange development / staging / backup / processing hardware.

Processing Sanity Checking

Sanity check at various points.

Check the encodings to make sure it makes sense at critical points.

- Use IE or another browser to verify that it looks ok
- If you are not sure, run text through translation software as a sanity check, or ask a native speaker.

When first looking at a schema, make sure it makes logical sense to you. If possible, ask someone who knows what the data is.

When you have finished processing, run a sanity check script before releasing to users.

General Software Requirements

Do not make the search engine overly complex.

- Search the prioritized data first. Everything else (properties of files, images, etc.) is secondary.

Let the user quickly search through as many types of data as needed as fast as possible. (Think google search).

Use existing technology when possible.

- Use available API's to existing search engines or write a script... Allow the user to stay in your app as much as possible.

Use a "mailbox" methodology to allow the user to view results as results become available.

Multiple servers are good.

- Particularly for fall back or data size issues.

Fallback Considerations

It’s important to have fallback database, web servers, and full text indexes.

We have database that has “connection” information, as to where this searching server should connect for information.

- This could be localhost, backup host, staging host, etc.
- This allows for “data previews” before data is fully processed.
- This allows for 100% uptime, as you can programmatically tell servers where to look for information.

General solution?

Not publicly released.
But if you are interested, can get a dhs suitability, can get a trusted background position, and commute to DC and would like to work for MITRE, let me know.
We’re hiring to help support this position.

Final Comments

Try to verify the information contained within a database.

There are no promises of correctness.