osquery

Internals: How sqlite3 & rocksdb are used

Alex Malone

Software Engineer

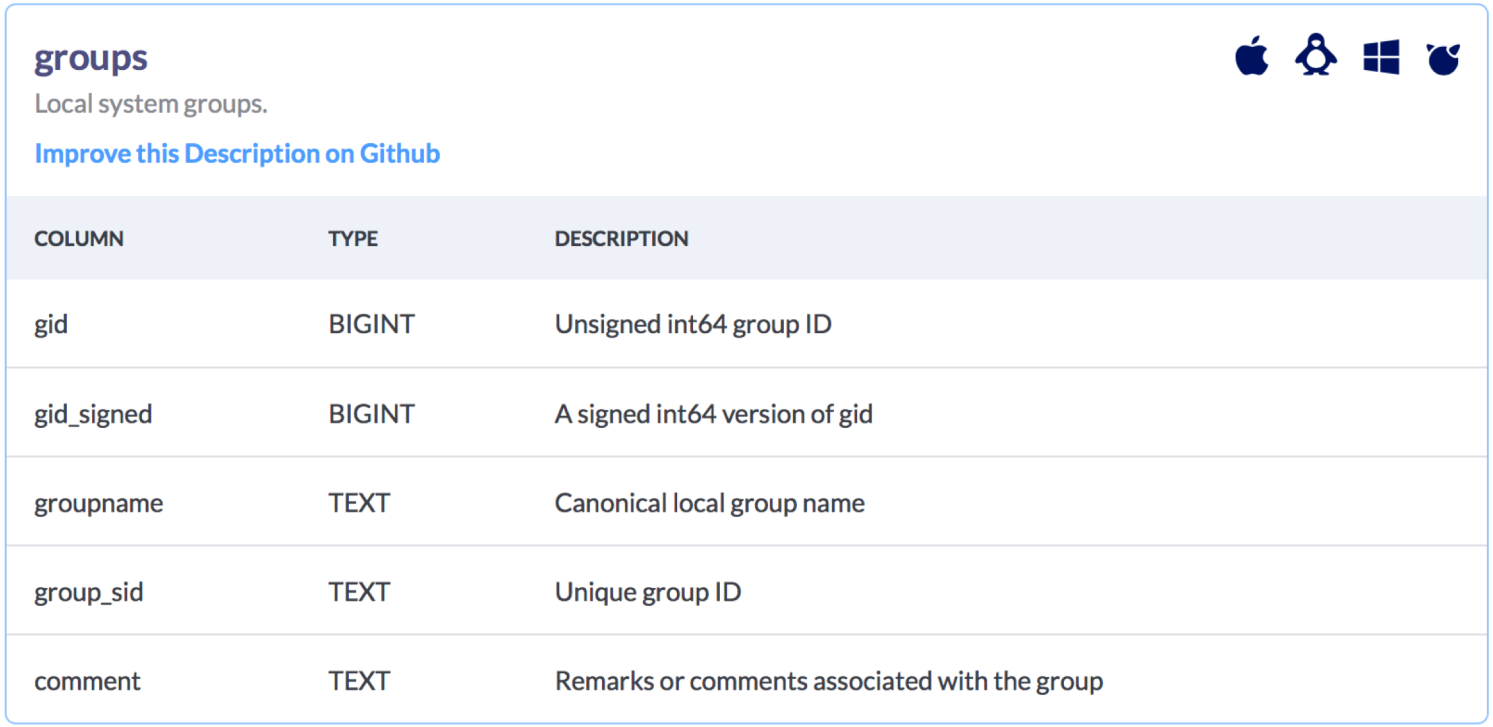
AT&T Cybersecurity

QUERYCON 2019

Schema describes ‘Virtual Tables’

- Published Schema `https://www.osql.io/schema/`
- Derived from /specs/*.table files

(these files going away in future? Pull data from instances?)

