Tracing as a Service

stephan.brandauer@it.uu.se *
tobias.wrigstad@it.uu.se

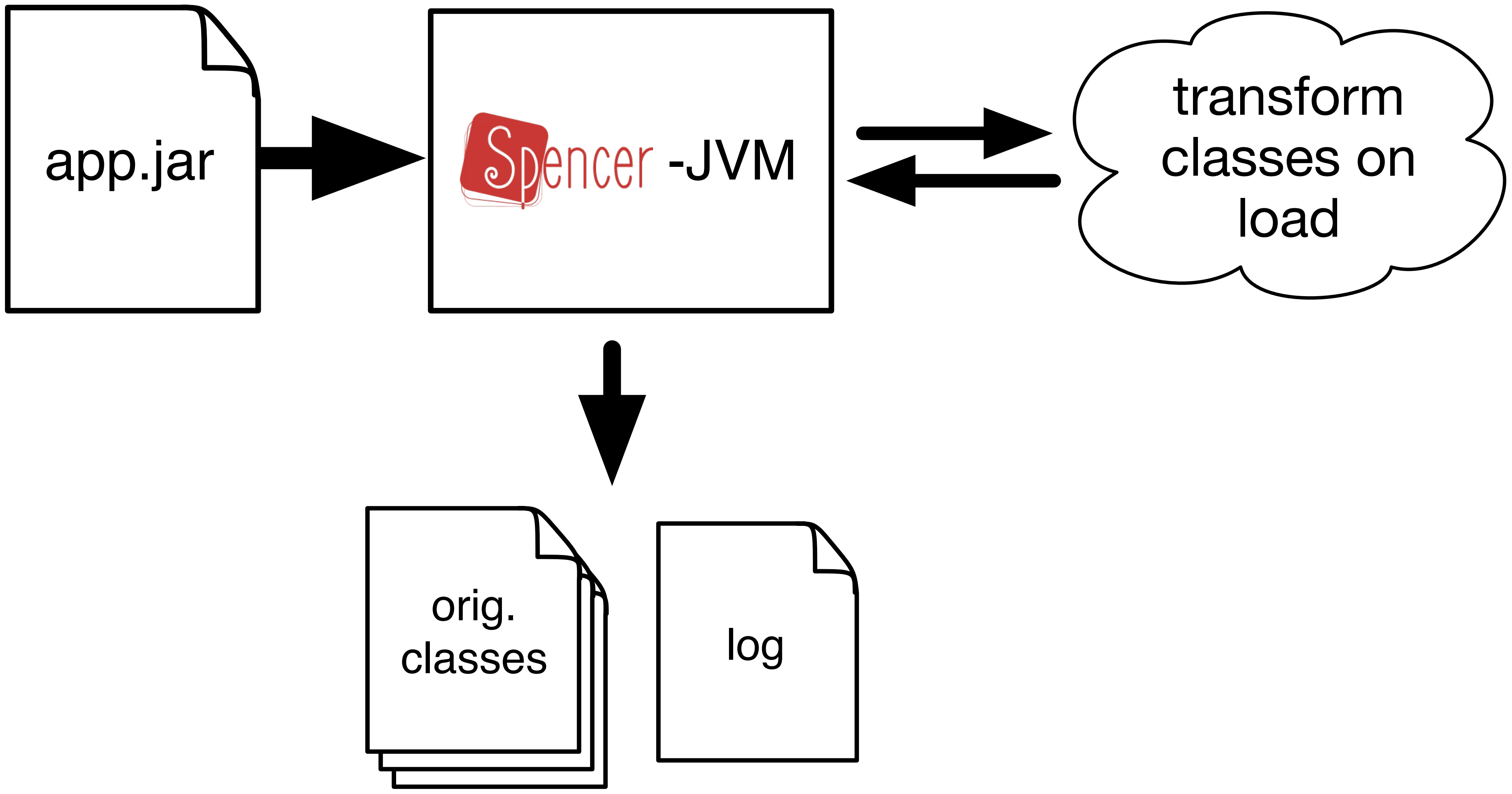
