How to test software without writing tests.

Manuel Rigger

ETH Zurich, Switzerland

@RiggerManuel

https://www.manuelrigger.at/

SQLancer

☆ 921 stars 138 forks

SQLANCER

https://github.com/sqlancer

SQLancer implements new techniques for testing DBMSs

SQLancer

899 stars 129 forks

General insights on how you can apply automated testing techniques on your project

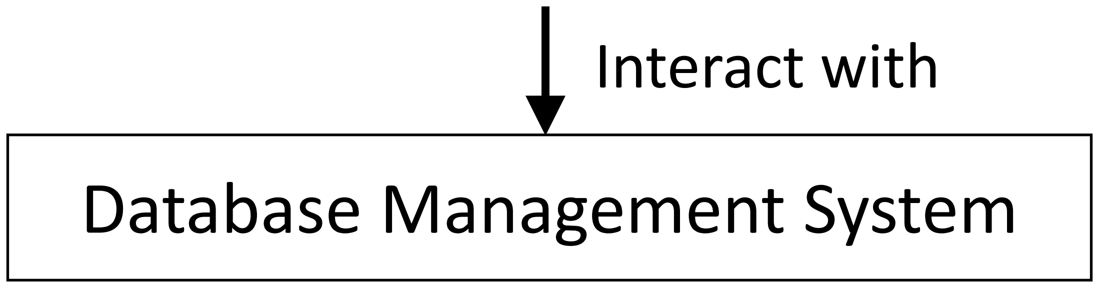
SQLancer implements new techniques for testing DBMSs

Database Management Systems (DBMS)

$$\begin{tikzcd}
\textbf{Structured Query Language (SQL)} \arrow[d, "\text{Interact with}"] \\
\boxed{\text{Database Management System}}
\end{tikzcd}$$

Database Management Systems (DBMS)

```sql
CREATE TABLE t0(c0 INT);
INSERT INTO t0 VALUES (-1) ;
SELECT * FROM t0 WHERE <p>;
```



How can we write test cases
for Database Management Systems?

How to manually test DBMSs?

zlob_print.test

```
--source include/have_debug.inc
--source include/have_innodb_max_16k.inc

set global innodb_compression_level = 0;
create table t1 (f1 int primary key, f2 longblob)
   row_format=compressed, en
set debug='+d,innodb_zlob_pri
insert into t1 values (1, repeat(
set debug='-d,innodb_zlob_pri
select f1, right(f2, 40) from t1;
drop table t1;
set global innodb_compression_level = default;
```

zlob_print.result

```
set global innodb_compression_level = 0;
create table t1 (f1 int primary key, f2 longblob)
row_format=compressed, engine=innodb;
set debug='+d,innodb_zlob_print';
insert into t1 values (1, repeat('+', 1048576));



                     +++++++++++++++++++

set global innodb_compression_level = default;
```

It is challenging and time-consuming to write manual tests for large software systems

https://github.com/mysql/mysql-server/blob/8.0/mysql-test/suite/innodb/t/zlob_print.test

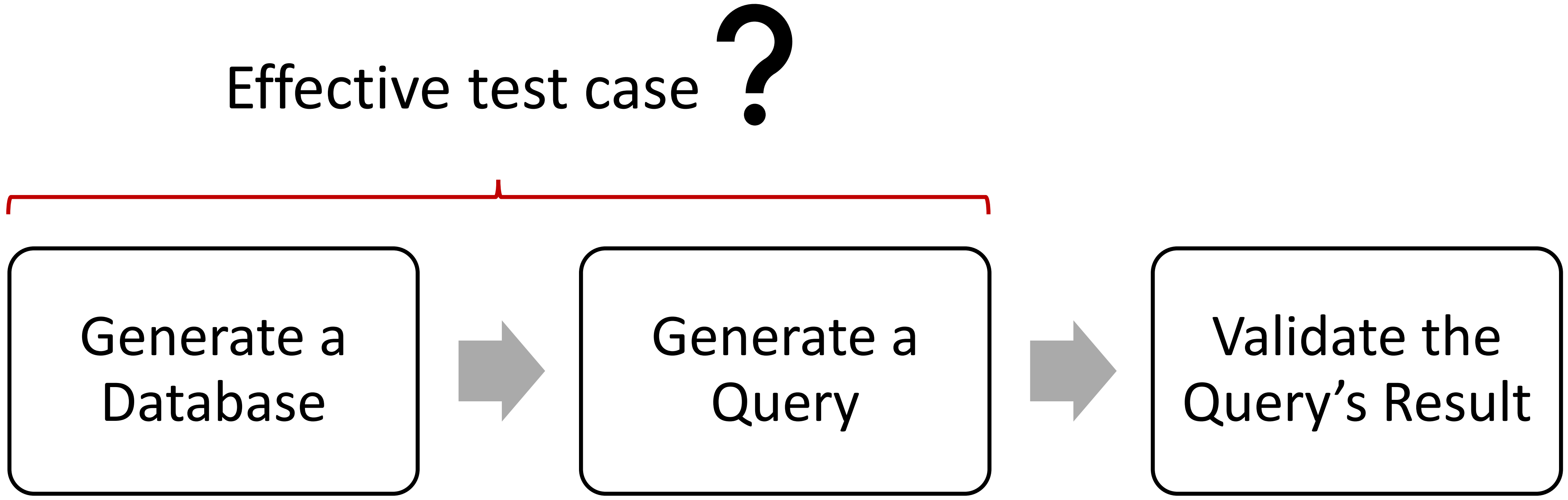
https://github.com/mysql/mysql-server/blob/8.0/mysql-test/suite/innodb/r/zlob_print.result

ETH zürich

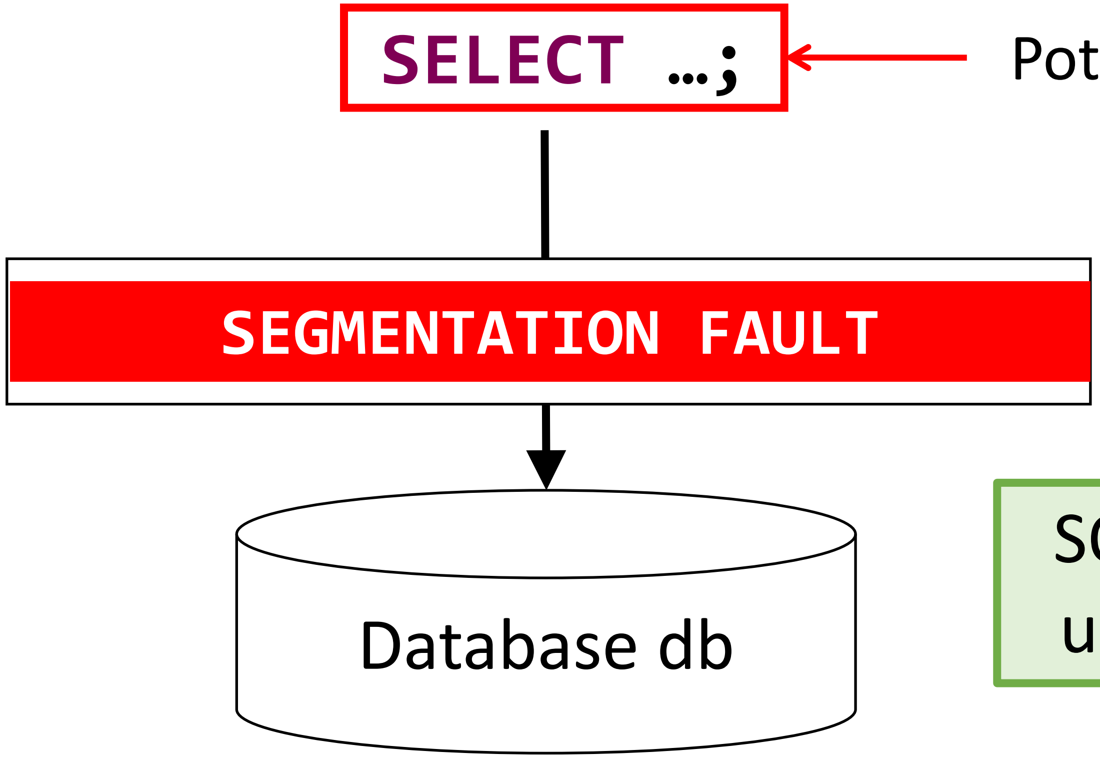
?