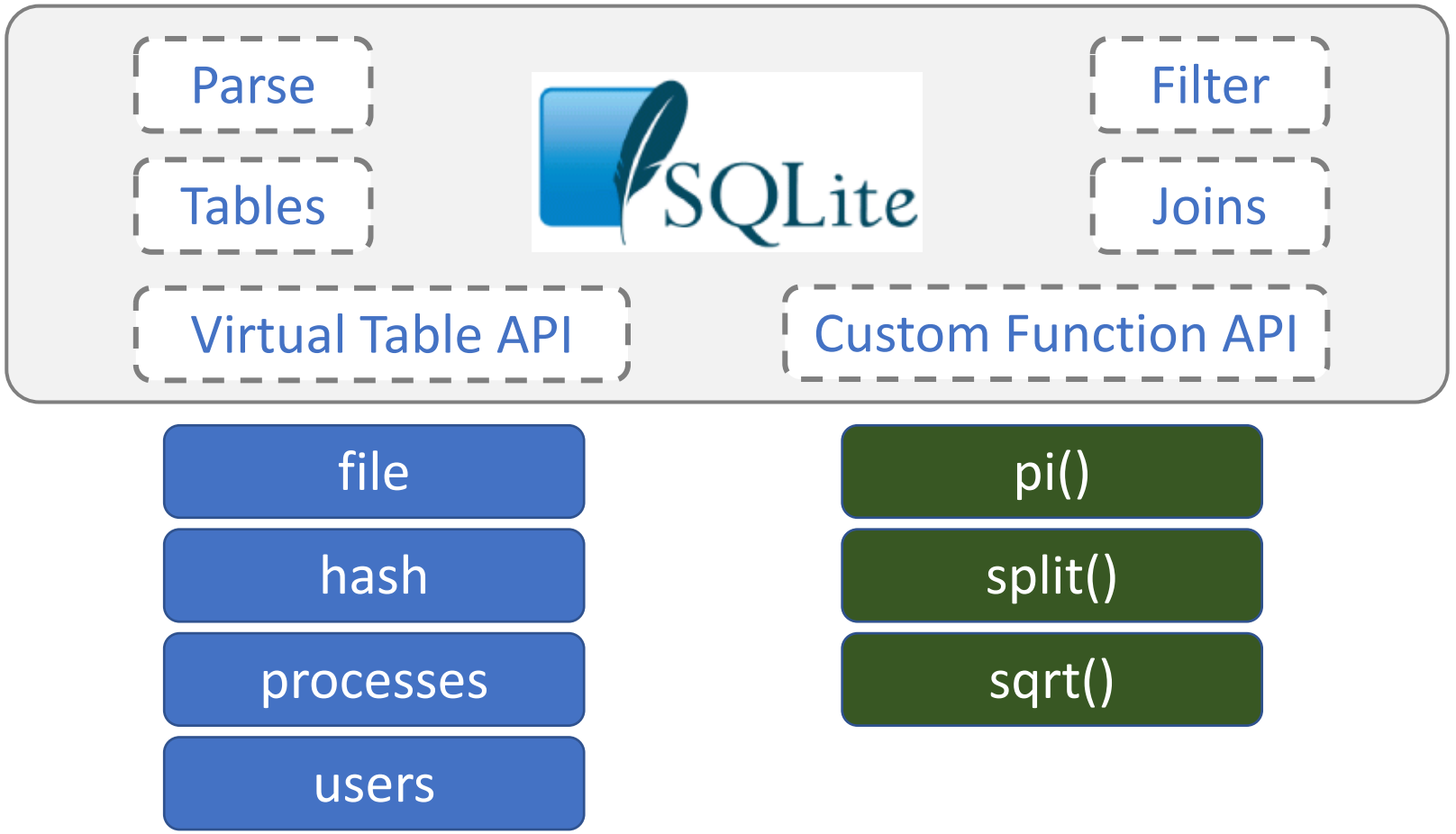


groups

Local system groups.

Improve this Description on Github

COLUMN	TYPE	DESCRIPTION
gid	BIGINT	Unsigned int64 group ID
gid_signed	BIGINT	A signed int64 version of gid
groupname	TEXT	Canonical local group name
group_sid	TEXT	Unique group ID
comment	TEXT	Remarks or comments associated with the group


Parse
Filter
SQLite
Tables
Joins
Virtual Table API
Custom Function API
file
hash
processes
users
pi()
split()
sqrt()

```
db = sqlite3_open(":memory:")

register_my_tables(db)
register_my_functions(db)

stmt = sqlite3_prepare_v2(db,
                          “SELECT * FROM processes”)

while (!done) :
    sqlite3_step(stmt)
    results += extract_row(stmt)
```

Virtual Table Example generate()

```
QueryData genLoggedInUsers(QueryContext& context) {
  QueryData results;

  while ((entry = getutxent()) != nullptr) {

    Row r;
    if (kLoginTypes.count(entry->ut_type) == 0) {
      r["type"] = "unknown";
    } else {
      r["type"] = kLoginTypes.at(entry->ut_type);
    }
    r["user"] = TEXT(entry->ut_user);
    r["tty"] = TEXT(entry->ut_line);
    r["host"] = TEXT(entry->ut_host);
    r["time"] = INTEGER(
    r["pid"] = INTEGER(entry->ut_pid);
    results.push_back(r);
  }

  return results;
}
```