Spencer

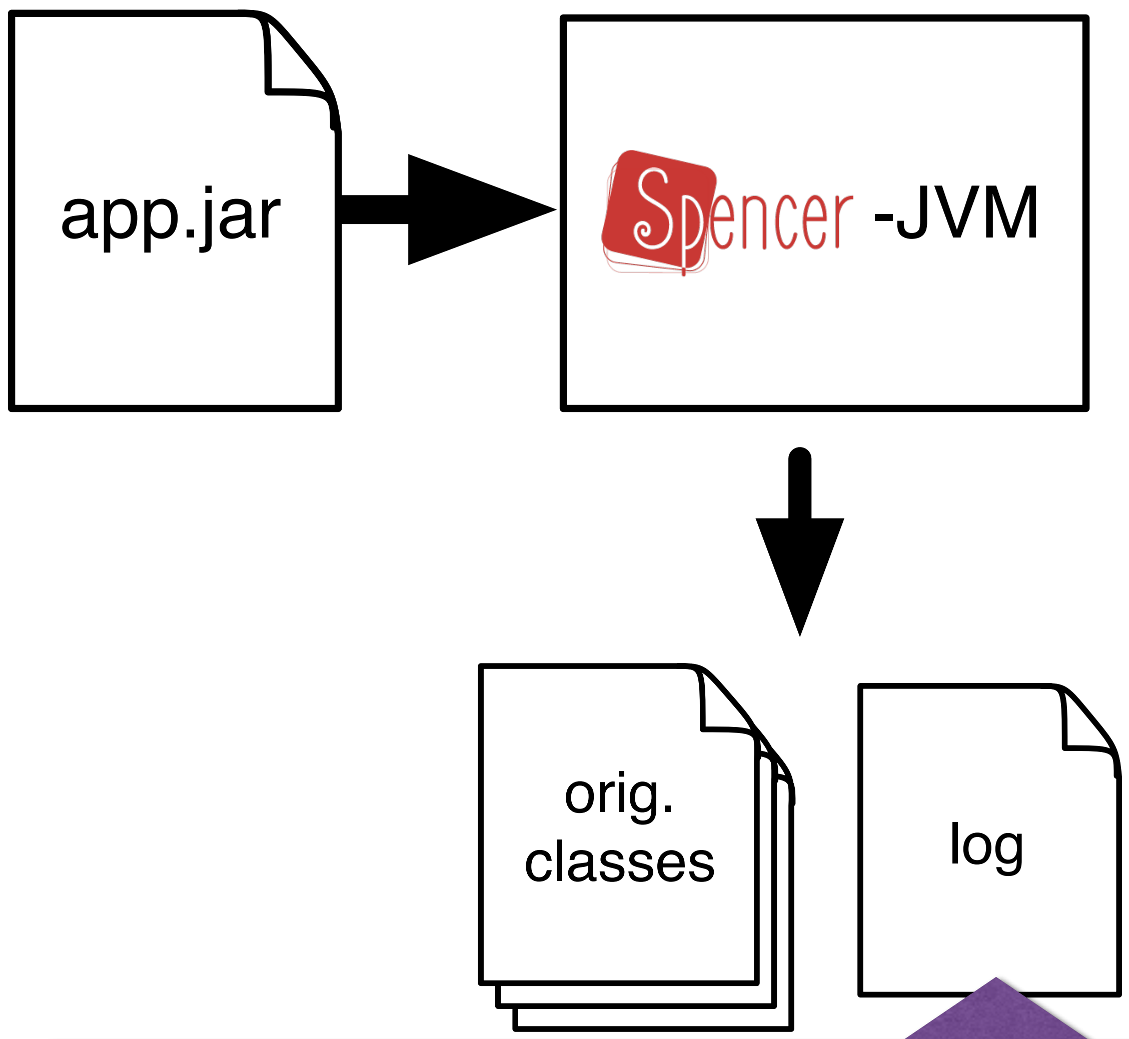
Web based service to query program traces.