Can we automate the testing process?

Automatic Testing Core Challenges



SQLsmith



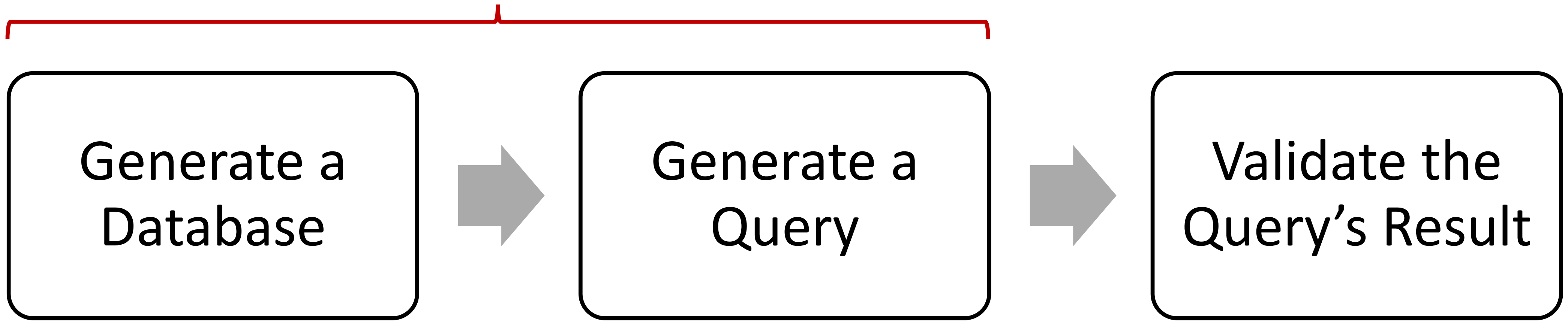
Potentially highly complex

SQLsmith is an effective and widely-used random query generation tool

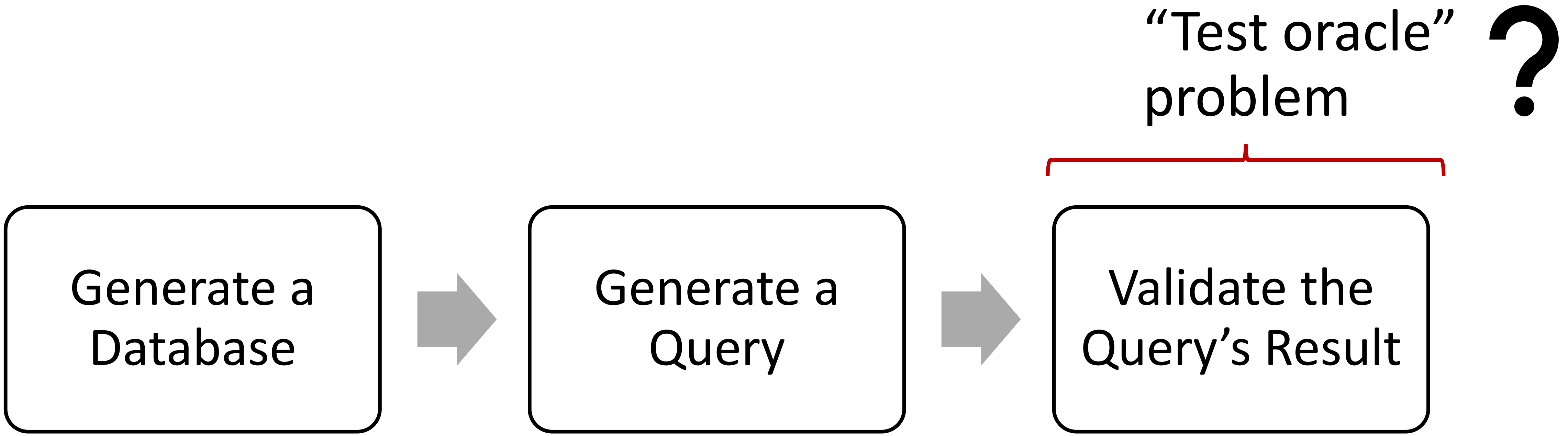
https://github.com/anse1/sqlsmith

Automatic Testing Core Challenges

Use a random-generation approach to automatically generate tests ✓



Automatic Testing Core Challenges



Test Oracle

Incorrect result!

Difficult, even for manually-written test cases!

"a test oracle (or just oracle) is a mechanism for determining whether a test has passed or failed"

https://en.wikipedia.org/wiki/Test_oracle

ETH zürich

Motivating Example

t0

c0
0.0

t1

c0
-0.0

```
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

$\longrightarrow$?

It might seem **disputable** whether the predicate should evaluate to `true`

Motivating Example

t0

c0
0.0

t1

c0
-0.0

```
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

false?

t0.c0	t1.c0

0	0	0	0	0	…	0	0	0	0
-0	1	0	0	0	…	0	0	0	0

Different binary representation

Motivating Example

t0

c0
0.0

t1

c0
-0.0

true?

Evaluates to `true` for **most programming languages**

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

MySQL

t0.c0	t1.c0
0	-0

Motivating Example

t0

c0
0.0

t1

c0
-0.0

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

MySQL

t0.c0	t1.c0
0	-0

✓

Motivating Example

t0

c0
0.0

t1

c0
-0.0

The latest version of MySQL that we tested failed to fetch the row

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

$\longrightarrow$

t0.c0	t1.c0

✗

https://bugs.mysql.com/bug.php?id=99122

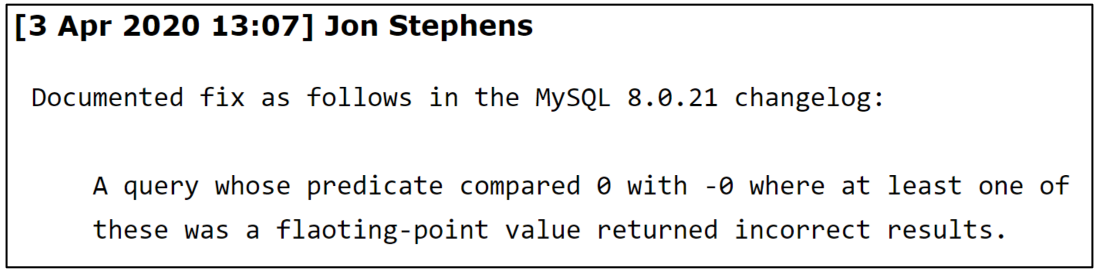
Motivating Example

t0

c0
0.0

t1

c0
-0.0



[3 Apr 2020 13:07] Jon Stephens

```
Documented fix as follows in the MySQL 8.0.21 changelog:

    A query whose predicate compared 0 with -0 where at least one of
    these was a flaoting-point value returned incorrect results.
```