Convert To String

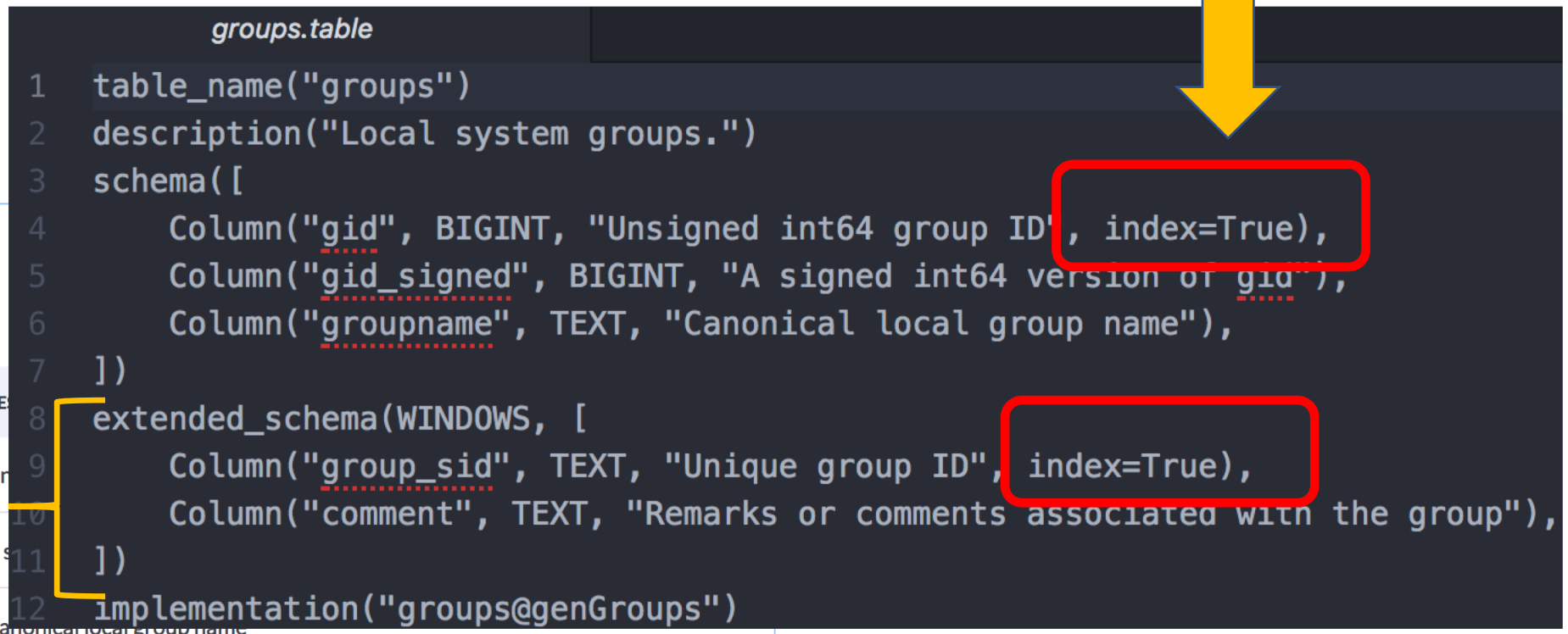
QueryData *genStuff*(QueryContext& context)

QueryContext provides:

- What columns were requested?
- Constraints
  e.g. path == ‘/usr/bin/grep’
  e.g. path LIKE ‘/usr/bin/%’

Getting Constraints: indexes

‘groups’ table spec file



groups

Local system groups.

Improve this Description on Github

COLUMN	TYPE	DE
gid	BIGINT	Un
gid_signed	BIGINT	A s
groupname	TEXT	Canonical local group name
group_sid	TEXT	Unique group ID
comment	TEXT	Remarks or comments associated with the group

```
struct sqlite3_module {

  • xBestIndex
  • xFilter
  • xNext
  • xEof
  • xColumn
  …
}
```

```
struct sqlite3_module {

   • xBestIndex
   • xFilter
   • xNext
   • xEof
   • xColumn
   …
}
```

osquery glue

```
.table spec

generate()
```

osquery/sql/virtual_table.cpp

xBestIndex :
Who's better?

Table : processes
Constraints :
    column : pid
    operator : EQUALS

Example 1: no index

```
SELECT * FROM processes
WHERE path LIKE '/usr/local/%'
```

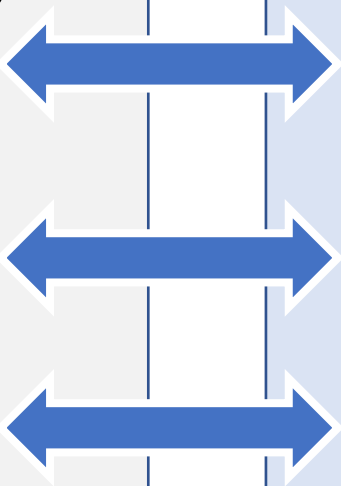
```
xBestIndex (column: processes.path, op:LIKE)
xFilter ()
  xNext – row data
  xNext – row data
  xNext – row data
  ...
  xNext – DONE
```

```
genProcesses()
```

Example 2: using index

```
SELECT * FROM processes
WHERE pid IN (1,4,20,100,200,300)
```

```
xBestIndex (processes.pid, op:EQUALS)
xFilter (pid == 1)
  xNext – row data
  xNext – DONE
xFilter (pid == 4)
  xNext – row data
  xNext – DONE
xFilter (pid == 20)
  xNext – row data
  xNext – DONE
…
```