“What do typical programs look like?”

Java bytecode

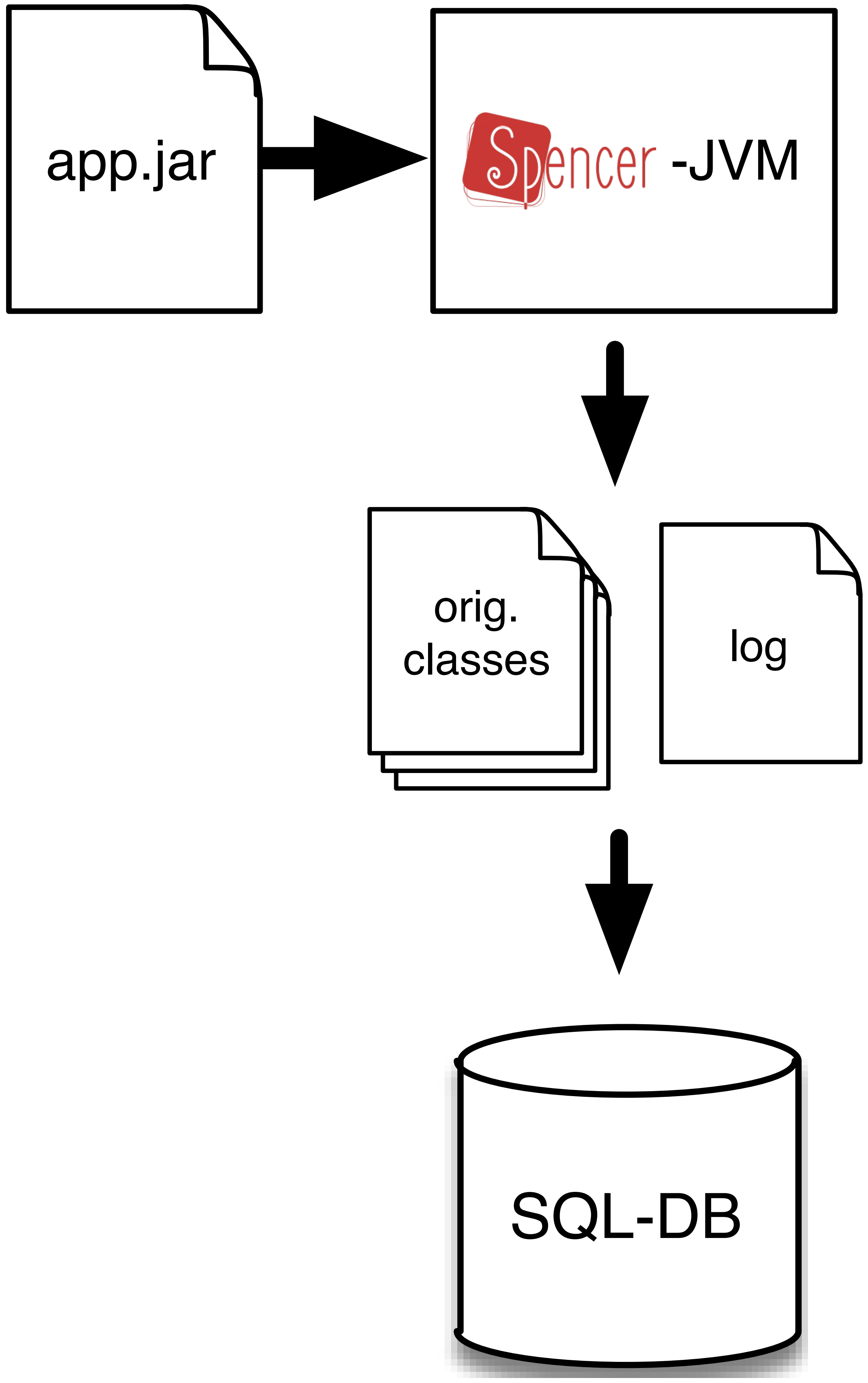
Workflow


app.jar
Spencer -JVM
transform classes on load
orig. classes
log



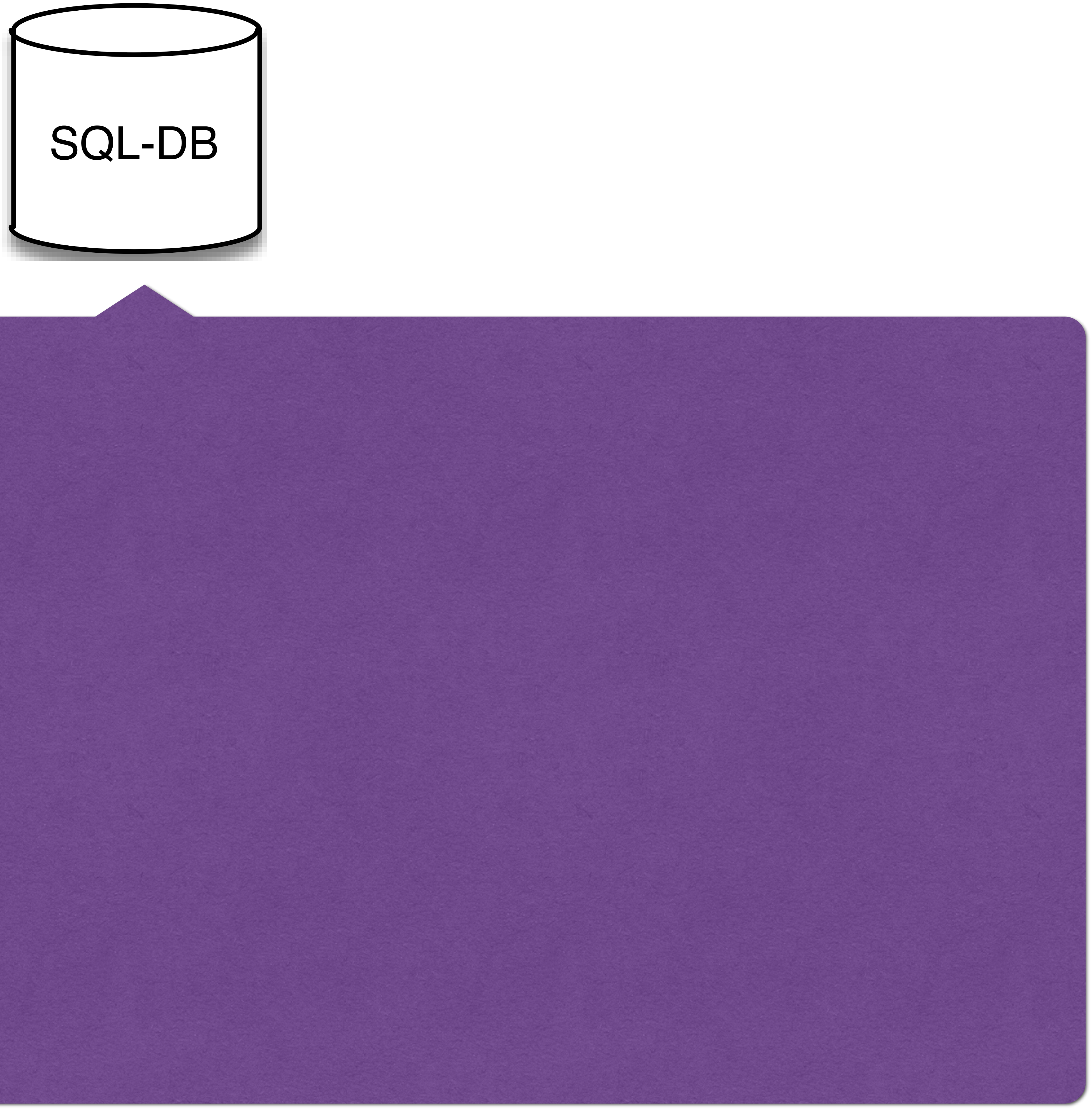
Unprocessed data

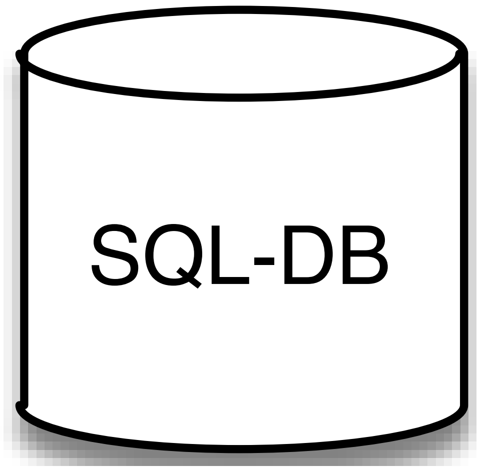
```
#511073: ⟹  (java/lang/String @ 10247) . startsWith(Ljava/lang/String;)Z , callsite=MetaIndex.java:242 , thread=main
#511074:   varstore - caller=java/lang/String :: startsWith @ 10247 var 1 , value was 0 , now is 10452 , thread=main
#511075:   varload - caller=java/lang/String @ 10247 , var 1 , val=10452 , thread=main
#511076:   ⟹  (java/lang/String @ 10247) . startsWith(Ljava/lang/String;I)Z , callsite=String.java:1434 , thread=main