MySQL

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

$\longrightarrow$

t0.c0	t1.c0

✗

https://bugs.mysql.com/bug.php?id=99122

Motivating Example

t0

c0
`0.0`

t1

c0
`-0.0`

We could **find the bug without having an accurate understanding** ourselves

MySQL

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

t0.c0	t1.c0

✗

https://bugs.mysql.com/bug.php?id=99122

ETH zürich

Motivating Example

t0

c0
0.0

t1

c0
-0.0

```
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

$\longrightarrow$

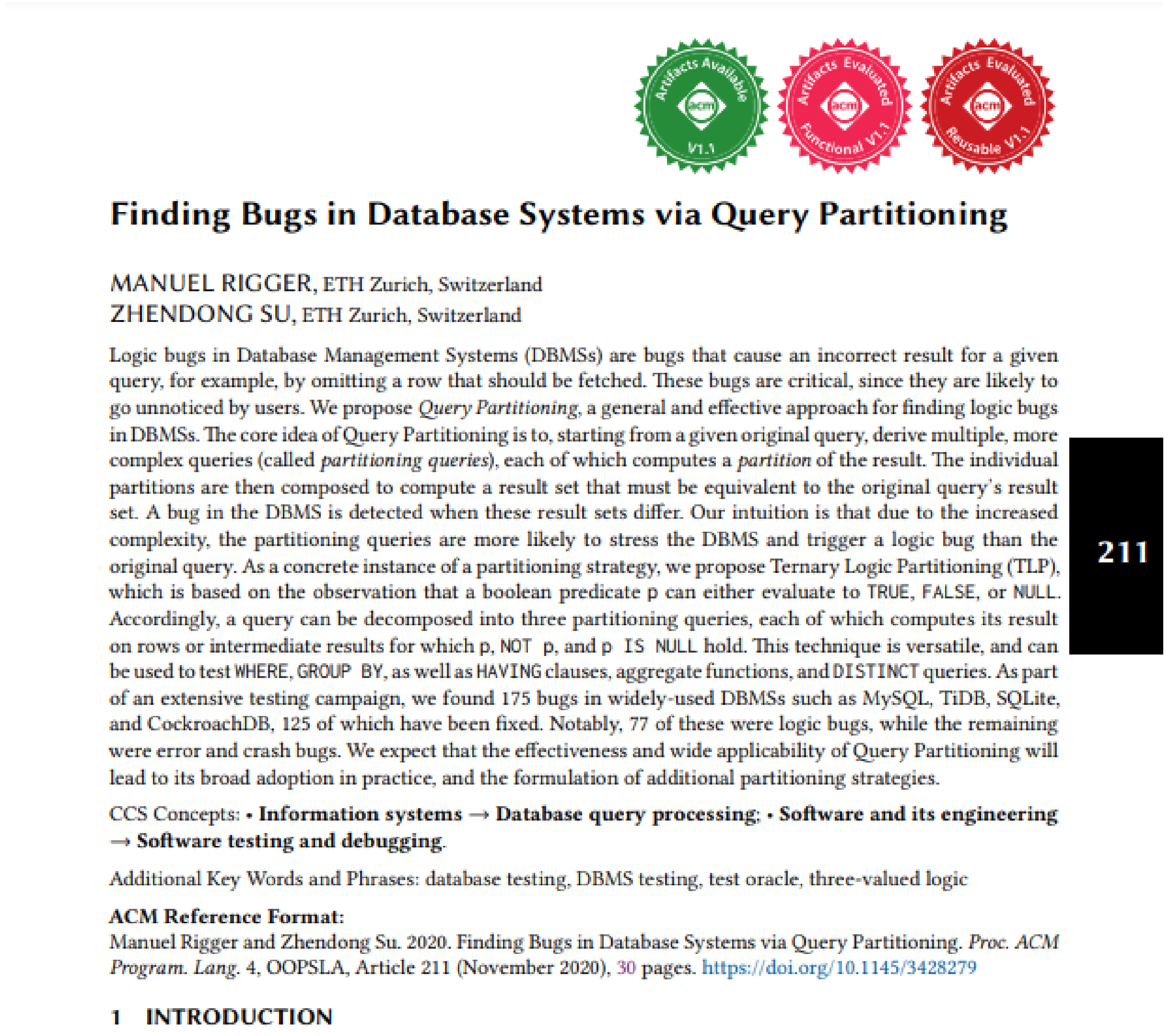
t0.c0	t1.c0

Incorrect result!

https://bugs.mysql.com/bug.php?id=99122

ETH zürich

Ternary Logic Partitioning (TLP)



Finding Bugs in Database Systems via Query Partitioning

MANUEL RIGGER, ETH Zurich, Switzerland
ZHENDONG SU, ETH Zurich, Switzerland

Logic bugs in Database Management Systems (DBMSs) are bugs that cause an incorrect result for a given query, for example, by omitting a row that should be fetched. These bugs are critical, since they are likely to go unnoticed by users. We propose *Query Partitioning*, a general and effective approach for finding logic bugs in DBMSs. The core idea of Query Partitioning is to, starting from a given original query, derive