The Database Forensic File Format and DF-Toolkit

By

James Wagner, Alexander Rasin, Karen Heart, Rebecca Jacob, and Jonathan Grier

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2019 USA

Portland, OR (July 15th - 19th)

The Database Forensic File Format and DF-Toolkit

James Wagner, Alexander Rasin, Karen Heart,
Rebecca Jacob, Jonathan Grier

DFRWS 2019

Portland, OR

DePaul Data Systems Lab

James Wagner

Alexander Rasin
(DB Systems Faculty)

Tanu Malik
(DB Systems Faculty)

Karen Heart (Systems Faculty
& Litigation Attorney)

Jacob Furst
(Security Faculty)

Jonathan Grier
(Grier Forensics)

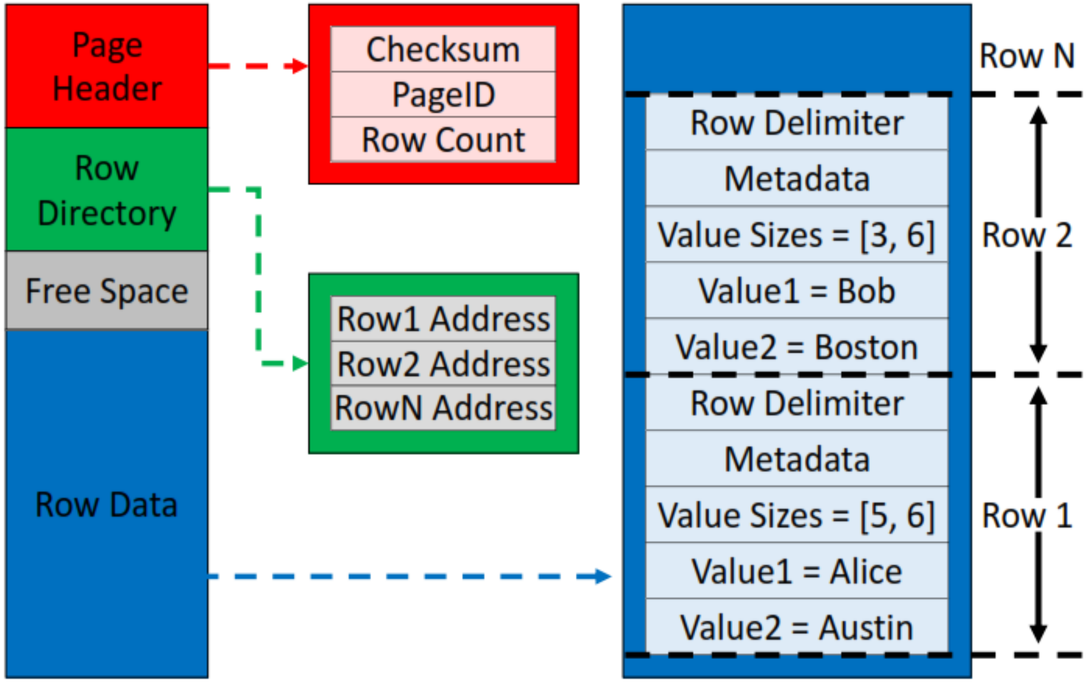
What is database forensics?

- **Database management system (DBMS):** software that stores and manages a collection of logically related data
  - Oracle and Microsoft SQL Server - corporate data
  - MySQL and PostgreSQL - webstore back-end
  - SQLite - personal applications (e.g., browser history and SMS)

- **Digital Forensics:**
  - solve crimes committed with computers (e.g., phishing and bank fraud)
  - solve crimes where evidence may reside on a computer (e.g., money laundering and child exploitation)
  - trace security breaches

Page Carving and DBCarver

- Pages
  - DBMS I/O
  - 4KB or 8KB
  - Tables, indexes, etc.
- DBCarver
  - Systematic carving process for database storage at page level.
  - **DFRWS ’15, ’16 & CIDR ’17**

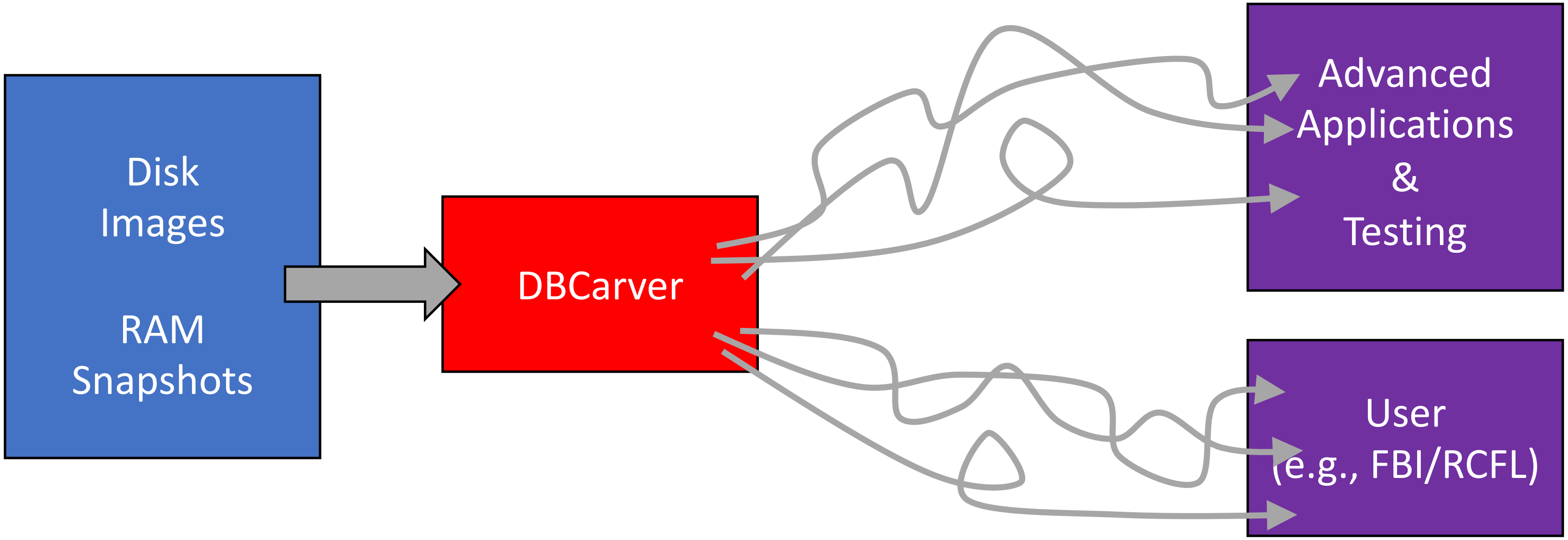


ApacheDerby, Firebird, IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, SQLite

Why Page Carving? Metadata!