#511077:     varstore - caller=java/lang/String :: startsWith @ 10247 var 1 , value was 0 , now is 10452 , thread=main
#511078:     fieldLoad - caller=java/lang/String :: startsWith @ 10247 , holder=java/lang/String @ 10247 , field=[C value , thread=main
#511079:     varstore - caller=java/lang/String :: startsWith @ 10247 var 3 , value was 0 , now is 10248 , thread=main
#511080:     varload - caller=java/lang/String @ 10247 , var 1 , val=10452 , thread=main
#511081:     fieldLoad - caller=java/lang/String :: startsWith @ 10247 , holder=java/lang/String @ 10452 , field=[C value , thread=main
#511082:     varstore - caller=java/lang/String :: startsWith @ 10247 var 5 , value was 0 , now is 10453 , thread=main
#511083:     varload - caller=java/lang/String @ 10247 , var 1 , val=10452 , thread=main
#511084:     fieldLoad - caller=java/lang/String :: startsWith @ 10247 , holder=java/lang/String @ 10452 , field=[C value , thread=main
#511085:     fieldLoad - caller=java/lang/String :: startsWith @ 10247 , holder=java/lang/String @ 10247 , field=[C value , thread=main
#511086:     varload - caller=java/lang/String @ 10247 , var 3 , val=10248 , thread=main
#511087:     readmodify - callee=[C @ 10248 , caller=java/lang/String @ 10247 reads _0
#511088:     varload - caller=java/lang/String @ 10247 , var 5 , val=10453 , thread=main
#511089:     readmodify - callee=[C @ 10453 , caller=java/lang/String @ 10247 reads _0
#511090:   ⟸ (??? @ java/lang/String) . startsWith(???), thread=main
#511091: ⟸ (??? @ java/lang/String) . startsWith(???), thread=main
```