multiple, more complex queries (called *partitioning queries*), each of which computes a *partition* of the result. The individual partitions are then composed to compute a result set that must be equivalent to the original query's result set. A bug in the DBMS is detected when these result sets differ. Our intuition is that due to the increased complexity, the partitioning queries are more likely to stress the DBMS and trigger a logic bug than the original query. As a concrete instance of a partitioning strategy, we propose Ternary Logic Partitioning (TLP), which is based on the observation that a boolean predicate p can either evaluate to `TRUE`, `FALSE`, or `NULL`. Accordingly, a query can be decomposed into three partitioning queries, each of which computes its result on rows or intermediate results for which p, `NOT p`, and `p IS NULL` hold. This technique is versatile, and can be used to test `WHERE`, `GROUP BY`, as well as `HAVING` clauses, aggregate functions, and `DISTINCT` queries. As part of an extensive testing campaign, we found 175 bugs in widely-used DBMSs such as MySQL, TiDB, SQLite, and CockroachDB, 125 of which have been fixed. Notably, 77 of these were logic bugs, while the remaining were error and crash bugs. We expect that the effectiveness and wide applicability of Query Partitioning will lead to its broad adoption in practice, and the formulation of additional partitioning strategies.

211

CCS Concepts: • **Information systems → Database query processing**; • **Software and its engineering → Software testing and debugging**.

Additional Key Words and Phrases: database testing, DBMS testing, test oracle, three-valued logic

ACM Reference Format:
Manuel Rigger and Zhendong Su. 2020. Finding Bugs in Database Systems via Query Partitioning. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 211 (November 2020), 30 pages. https://doi.org/10.1145/3428279

1 INTRODUCTION

https://dl.acm.org/doi/abs/10.1145/3428279

My Research

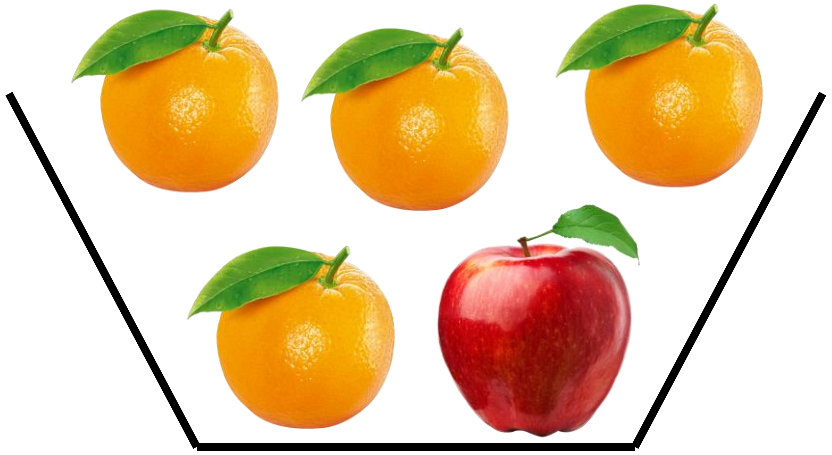
Idea: test the DBMS against itself

Incorrect result!

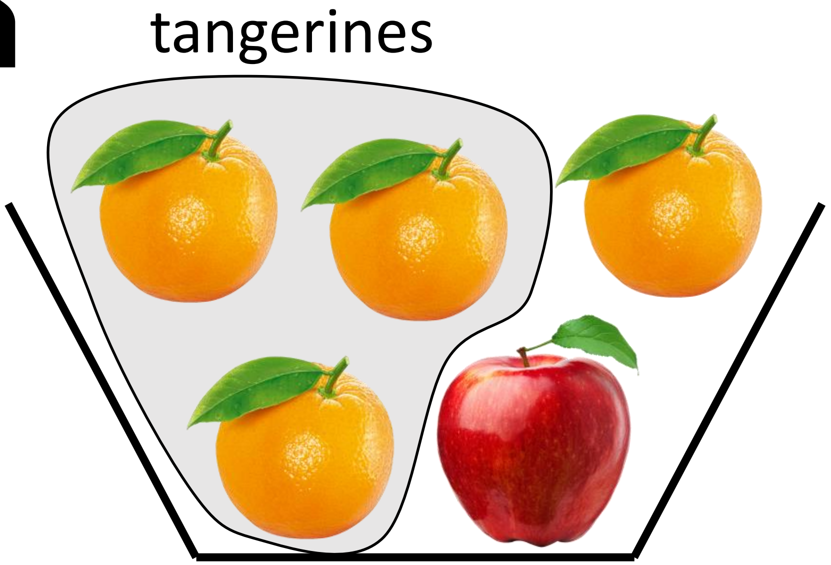
ETH zürich

Scenario: Coffee Kitchen

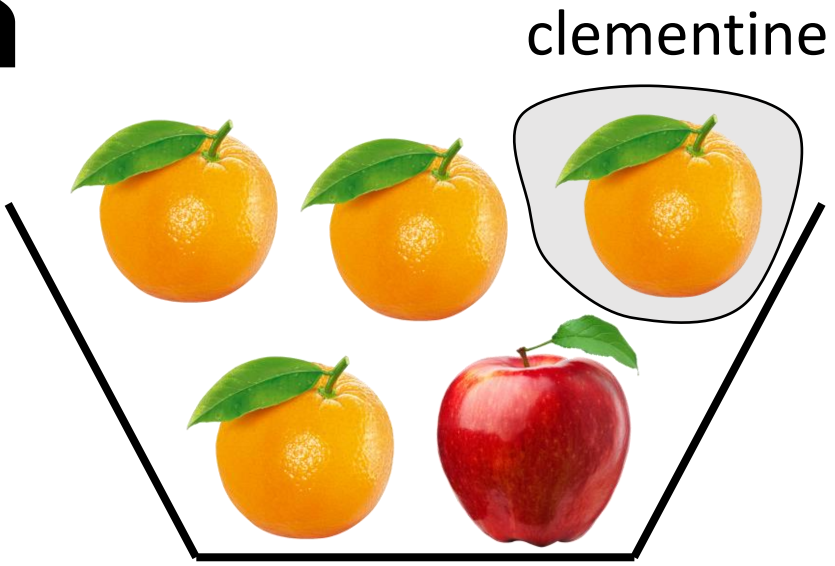
Tangerines vs. Clementines



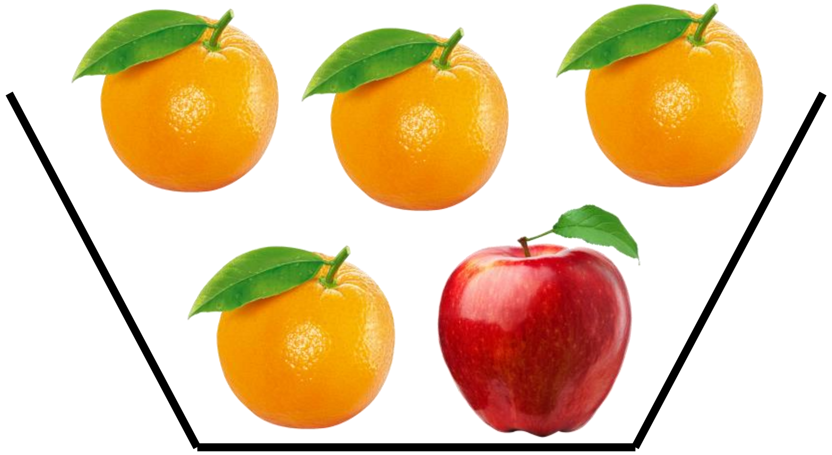
Tangerines vs. Clementines



Tangerines vs. Clementines

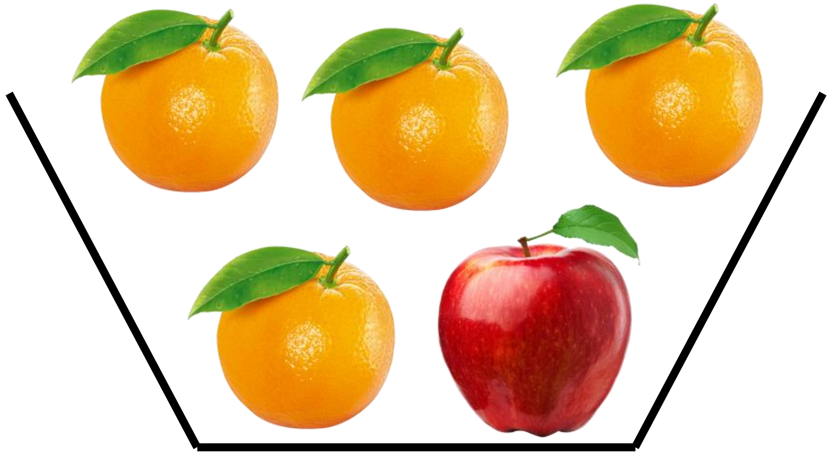


Tangerines vs. Clementines

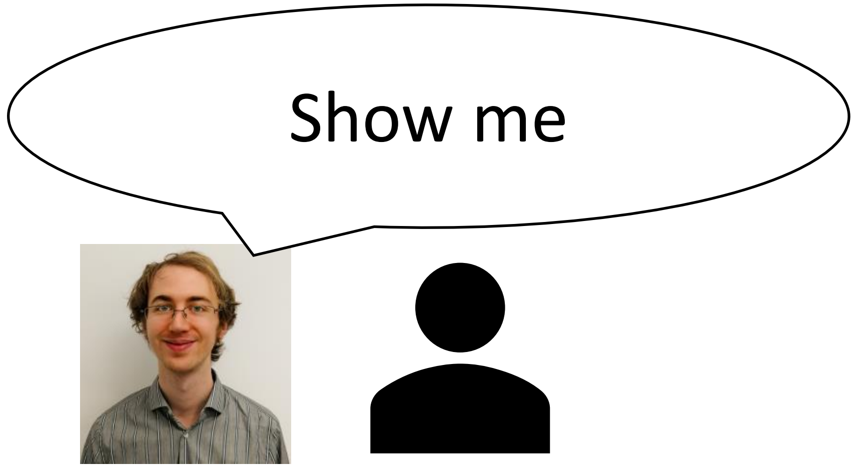


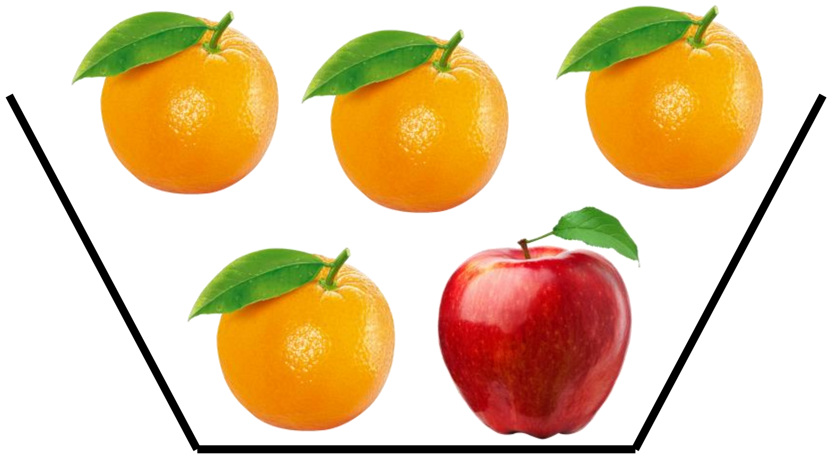