- Our philosophy:
  “Reconstruct the system, not only the data.”

- Metadata and data allow for more complete timelines.

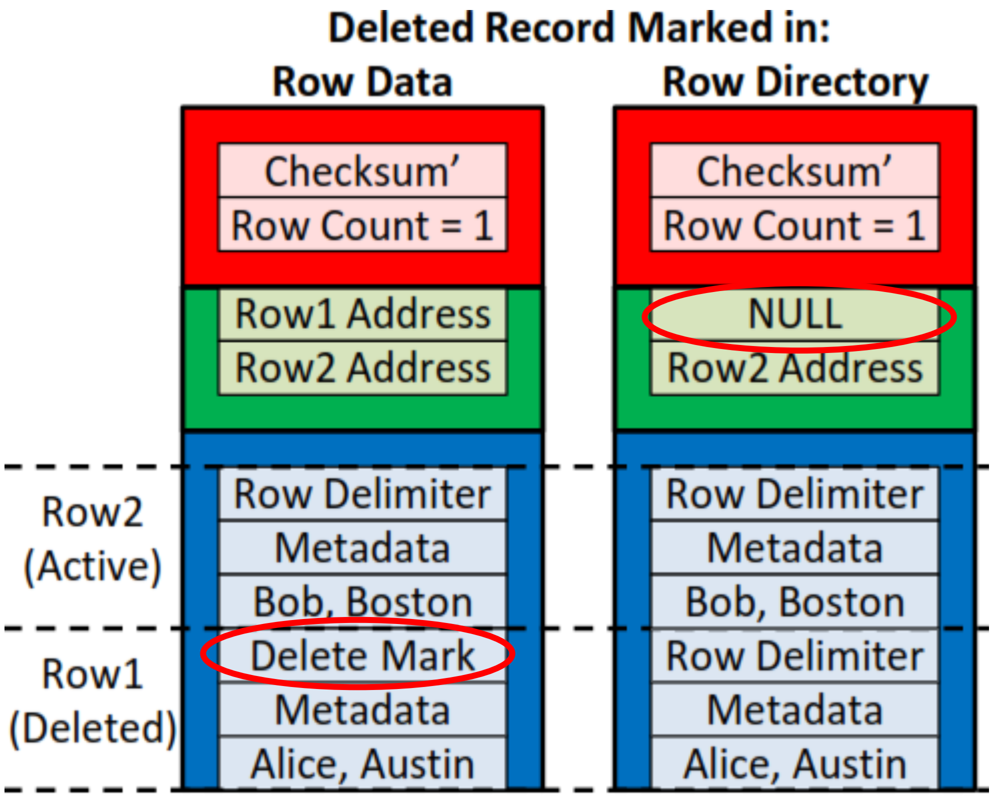


Metadata: Simple Example

- *"Return all deleted records and their offsets."*

- **Offset** – within disk image? DBMS file?

- **Delete flag** – multiple "delete" concepts.
  - DFRWS '16



Deleted Record Marked in:

Row Data

Checksum'
Row Count = 1
Row1 Address
Row2 Address
Row Delimiter
Metadata
Bob, Boston
Delete Mark
Metadata
Alice, Austin

Row Directory

Checksum'
Row Count = 1
NULL
Row2 Address
Row Delimiter
Metadata
Bob, Boston
Row Delimiter
Metadata
Alice, Austin

Row2
(Active)

Row1
(Deleted)

MySQL
PostgreSQL
Oracle
SQLite

IBM DB2
SQL Server

Metadata: Advanced Examples

Examples

- DBA bypasses (or tampers with) logs. -DFRWS ’17
- Sys Admin modifies DBMS file bytes. - EDBT ’18
- Storage optimization
    - SSDBM ‘17, DAPD ’19
- Data Sanitization