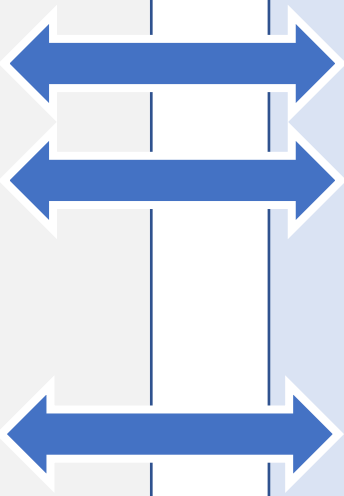


```
genProcesses(pid == 1)

genProcesses(pid == 4)

genProcesses(pid == 20)

...
```

Example 3: JOIN on index

```
SELECT * FROM listening_ports
LEFT JOIN processes USING (pid) WHERE protocol=6
```

```
xBestIndex (listening_ports.protocol, op:EQUALS)
xBestIndex (processes.pid, op:EQUALS)
xFilter (listening_ports)
  xNext – row data
  xFilter (processes.pid = ?)
     xNext row data
     xNext – DONE
  xNext – row data
  xFilter (processes.pid = ?)
     xNext row data
     xNext – DONE
  xNext – DONE
…
```



```
genListeningPorts()
genProcesses(pid == ?)
genProcesses(pid == ?)
...
```

But How Can I Check?

osqueryd -S --verbose=1 **--planner=1**

```
osquery> SELECT l.protocol,l.port,p.pid,p.name,p.parent FROM listening_ports l LEFT JOIN processes p USING (pid) WHERE protocol =6;
osquery planner: Evaluating constraints for table: listening_ports [index=0 column=2 term=0 usable=1]
osquery planner: Recording constraint set for table: listening_ports [cost=1.000000 size=0 idx=16]
osquery planner: Evaluating constraints for table: processes [index=0 column=0 term=2 usable=1]
osquery planner: Adding constraint for table: processes [column=pid arg_index=1 op=2]
osquery planner: Recording constraint set for table: processes [cost=1.000000 size=1 idx=17]
osquery planner: Opening cursor (16) for table: listening_ports
osquery planner: Opening cursor (17) for table: processes
osquery planner: Filtering called for table: listening_ports [constraint_count=1 argc=0 idx=16]
osquery planner: Scanning rows for cursor (16)
osquery planner: listening_ports generate returned row count:204
osquery planner: Filtering called for table: processes [constraint_count=1 argc=1 idx=17]
osquery planner: Adding constraint to cursor (17): pid = 369
osquery planner: Scanning rows for cursor (17)
osquery planner: processes generate returned row count:1
osquery planner: Filtering called for table: processes [constraint_count=1 argc=1 idx=17]
osquery planner: Adding constraint to cursor (17): pid = 369
osquery planner: Scanning rows for cursor (17)
osquery planner: processes generate returned row count:1
osquery planner: Filtering called for table: processes [constraint_count=1 argc=1 idx=17]
osquery planner: Adding constraint to cursor (17): pid = 527
osquery planner: Scanning rows for cursor (17)
osquery planner: processes generate returned row count:1
osquery planner: Closing cursor (16)
osquery planner: Closing cursor (17)
+----------+-------+-----+------------+--------+
| protocol | port  | pid | name       | parent |
+----------+-------+-----+------------+--------+
| 6        | 53444 | 369 | rapportd   | 1      |
| 6        | 53444 | 369 | rapportd   | 1      |
| 6        | 19421 | 527 | ZoomOpener | 1      |
+----------+-------+-----+------------+--------+
```

Table Implementations


4GIFs.com

```
std::set<int> getProcList(const QueryContext& context) {
  std::set<int> pidlist;
  if (context.constraints.count("pid") > 0 &&
      context.constraints.at("pid").exists(EQUALS)) {
    for (const auto& pid : context.constraints.at("pid").getAll<int>(EQUALS)) {
      if (pid >= 0) {
        pidlist.insert(pid);
      }
    }
    return pidlist;
  }
```