Tangerines vs. Clementines

I can never tell different citrus fruits apart



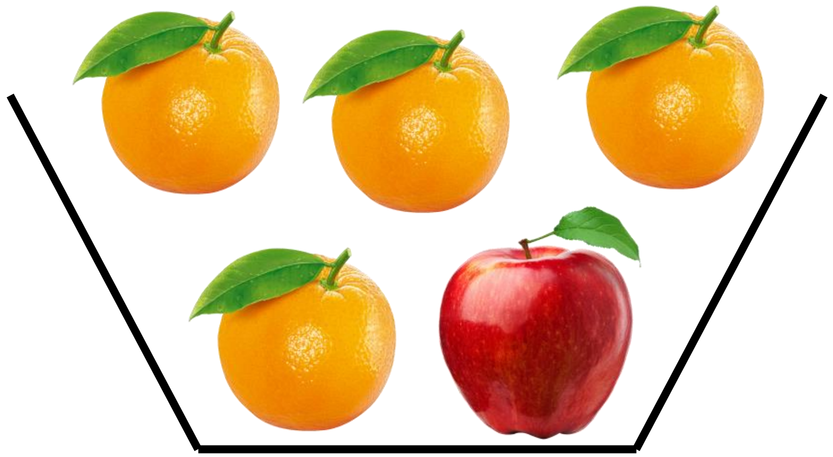
Tangerines vs. Clementines



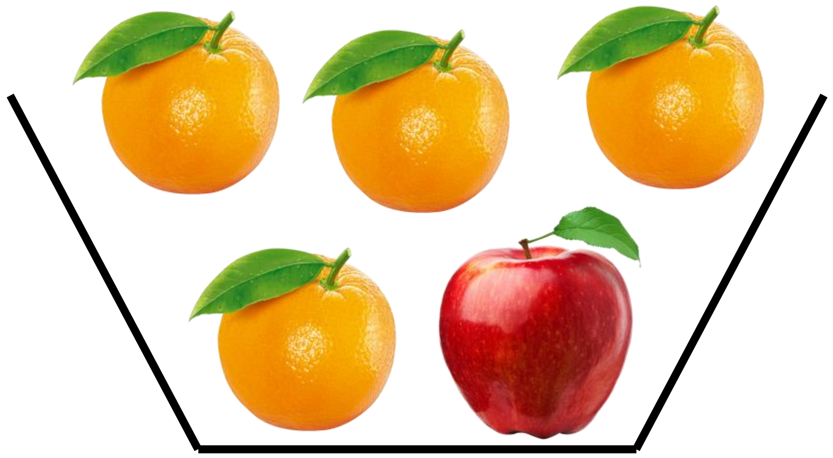


Tangerines vs. Clementines

2 fruits

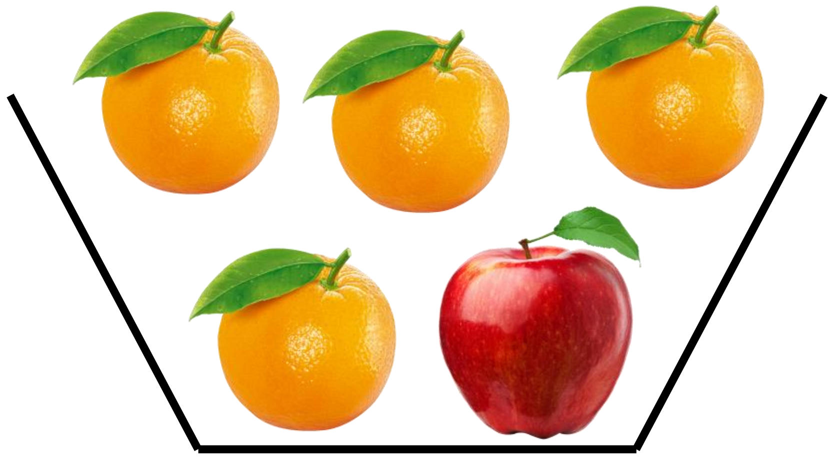


Tangerines vs. Clementines

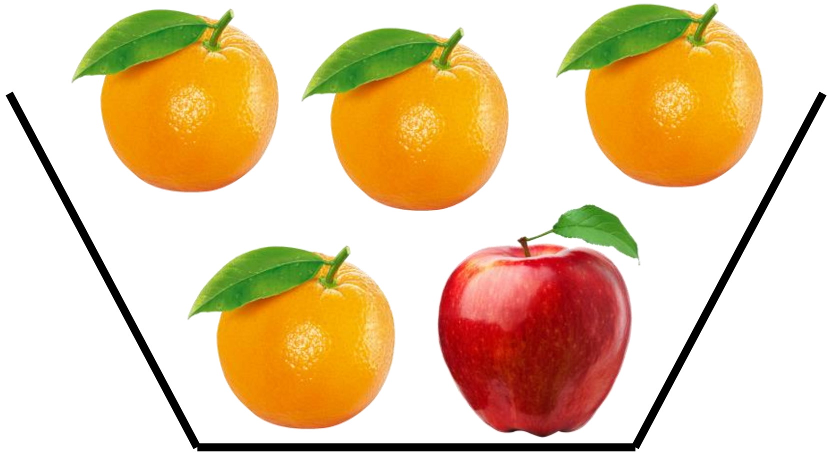


Tangerines vs. Clementines

4 fruits



Tangerines vs. Clementines



2 fruits

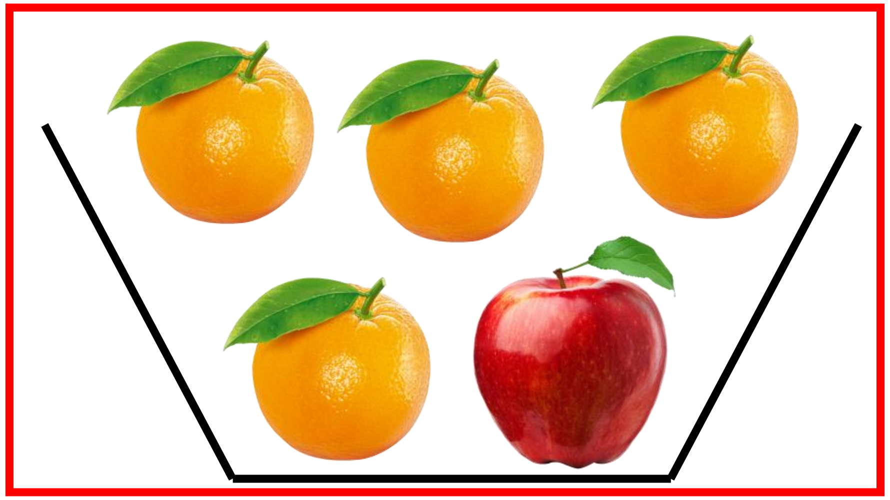
4 fruits

6 fruits

Tangerines vs. Clementines

You likely classified a fruit as **both** a tangerine and a clementine!

5 fruits



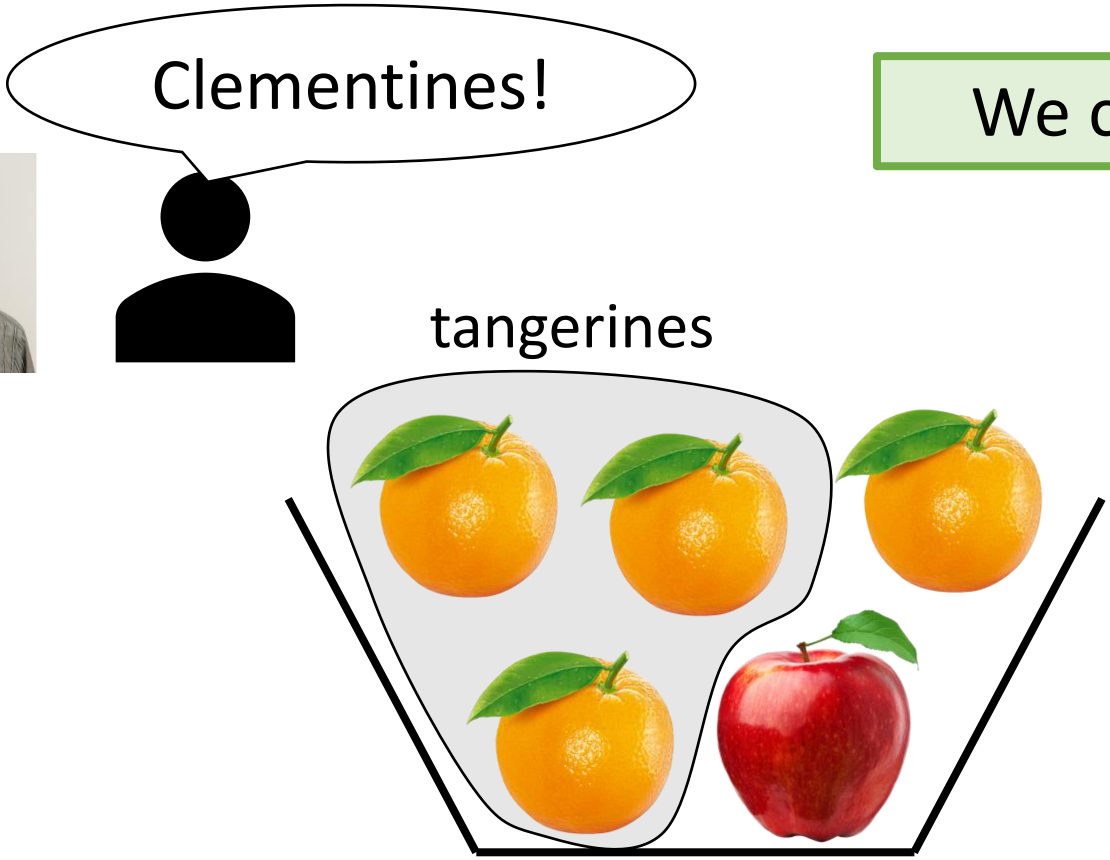
2 fruits
4 fruits

6 fruits

Insight

Insight: Every object in a (mathematical) universe is either **a clementine** or **not a clementine**

Tangerines vs. Clementines



We cannot find all bugs

Ternary Logic

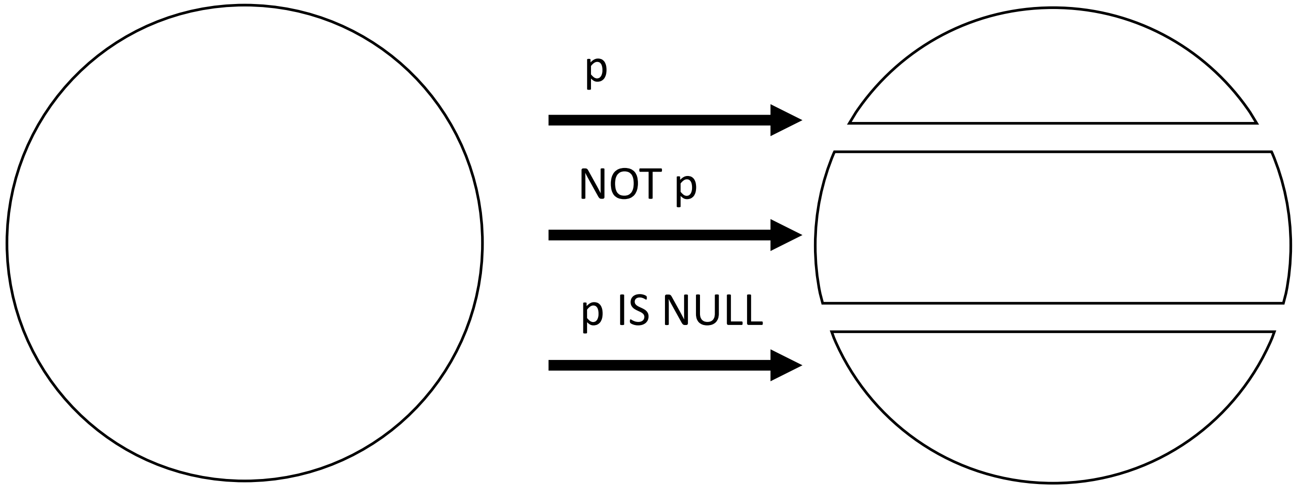
Consider a predicate p and a given row r.
Exactly **one** of the following must hold:

- p
- NOT p
- p IS NULL

Ternary Logic

Consider a predicate p and a given row r.
Exactly **one** of the following must hold:

- p
- NOT p
- p IS NULL



Motivating Example

t0

c0
0.0

t1

c0
-0.0

How did this insight allow us to detect this bug?