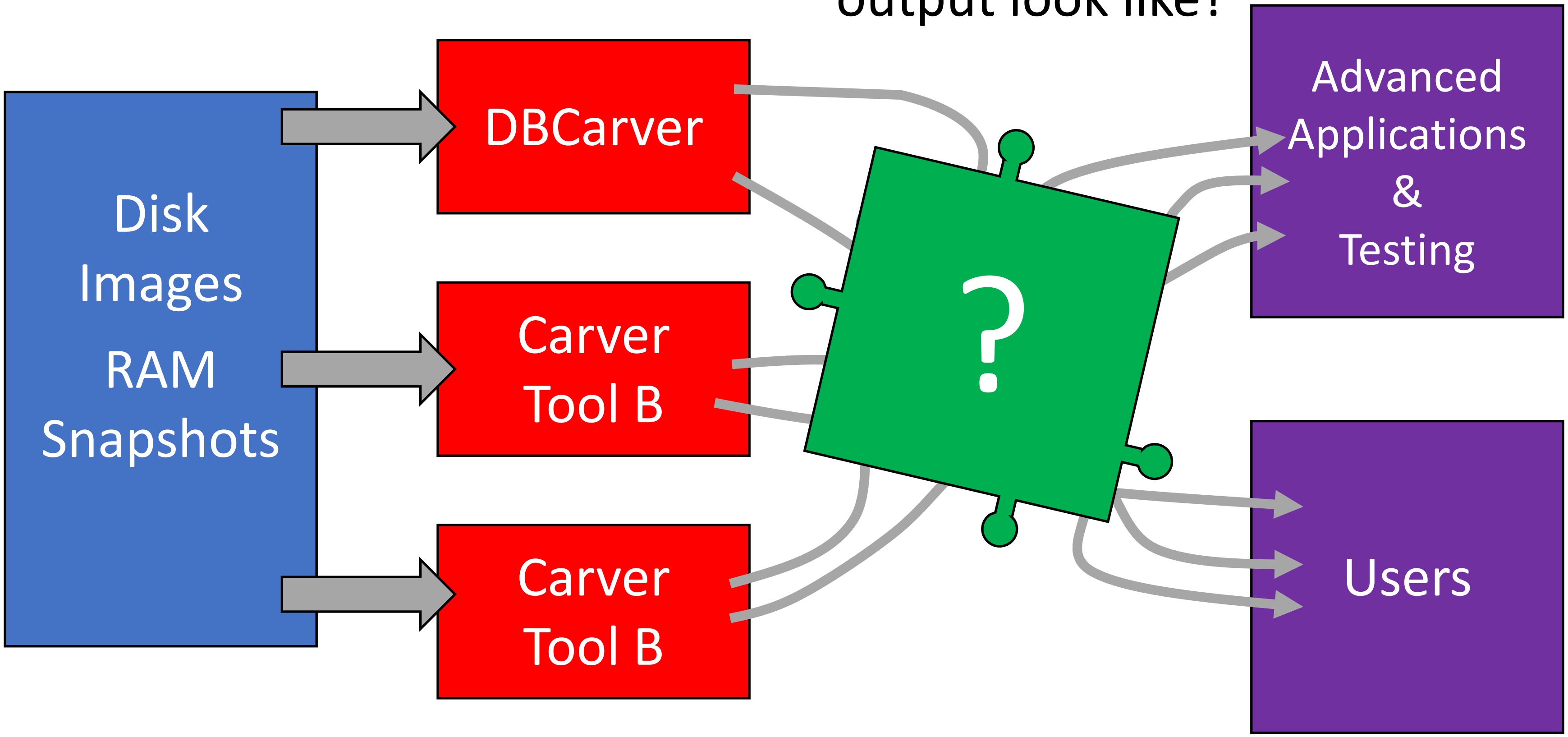
Relevant Metadata

- Deletion flags
- Pointer deconstruction
- Object identifiers
- Page Identifiers
- Caching patterns
- Wait! There’s more!
    - Checksums
    - Free space pointers

Motivation

How do you compare carvers?

What should output look like?



Disk Images

RAM Snapshots

DBCarver

Carver Tool B

Carver Tool B

?

Advanced Applications & Testing

Users

It’s a Database!
Just recreate the tables…

Employee ID	Name	Department	Salary	Office#
1	Karen	CSC	90K	101
2	Alex	Chemistry	88K	102
3	Tanu	Math	92K	104
4	Jacob	History	75K	107

- Example 1: Return all deleted records & their offsets.

- Example 2: Find all records containing a string.

Example 1: Return all deleted records and their offsets.

Offset	Delete Flag	Employee ID	Name	Department	Salary	Office#
1K	No	1	Karen	CSC	90K	101
2K	No	2	Alex	Chemistry	88K	102
3K	Yes	3	Tanu	Math	92K	104
4K	No	4	Jacob	History	75K	107

★Not “DELETE” FROM VACUUM

- Metadata columns are not part of original instance
  - Must be added to every table - What if I want more metadata?
  - Users must be able to distinguish “real” columns.
- The data and metadata do not fit the relational model

Example 2: Find all records containing a string.

Offset	Delete Flag	Employee ID	Name	Department	Salary	Office#
1K	No	1	Karen	CSC	90K	101
2K	No	2	Alex	Chemistry	88K	102
3K	Yes	3	Tanu	Math	92K	104
4K	No	4	Jacob	History	75K	107

```
SELECT * FROM Employee
WHERE Name LIKE ‘%MyString%’
OR Department LIKE ‘%MyString%’ …
```

- Can’t filter by all columns in SQL.

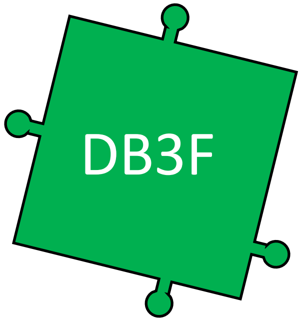
```
SELECT * FROM Customer
WHERE Name LIKE ‘%MyString%’
OR Address LIKE ‘%MyString%’ …
…
```

- How should output be saved?

Our contributions

Database Forensic File Format (DB3F):

- Abstract DBMS storage engine specifics.
- Simple to generate and ingest.
- Open and extensible
- Scalable