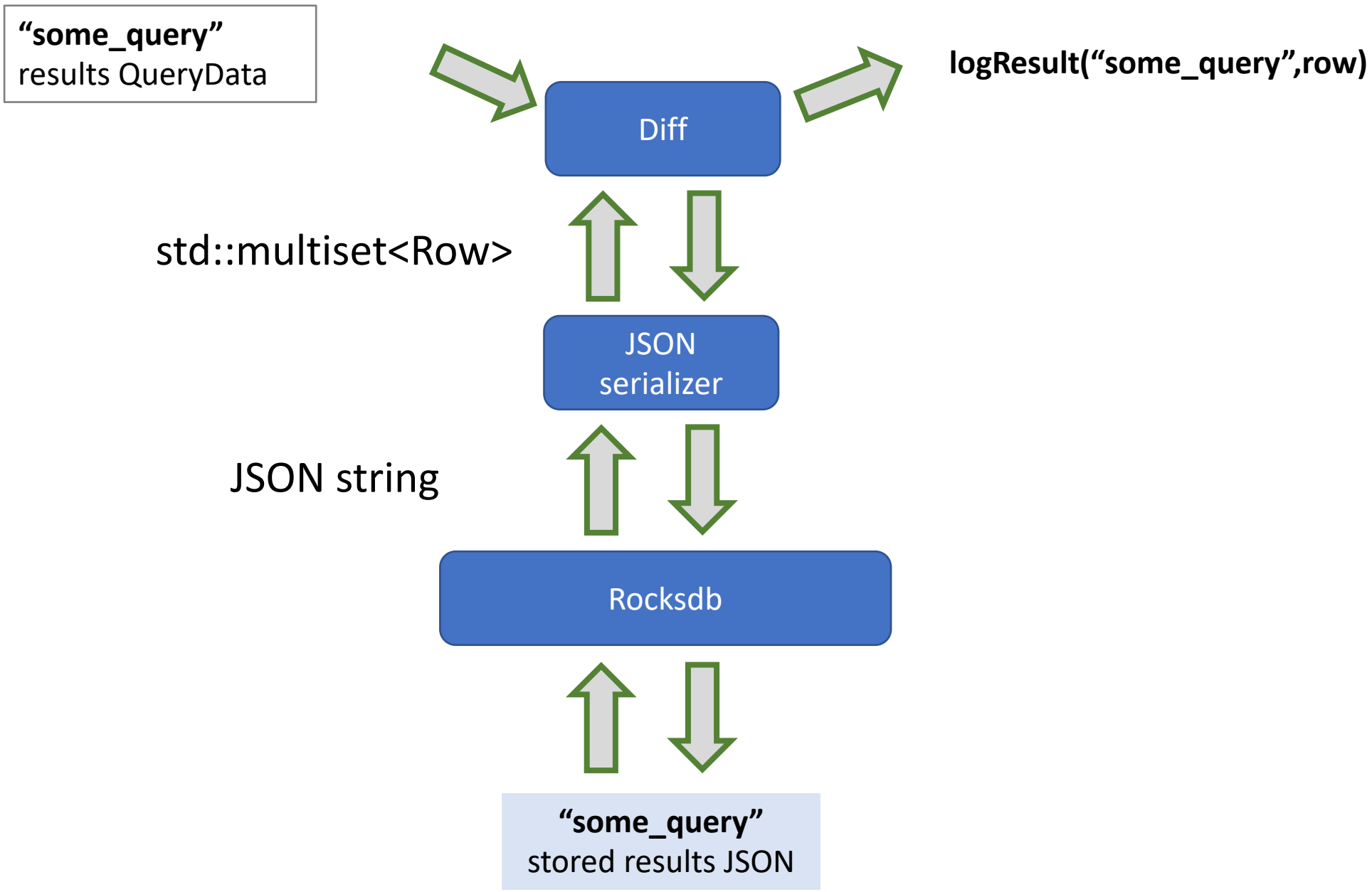
Most table implementations assume:

- generate() called once
- all constraints come in together in a list.

```
void genProcNamePathAndOnDisk(const QueryContext& context,
                              int pid,
                              const struct proc_cred& cred,
                              Row& r) {
  if (!context.isAnyColumnUsed({"name", "path", "on_disk"})) {
    return;
  }

  std::string path;
  if (pid == 0) {
```

Optimize by not doing extra work
when columns not requested.

Persistence : rocksdb


“some_query”
results QueryData
Diff
logResult(“some_query”,row)
std::multiset<Row>
JSON
serializer
JSON string
Rocksdb
“some_query”
stored results JSON

Diff pseudo-code

osquery/core/query.cpp

```
std::multiset<Row> prevResults
DiffResults r

foreach(row)
   if prevResults.find(row) :
      prevResults.remove(row)
   else
     r.added(row)

foreach(prevResults):
   r.removed(prevRow)
```

Rocksdb is a key,value database.

Database keys for scheduled queries:

“query.<query_name>”

“cache.<query_name>”

“<query_name>_counter”

“<query_name>_epoch”

Batches

- events codepath has added support for setDatabaseBatch()
- no need to acquire lock for each row