```sql
SELECT * FROM t0, t1
WHERE t0.c0 = t1.c0;
```

MySQL

t0.c0	t1.c0

✗

Example: MySQL

```
SELECT * FROM t0, t1;
```

t0.c0	t1.c0
0.0	-0.0

p

```
SELECT * FROM t0, t1 WHERE t0.c0=t1.c0
UNION ALL
SELECT * FROM t0, t1 WHERE NOT (t0.c0=t1.c0)
UNION ALL
SELECT * FROM t0, t1 WHERE (t0.c0=t1.c0) IS NULL;
```

t0.c0	t1.c0

≠

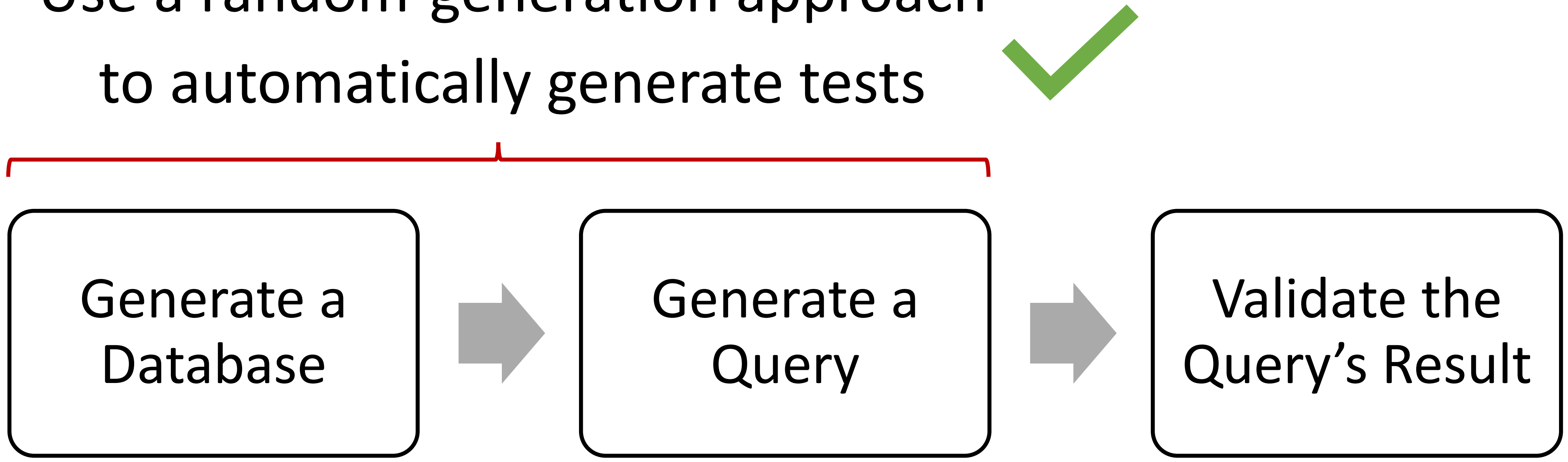
https://bugs.mysql.com/bug.php?id=99122

Scope

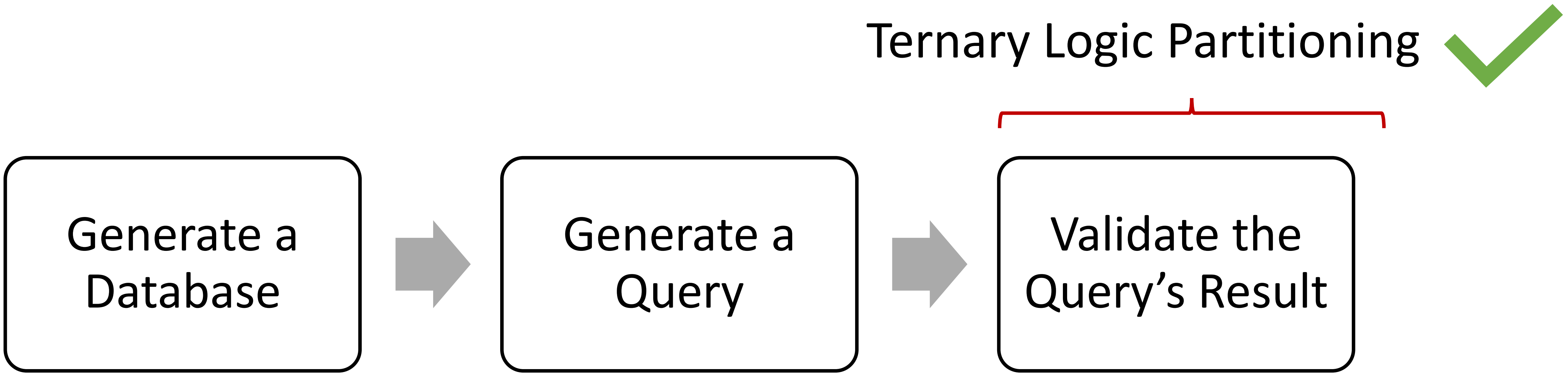
- WHERE
- GROUP BY
- HAVING
- DISTINCT queries
- Aggregate functions

Automatic Testing Core Challenges

Use a random-generation approach to automatically generate tests



Automatic Testing Core Challenges



?

Can such a simple technique be effective?

More than 450 Bugs

I used SQLancer to find **over 450 unique, previously unknown bugs** in widely-used DBMSs

What should I take away from this talk?

Generalizing the Findings

Insight: While the specific technique works primarily for data-oriented systems, it is based on a more general technique

Generalizing the Technique

```
SELECT * FROM t0, t1;
```

$\xrightarrow{\text{MySQL}}$

t0.c0	t1.c0
0.0	-0.0

$\underbrace{\hspace{20em}}_{\text{Derive}}$