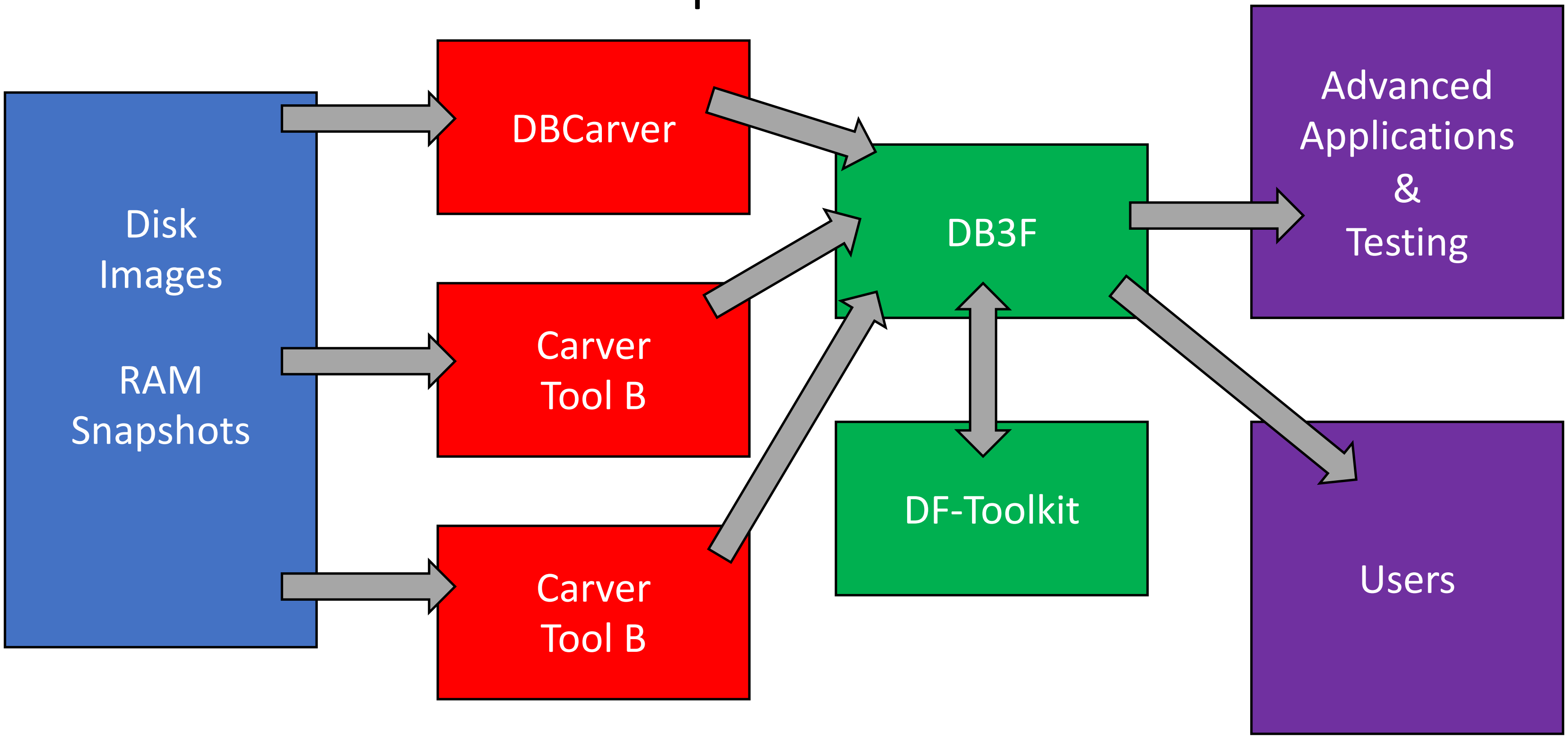
Database Forensic Toolkit (DF-Toolkit):

- Visibility – traverse data with a tree
- Display DBMS Objects
- Object Filtering
- Keyword Searches

Our Contributions

How do you compare carvers? – DB3F

What should output look like? – DB3F



Disk Images

RAM Snapshots

DBCarver

Carver Tool B

Carver Tool B

DB3F

DF-Toolkit

Advanced Applications & Testing

Users

DB3F

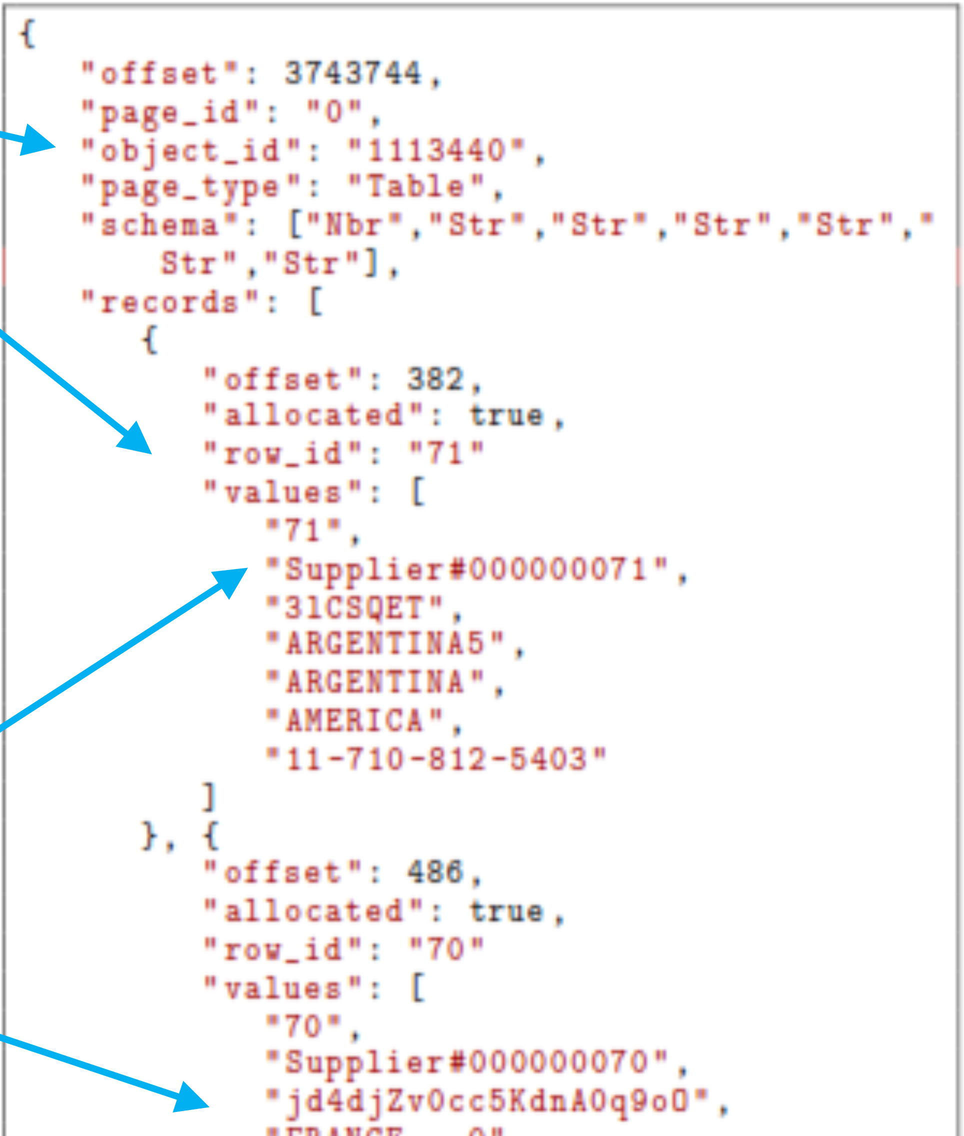
- Usable for all database carving tools

- 1 DBMS represented by 1 DB3F file
  - A disk image has multiple DB3F files if multiple DBMSes are present (e.g., don’t mix PostgreSQL and SQLite data)