Unprocessed data

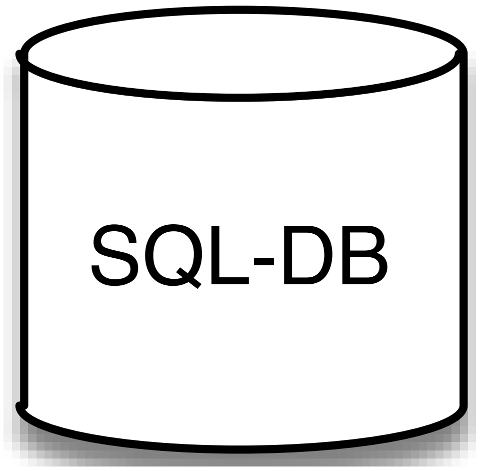
preprocess+load


SQL-DB



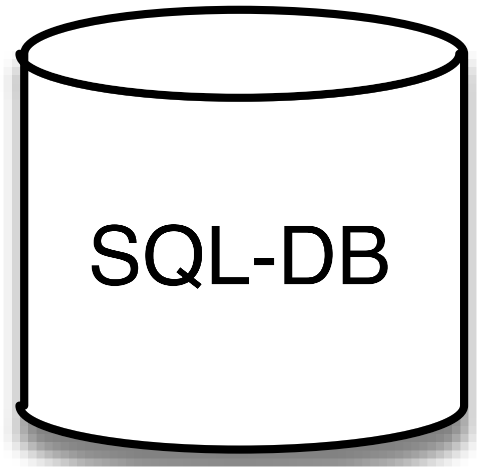
calls ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile   | callsiteline | thread
--------+--------+------------+-----------+---------+-----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java  |          242 | main
```



calls ✓
uses ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |   callsitefile   | callsiteline | thread
--------+--------+------------+-----------+---------+------------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java |          242 | main


# SELECT * FROM uses WHERE idx ⩾ 511073 AND idx ⩽ 511091 ;
 caller | callee | name  |   method   |   kind     |  idx   | thread
--------+--------+-------+------------+------------+--------+--------
  10247 |  10247 | var_1 | startsWith | varstore   | 511074 | main
  10247 |  10247 | var_1 | startsWith | varload    | 511075 | main
  … snip …
  10247 |  10247 | var_5 | startsWith | varload    | 511088 | main
  10247 |  10453 | _0    | startsWith | read       | 511089 | main
```



calls ✓
uses ✓
refs ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile   | callsiteline | thread
--------+--------+------------+-----------+---------+-----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java  |          242 | main


# SELECT * FROM uses WHERE idx ⩾ 511073 AND idx ⩽ 511091 ;
 caller | callee | name  |   method   |   kind    |  idx   | thread
--------+--------+-------+------------+-----------+--------+--------
  10247 |  10247 | var_1 | startsWith | varstore  | 511074 | main
  10247 |  10247 | var_1 | startsWith | varload   | 511075 | main
  … snip …
  10247 |  10247 | var_5 | startsWith | varload   | 511088 | main
  10247 |  10453 | _0    | startsWith | read      | 511089 | main


# SELECT * FROM refs WHERE caller = 10247 AND kind = 'field' ;
 caller | callee | kind  | name  | refstart | refend | thread
--------+--------+-------+-------+----------+--------+--------
  10247 |  10248 | field | value |   421877 |        | main
```

Queries

- Spencer uses a query DSL.
    - Compiled to SQL, and cached.
    - Makes caching effective.
    - Easier to use.
    - No **NEED** for SQL — but better performance.
- PostgreSQL is surprisingly expressive!

ImmutableObj()