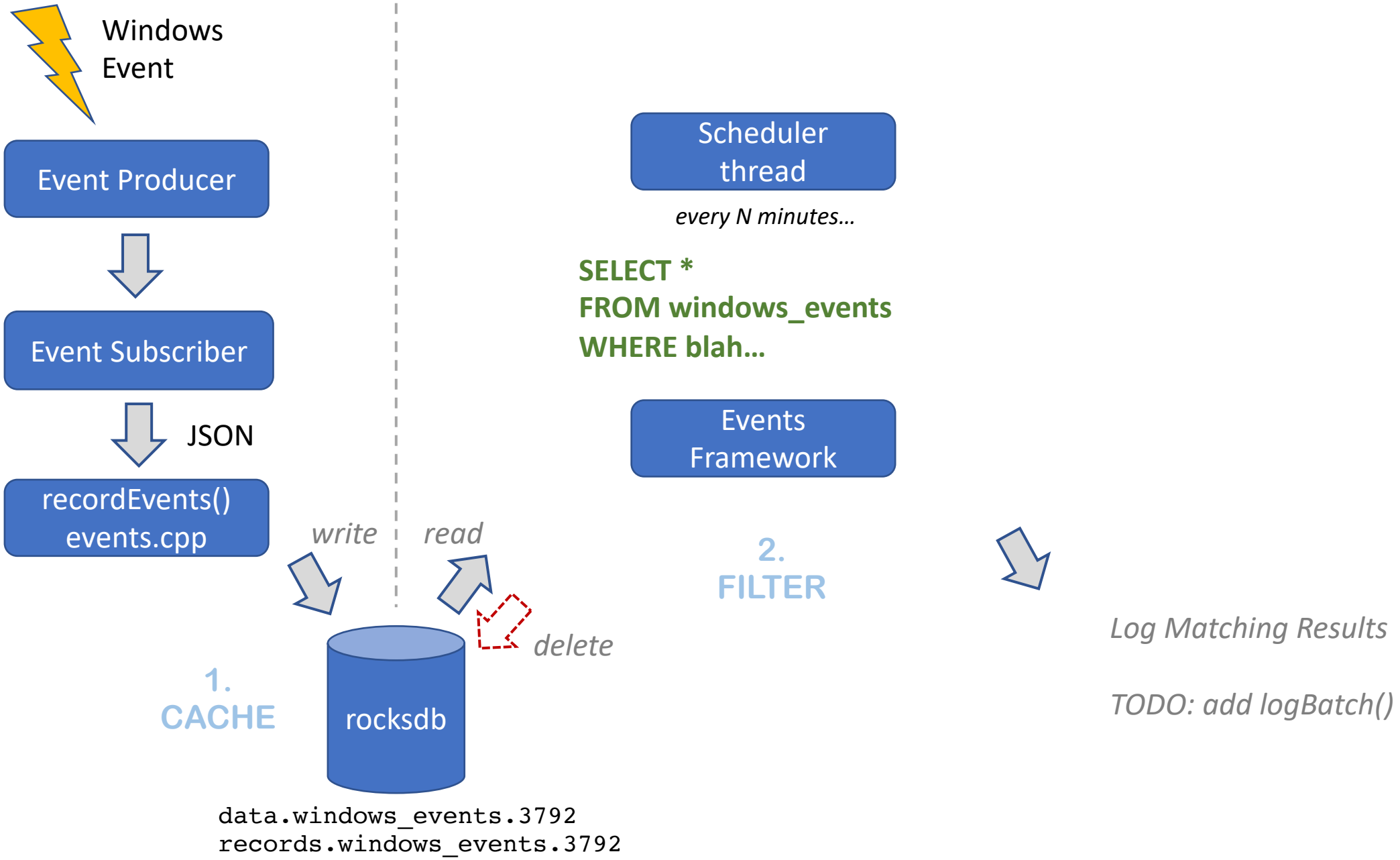
BufferedLogger (aws_kinesis, tls)

- Uses rocksdb to cache results. Adds lots of DB churn.
- no logStringBatch. Adding / removing each result is slow on windows.

Events


BH

Event Lifecycle (Buffered Logger)



Keys for Events tables

- data.<table>.<eid>
- eid.<table> # last
- records.<table>.60.<timestamp>
- indexes.<table>.60

Expiring

- Expiry is checked periodically
- Number of keys are queried to get ‘row count’ to see if over row limit
- Removes oldest first. Limit -