- A DB3F file contains a series of JSON objects:
  - 1st line – DB3F file header JSON
  - Every other line represents a single page

DB3F: JSON Database Pages

- Page header metadata
- Records w/ metadata
- Fields can be added without affecting previous versions.
  - Currently not an exhaustive list of metadata fields



```
{
    "offset": 3743744,
    "page_id": "0",
    "object_id": "1113440",
    "page_type": "Table",
    "schema": ["Nbr","Str","Str","Str","Str","
        Str","Str"],
    "records": [
      {
        "offset": 382,
        "allocated": true,
        "row_id": "71"
        "values": [
          "71",
          "Supplier#000000071",
          "31CSQET",
          "ARGENTINA5",
          "ARGENTINA",
          "AMERICA",
          "11-710-812-5403"
        ]
      }, {
        "offset": 486,
        "allocated": true,
        "row_id": "70"
        "values": [
          "70",
          "Supplier#000000070",
          "jd4djZv0cc5KdnA0q9o0",
          "FRANCE    0"
```

71, Supplier#000000071, 31CSQET, ARGENTINA5, ARGENTINA, AMERICA, 11-710-812-5403

70, Supplier#000000070, jd4jZv0cc5KdnA0q9o0, FRANCE   0, …

DB3F: Popular Questions

- DB3F supports all datatypes
  - Describe in *schema* field

- Some datatypes don’t fit in a single page
  - E.g., BLOBs, large text fields
  - DBMSes store refs to these files

- Reliability
  - DB3F files use about 2x storage than the DBMS files.
  - Reading is dependent on JSON parsing.

DF-Toolkit

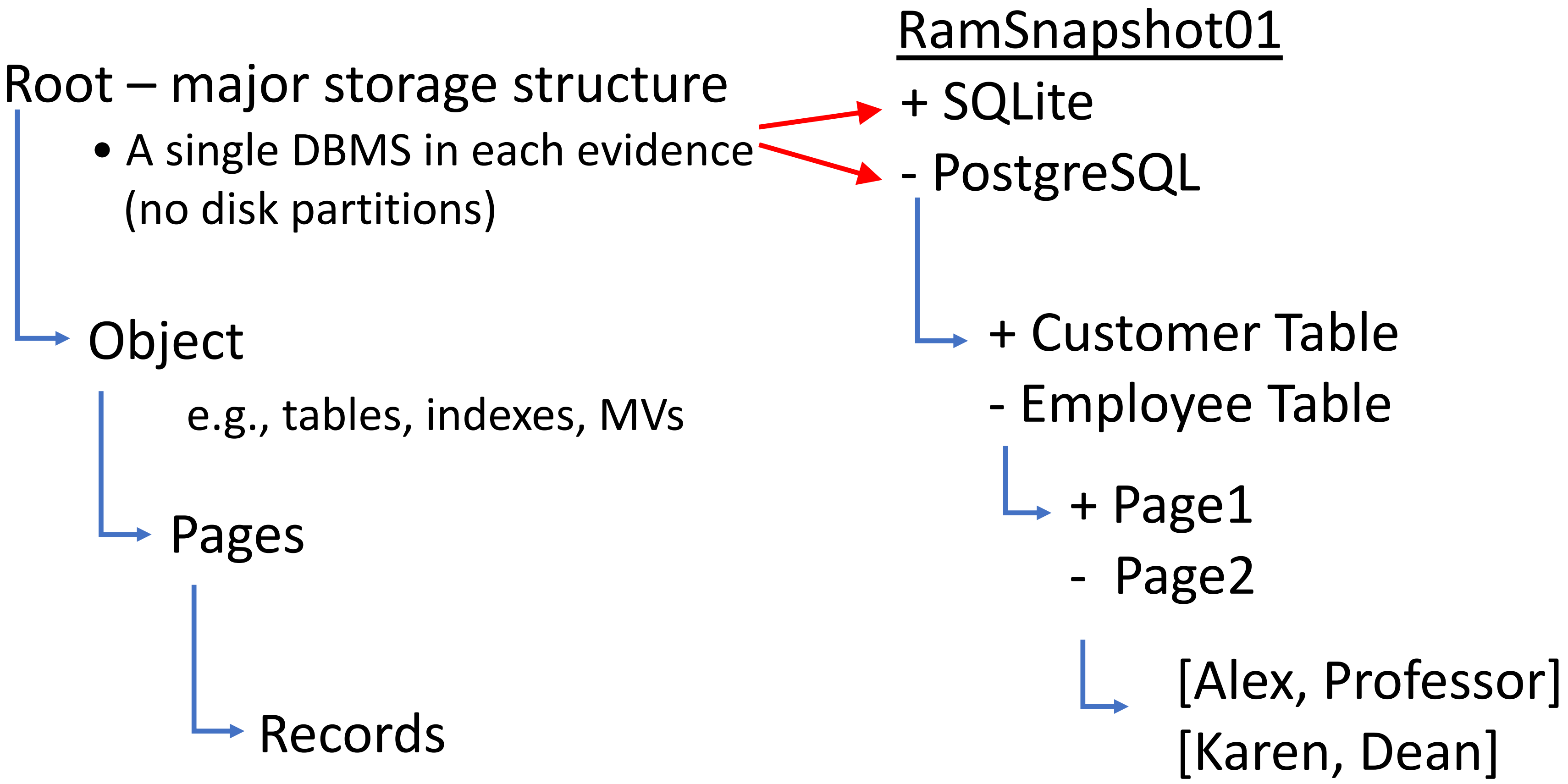
- A UI to view and filter DB3F files

- Tree structure
  - Traverse and view metadata and data

- Filtering and querying DB3F files

DF-Toolkit: Tree Nodes

- Our philosophy: “A DBMS as a separate system”



Root – major storage structure

- A single DBMS in each evidence (no disk partitions)

Object

e.g., tables, indexes, MVs

Pages

Records

RamSnapshot01

+ SQLite

- PostgreSQL

+ Customer Table

- Employee Table

+ Page1

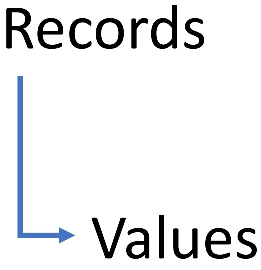
- Page2

[Alex, Professor]