```
SELECT id FROM objects WHERE id > 4
EXCEPT
  (SELECT DISTINCT callee AS id
   FROM uses_cstore
   WHERE callee > 4
   AND   NOT(caller = callee AND method = ‘<init>')
   AND   (kind = 'fieldstore' OR kind = 'modify'))
```

HeapDeeply(ImmutableObj())

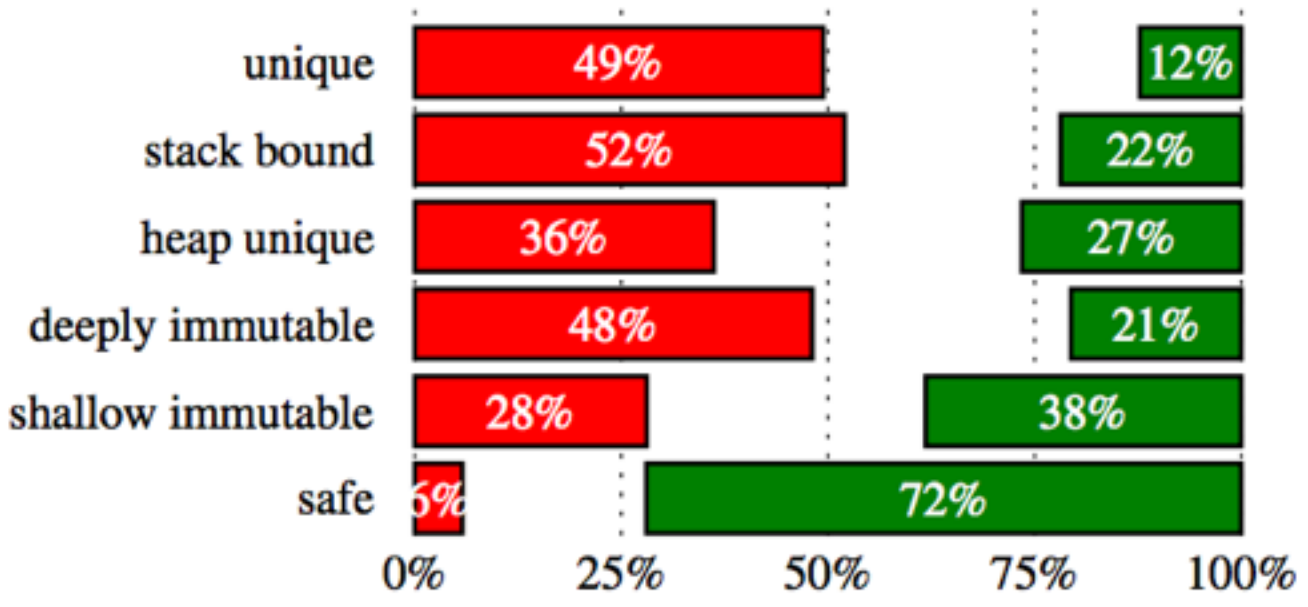
```
(
  SELECT id FROM objects WHERE id > 4
EXCEPT
  (SELECT DISTINCT callee AS id
FROM uses_cstore
WHERE
  callee > 4 AND
  NOT(caller = callee AND method = '<init>') AND
  (kind = 'fieldstore' OR kind = 'modify'))

) INTERSECT (
  SELECT id FROM objects WHERE id > 4
EXCEPT
  (WITH RECURSIVE canheapreach(id) AS (
    SELECT id FROM objects WHERE id > 4
EXCEPT
  (SELECT id FROM objects WHERE id > 4
EXCEPT
…
```

```
…
 (SELECT DISTINCT callee AS id
FROM uses_cstore
WHERE
  callee > 4 AND
  NOT(caller = callee AND method = '<init>') AND
  (kind = 'fieldstore' OR kind = 'modify'))
      )

  UNION
    SELECT
      refs.caller AS id
    FROM refs
    JOIN canheapreach ON canheapreach.id = refs.callee
    WHERE kind = 'field'
)
SELECT id FROM canheapreach)

)
```

API + Meta Info

- There exists an API that gives you results (formatted as JSON objects)
- http://www.spencer-t.racing/doc/api
- Meta info: “per-object tags”
  - class
  - allocation site + “time”
  - coming: number of reads, writes
  - I’m happy to add more

Example from API



(g) This shows, for each property, the average across data sets of the proportion of classes that *only* (green, right)/*never* (red, left) produced instances that had the property. We infer, unsoundly, that there exists an invariant that guarantees the property holds statically. Classes with less than 10 instances are ignored.

Status

- Tracing tool implementation: done, modulo maintenance

- Analysis DSL: useful for some use cases. New uses through extension.

- Web interface: work in progress, has performance issues

Sources

- http://spencer-t.racing
- "Spencer: Interactive Heap Analysis for the Masses", to appear (International Conference on Mining Software Repositories)
- "Mining for Safety using Interactive Trace Analysis", to appear (Workshop on Quantitative Aspects of Programming Languages and Systems)

Questions?