osquery bug 5379: Index on everything

Step 1 : Analyze surprisingly slow queries

- Make sure you have a query for **osquery_schedule** at least daily

- We found this one was using a ton of cpu. Why?

```
SELECT process.pid AS source_process_id
    , process.name AS source_process
    , process.path AS file_path
    , authenticode.result
    , authenticode.serial_number AS certificate_serial_number
    , authenticode.issuer_name AS certificate_issuer_name
    , authenticode.subject_name AS certificate_subject_name
    , parent_processes.path AS source_process_parent
    , parent_processes.cmdline AS source_process_parent_command
    , parent_processes.uid AS parent_uid
    , hashes.sha256 AS file_hash_sha256
FROM  processes  AS process
LEFT JOIN authenticode  ON process.path = authenticode.path
LEFT OUTER JOIN processes  AS parent_processes ON process.paren
LEFT JOIN hash  AS hashes ON hashes.path = process.path
WHERE authenticode.serial_number IN (
       '01a5d9599519b1bafcfad0e80b6d6735', '03e1e1aaa5bca19fba8
      , '0492f5c18e26fa0cd7e15067674aff1c', '065011a5bcbf83c09
      , '09813ee7318452c28a1f6426d1cee12d', '0b7279068beb15ffe
      , '0c3bc399adcc4235c45dee6da19324dc', '0cc0359c9c3cda00d
      , '1079bf56f4749ea4467874cb4f9264d8', '16163de1d788eb864
      , '19c697ffad46915bb4c3025140f1f2e1', '1caa0d0dadf32a240
      , '1f2851fc5df45da8446505203908f422', '20d0ee42fc901e6b3
      , '242c009f8ea4a360ba35405ee817769b', '256541e204619033f
      , '267bde888bc150151100f254d8caed67', '29126d77f17648a44
      , '2bc3be7a3033ab874a8517d89f49f3ab', '2edfb9fdcfa00ccb5
      , '30d3fe26591d8eac8c30667ac4999bd7', '30ef0d9710fa18be1
      , '31062e483e0106b18c982f0053185c36', '34f01ea2049bf24ab
```

```
SELECT process.pid AS source_process_id
   , process.name AS source_process
   , process.path AS file_path
   , authenticode.result
   , authenticode.serial_number AS certificate_serial_number
   , authenticode.issuer_name AS certificate_issuer_name
   , authenticode.subject_name AS certificate_subject_name
   , parent_processes.path AS source_process_parent
   , parent_processes.cmdline AS source_process_parent_commandline
   , parent_processes.uid AS parent_uid
   , hashes.sha256 AS file_hash_sha256
FROM  processes  AS process
LEFT JOIN authenticode  ON process.path = authenticode.path
LEFT OUTER JOIN processes  AS parent_processes ON process.parent = parent_processes.pid
LEFT JOIN hash  AS hashes ON hashes.path = process.path
WHERE authenticode.serial_number IN (
      '01a5d9599519b1bafcfad0e80b6d67
      , '0492f5c18e26fa0cd7e15067674af
      , '09813ee7318452c28a1f6426d1cee12d', '0b727
      , '0c3bc399adcc4235c45dee6da19324dc', '0cc0359c9c3cda
      , '1079bf56f4749ea4467874cb4f9264d8', '16163de1d788eb8645189d
      , '19c697ffad46915bb4c3025140f1f2e1', '1caa0d0dadf32a2404a75195ae
      , '1f2851fc5df45da8446505203908f422', '20d0ee42fc901e6b3a8fefe8c1
      , '242c009f8ea4a360ba35405ee817769b', '256541e204619033f8b09f9eb7
      , '267bde888bc150151100f254d8caed67', '29126d77f17648a440854bd522
      , '2bc3be7a3033ab874a8517d89f49f3ab', '2edfb9fdcfa00ccb5ab009ee3a
      , '30d3fe26591d8eac8c30667ac4999bd7', '30ef0d9710fa18be1d543acc99
      , '31062e483e0106b18c982f0053185c36', '34f01ea2049bf24ab7307fa61bapaaee
      , '387d56db4f6f01af42b54d1c7635baba', '3a677b45bf085b9a4f1634922ae0b640'
      , '3ac10e68f1ce519e84ddcd28b11fa542', '3f5
      , '3ffceba83fe00fef97f63cd92e77ebb9', '423
      , '42f3df8933b0fb6bba6d8bc364e916ef', '452
      , '45916947db98c766bf0978789b3890c9', '476
      , '47d5d5372bcb1562b4c9f4c2bdf13587', '48f
      , '4c0b2e9d2ef909d15270d4dd7fa5a4a5', '520
      , '54c6c1406fb4acb5d20674e99392c63e', '5c2
      , '5c2f97a31abc32b08cac0100598f32f6', '5cc
      , '5e3d76dc7e273e2f313fc0775847a2a2', '610
      , '613e2fa14e323c69ee3e720c27afe4ce', '66e
      , '6724340ddbc7252f7fb714b812a5c04d', '69a
      , '6eafa388654fa08e466b4bd38966d196', '758
      , '7bd55818c5971b63dc45cf57cbeb950b', '7ca
      , '7f01fb34d50690995e6f42c0771ce87f', '922
      , '65e6ade39a4170723ece567d635fe2eb')
```

- Serial_number column is not indexed.
- But osquery is telling sqlite it is.
- So authenticode table is queried 55 times!
  (once for each value IN list)