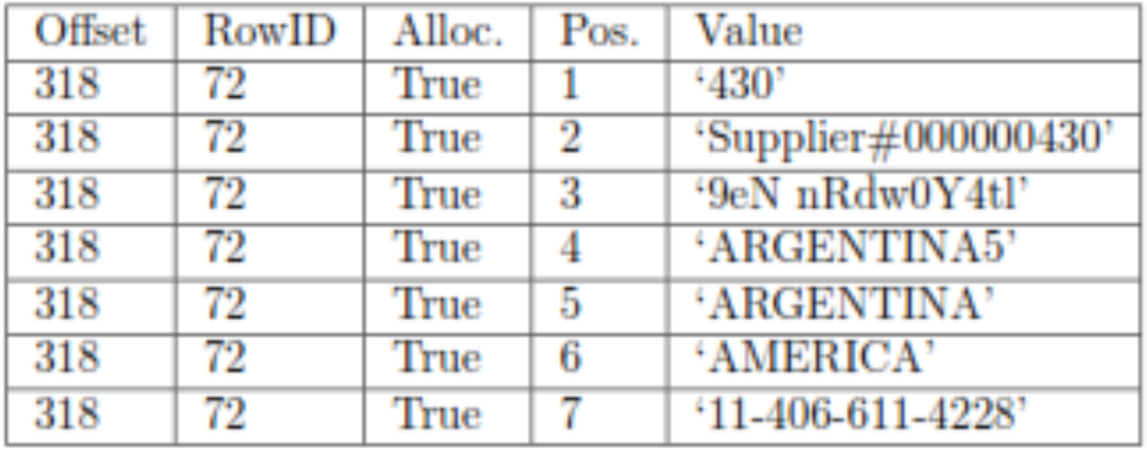
[Karen, Dean]

DF-Toolkit: Another tree level?

- Record values currently stored as a list.
- A Value table would store a row for every value and for all tables.
  - Ex. A table with 10 columns and 1M rows -> 10M rows
  - The SQL JOIN quickly becomes expensive


Records

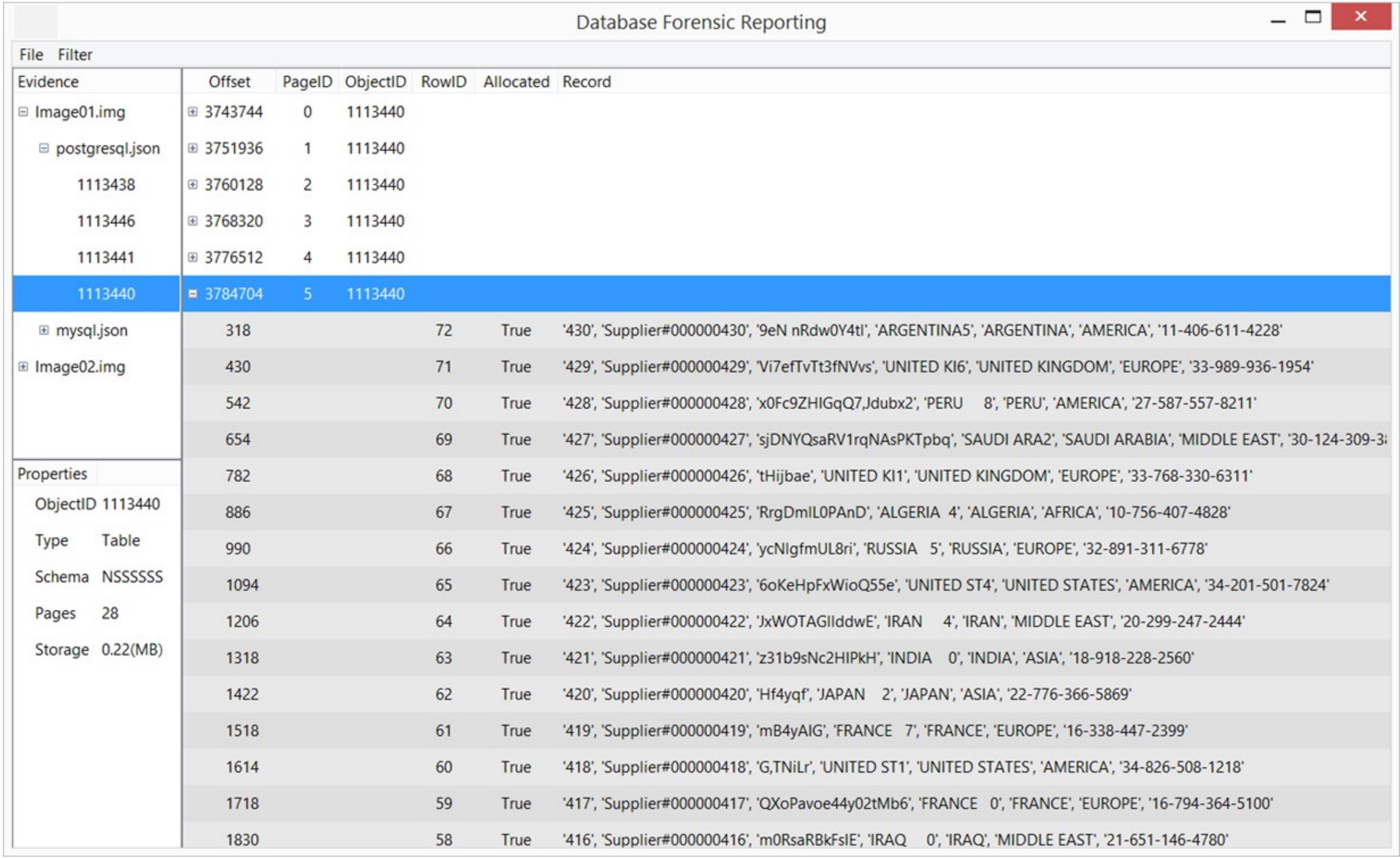
Values



Offset	RowID	Alloc.	Pos.	Value
318	72	True	1	‘430’
318	72	True	2	‘Supplier#000000430’
318	72	True	3	‘9eN nRdw0Y4tl’
318	72	True	4	‘ARGENTINA5’
318	72	True	5	‘ARGENTINA’
318	72	True	6	‘AMERICA’
318	72	True	7	‘11-406-611-4228’

DF-Toolkit: UI

DF-Toolkit User Interface Example



- ⊟ Image01.img
  - ⊟ postgresql.json
    - 1113438
    - 1113446
    - 1113441
    - 1113440
  - ⊞ mysql.json
- ⊞ Image02.img

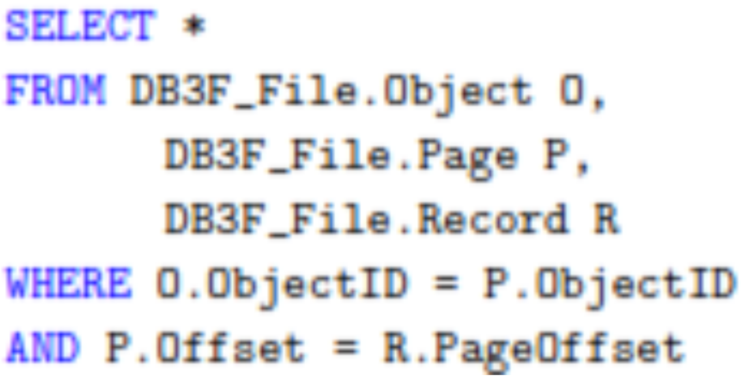
ObjectID	1113440
Type	Table
Schema	NSSSSSS
Pages	28
Storage	0.22(MB)

Offset	PageID	ObjectID	RowID	Allocated	Record
⊞ 3743744	0	1113440			
⊞ 3751936	1	1113440			
⊞ 3760128	2	1113440			
⊞ 3768320	3	1113440			
⊞ 3776512	4	1113440			
⊟ 3784704	5	1113440			
318			72	True	'430', 'Supplier#000000430', '9eN nRdw0Y4tl', 'ARGENTINA5', 'ARGENTINA', 'AMERICA', '11-406-611-4228'
430			71	True	'429', 'Supplier#000000429', 'Vi7efTvTt3fNVvs', 'UNITED KI6', 'UNITED KINGDOM', 'EUROPE', '33-989-936-1954'
542			70	True	'428', 'Supplier#000000428', 'x0Fc9ZHIGqQ7,Jdubx2', 'PERU 8', 'PERU', 'AMERICA', '27-587-557-8211'
654			69	True	'427', 'Supplier#000000427', 'sjDNYQsaRV1rqNAsPKTpbq', 'SAUDI ARA2', 'SAUDI ARABIA', 'MIDDLE EAST', '30-124-309-3
782			68	True	'426', 'Supplier#000000426', 'tHijbae', 'UNITED KI1', 'UNITED KINGDOM', 'EUROPE', '33-768-330-6311'
886			67	True	'425', 'Supplier#000000425', 'RrgDmIL0PAnD', 'ALGERIA 4', 'ALGERIA', 'AFRICA', '10-756-407-4828'
990			66	True	'424', 'Supplier#000000424', 'ycNIgfmUL8ri', 'RUSSIA 5', 'RUSSIA', 'EUROPE', '32-891-311-6778'
1094			65	True	'423', 'Supplier#000000423', '6oKeHpFxWioQ55e', 'UNITED ST4', 'UNITED STATES', 'AMERICA', '34-201-501-7824'
1206			64	True	'422', 'Supplier#000000422', 'JxWOTAGIlddwE', 'IRAN 4', 'IRAN', 'MIDDLE EAST', '20-299-247-2444'
1318			63	True	'421', 'Supplier#000000421', 'z31b9sNc2HIPkH', 'INDIA 0', 'INDIA', 'ASIA', '18-918-228-2560'
1422			62	True	'420', 'Supplier#000000420', 'Hf4yqf', 'JAPAN 2', 'JAPAN', 'ASIA', '22-776-366-5869'
1518			61	True	'419', 'Supplier#000000419', 'mB4yAIG', 'FRANCE 7', 'FRANCE', 'EUROPE', '16-338-447-2399'
1614			60	True	'418', 'Supplier#000000418', 'G,TNiLr', 'UNITED ST1', 'UNITED STATES', 'AMERICA', '34-826-508-1218'
1718			59	True	'417', 'Supplier#000000417', 'QXoPavoe44y02tMb6', 'FRANCE 0', 'FRANCE', 'EUROPE', '16-794-364-5100'
1830			58	True	'416', 'Supplier#000000416', 'm0RsaRBkFsIE', 'IRAQ 0', 'IRAQ', 'MIDDLE EAST', '21-651-146-4780'

DF-Toolkit: Filtering

Pre-filled SQL

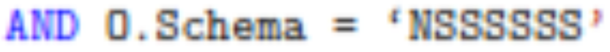
- The JOIN needed for a single evidence.
    - Does not change.



```
SELECT *
FROM DB3F_File.Object O,
      DB3F_File.Page P,
      DB3F_File.Record R
WHERE O.ObjectID = P.ObjectID
AND P.Offset = R.PageOffset
```

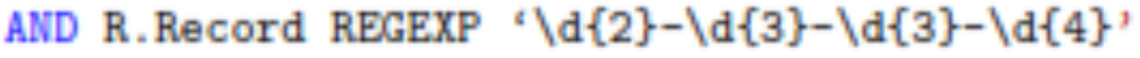
User-added Conditions

- Example conditions:
    - Filter on column datatypes for a table.



```
AND O.Schema = ‘NSSSSSS’
```

- Filter on a REGEX



```
AND R.Record REGEXP ‘\d{2}-\d{3}-\d{3}-\d{4}’
```

Future Work

Contact Us!