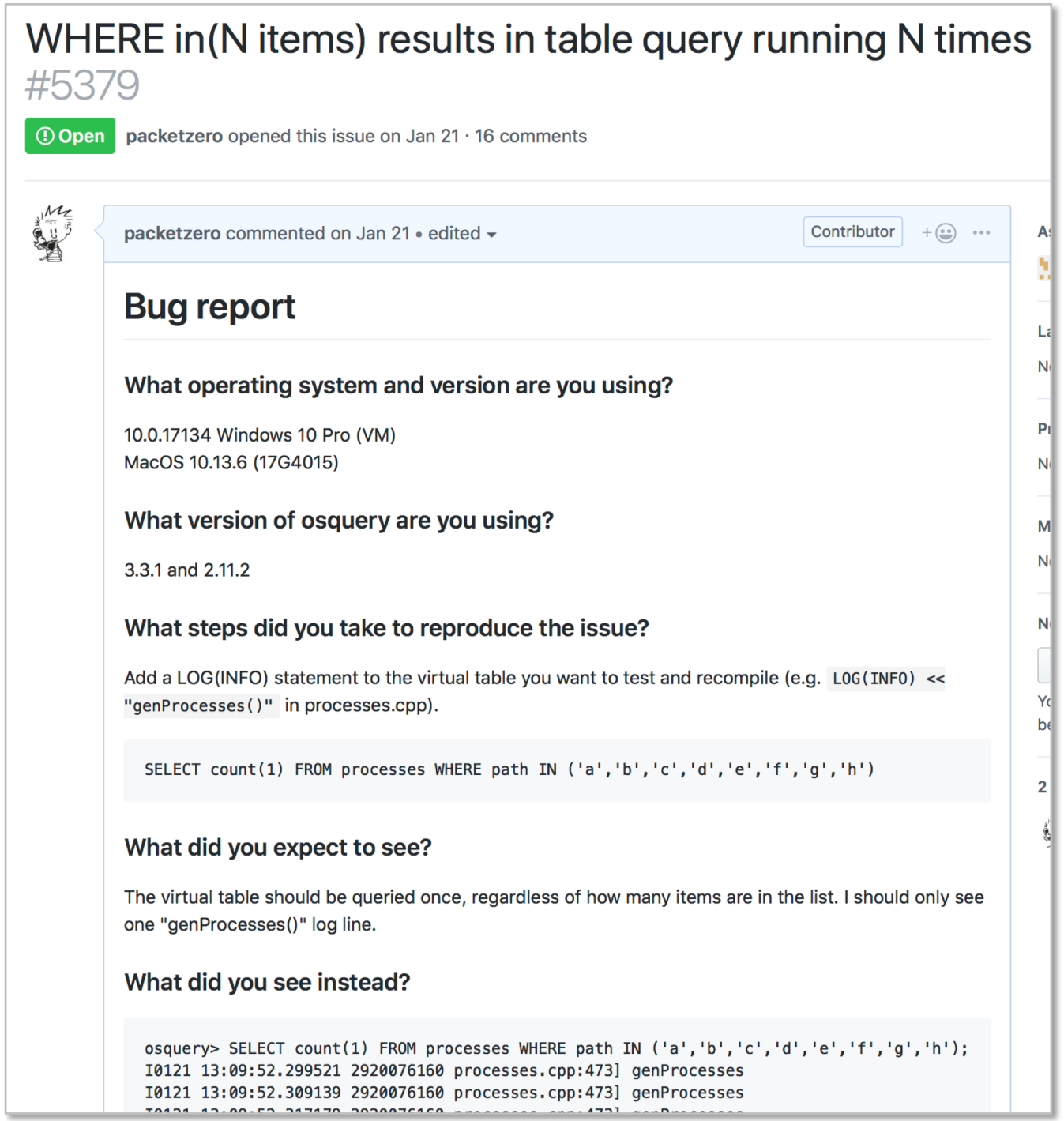
```
table_name("authenticode")
description("File (executable, bundle, installer, disk) code signing status.")
schema([
    Column("path", TEXT, "Must provide a path or directory", required=True),
    Column("original_program_name", TEXT, "The original program name that the
    Column("serial_number", TEXT, "The certificate serial number"),
    Column("issuer_name", TEXT, "The certificate issuer name"),
    Column("subject_name", TEXT, "The certificate subject name"),
    Column("result", TEXT, "The signature check result")
])
```

It turns out that osquery is telling
SQLite that it has an INDEX for everything.

INDEX means:

You can quickly lookup and return one row

So for tables that didn't implement an index,
Table will return all rows for each call



WHERE in(N items) results in table query running N times #5379

Open **packetzero** opened this issue on Jan 21 · 16 comments

packetzero commented on Jan 21 • edited — Contributor

Bug report

What operating system and version are you using?

10.0.17134 Windows 10 Pro (VM)
MacOS 10.13.6 (17G4015)

What version of osquery are you using?

3.3.1 and 2.11.2

What steps did you take to reproduce the issue?

Add a LOG(INFO) statement to the virtual table you want to test and recompile (e.g. `LOG(INFO) << "genProcesses()"` in processes.cpp).

```
SELECT count(1) FROM processes WHERE path IN ('a','b','c','d','e','f','g','h')
```

What did you expect to see?

The virtual table should be queried once, regardless of how many items are in the list. I should only see one "genProcesses()" log line.

What did you see instead?

```
osquery> SELECT count(1) FROM processes WHERE path IN ('a','b','c','d','e','f','g','h');
I0121 13:09:52.299521 2920076160 processes.cpp:473] genProcesses
I0121 13:09:52.309139 2920076160 processes.cpp:473] genProcesses
```

Questions?

Thank You.

@packetzero

github.com/packetzero

Alex Malone
AT&T Cybersecurity