- User study
  - Aggregate collaborator criteria
- System Catalog Information
  - E.g., Replace ObjectID# with table “Customer”
  - We assume incomplete systems
  - DBMS-specific
- Integration of Non-DBMS Data
  - E.g., those references to large data

Contact and Info

- Email
  - Jay: jwagne32@depaul.edu
  - Alex: arasin@depaul.edu
  - Jonathan: jdgrier@grierforensics.com

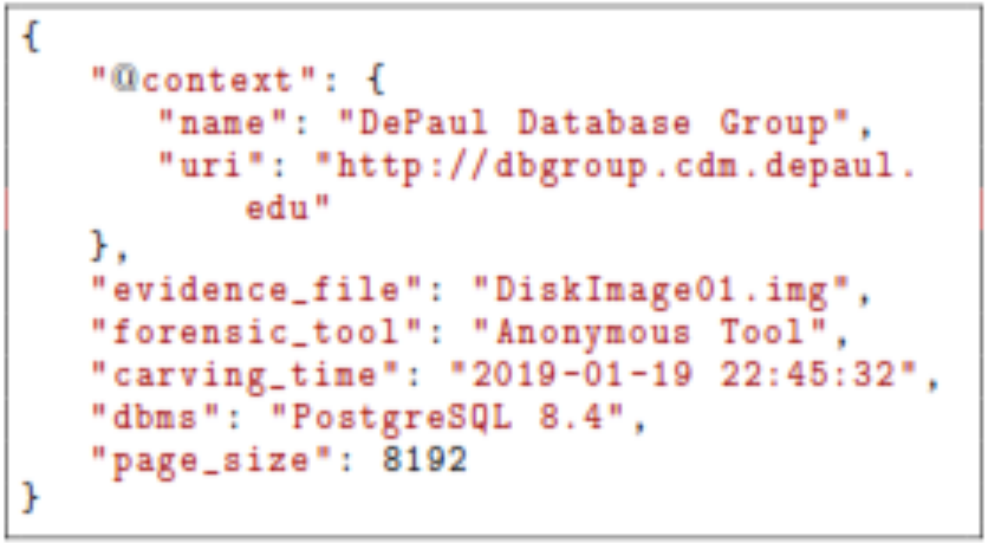
- DB3F Examples and DF-Toolkit: http://dbgroup.cdm.depaul.edu/DF-Toolkit.html

- Poster – tomorrow at lunch

  “Database Forensics: Where the Wild Things Are”

DB3F: JSON Header

- High-level metadata
- Organizations can easily add/remove fields for their SOP requirements.

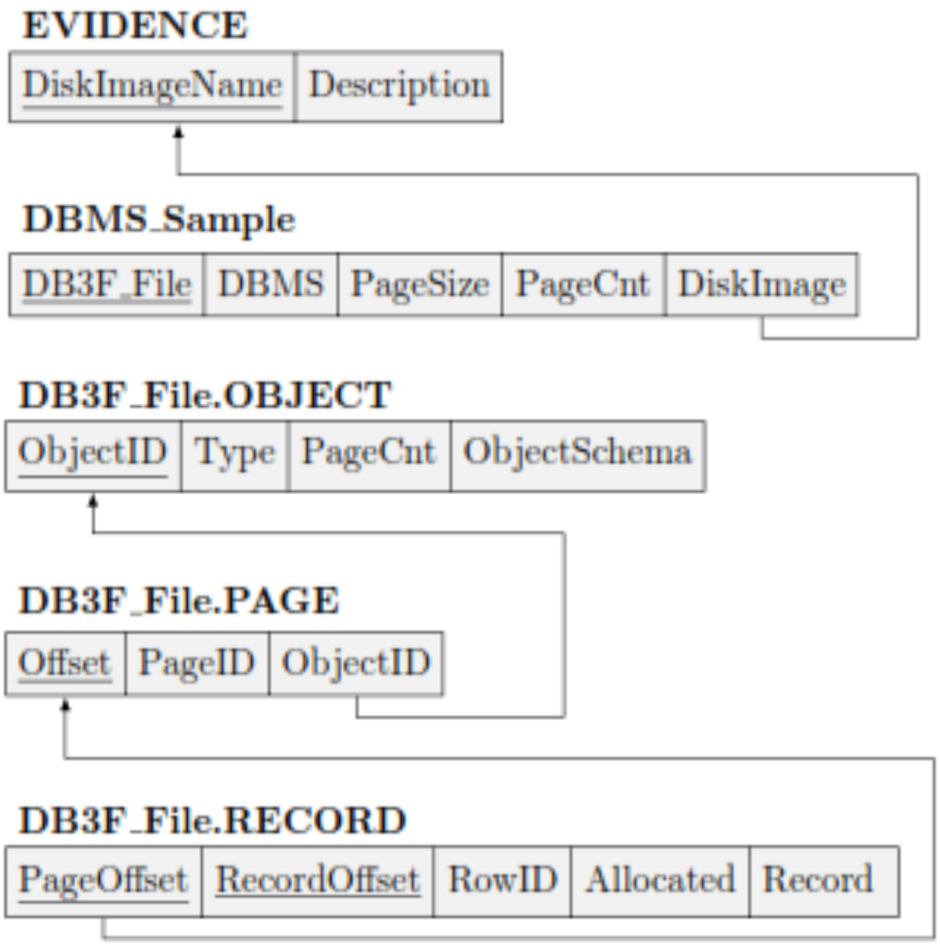


```
{
    "@context": {
        "name": "DePaul Database Group",
        "uri": "http://dbgroup.cdm.depaul.
            edu"
    },
    "evidence_file": "DiskImage01.img",
    "forensic_tool": "Anonymous Tool",
    "carving_time": "2019-01-19 22:45:32",
    "dbms": "PostgreSQL 8.4",
    "page_size": 8192
}
```

DF-Toolkit: Tree Nodes

- Our philosophy: “A DBMS as a separate system”

+ Root – major storage structure
  - A single DBMS in each evidence (no disk partitions)
  - Ex: RAM snapshot w/ PostgreSQL and SQLite
  + Object
    – e.g., tables, indexes, MVs
    + Pages
      + Records



EVIDENCE

DiskImageName	Description

DBMS_Sample

DB3F_File	DBMS	PageSize	PageCnt	DiskImage

DB3F_File.OBJECT

ObjectID	Type	PageCnt	ObjectSchema

DB3F_File.PAGE

Offset	PageID	ObjectID

DB3F_File.RECORD

PageOffset	RecordOffset	RowID	Allocated	Record