```
SELECT * FROM t0, t1 WHERE t0.c0=t1.c0
UNION ALL
SELECT * FROM t0, t1 WHERE NOT (t0.c0=t1.c0)
UNION ALL
SELECT * FROM t0, t1 WHERE (t0.c0=t1.c0) IS NULL;
```

+

Generalizing the Technique

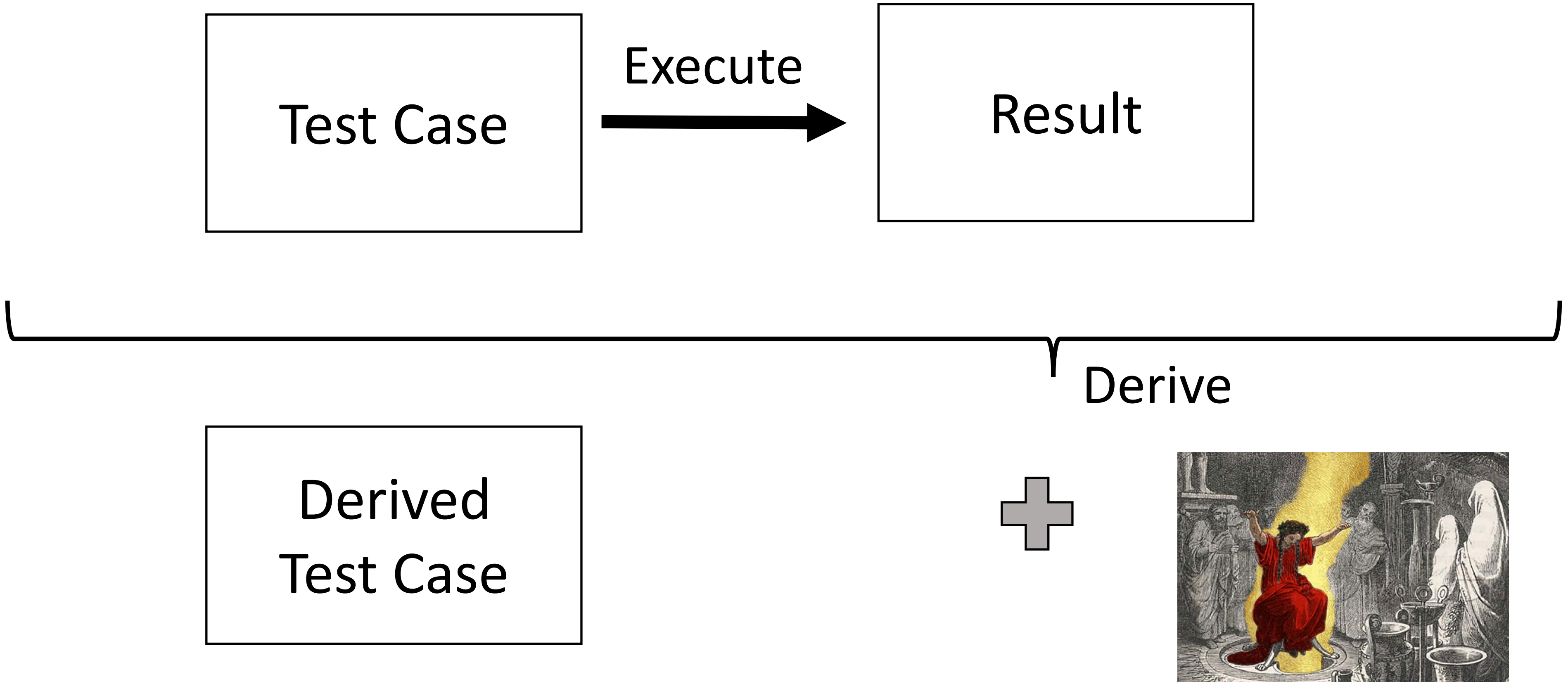
$$\begin{tikzcd}\boxed{\text{Test Case}} \arrow[r, "\text{Execute}"] & \boxed{\text{Result}}\end{tikzcd}$$

Derive

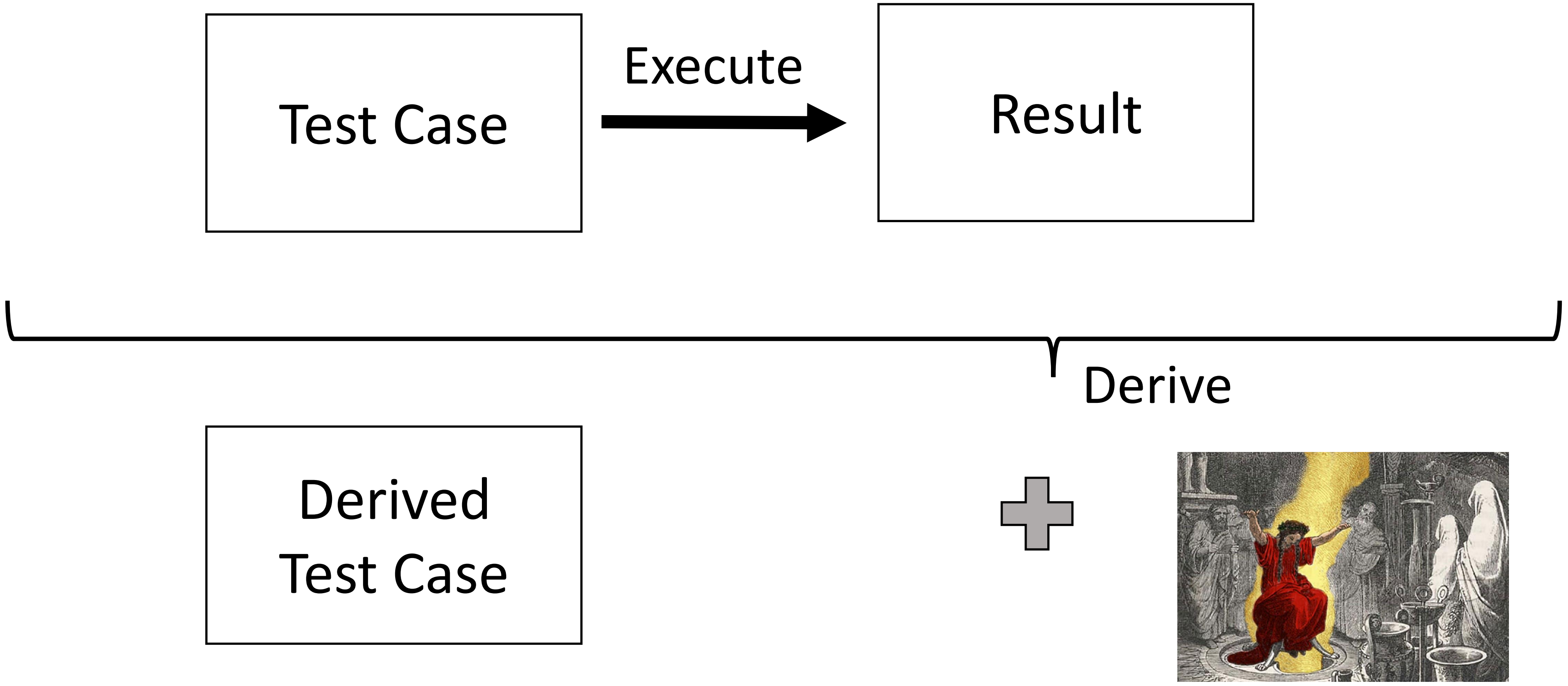
```
SELECT * FROM t0, t1 WHERE t0.c0=t1.c0
UNION ALL
SELECT * FROM t0, t1 WHERE NOT (t0.c0=t1.c0)
UNION ALL
SELECT * FROM t0, t1 WHERE (t0.c0=t1.c0) IS NULL;
```

+

Generalizing the Technique

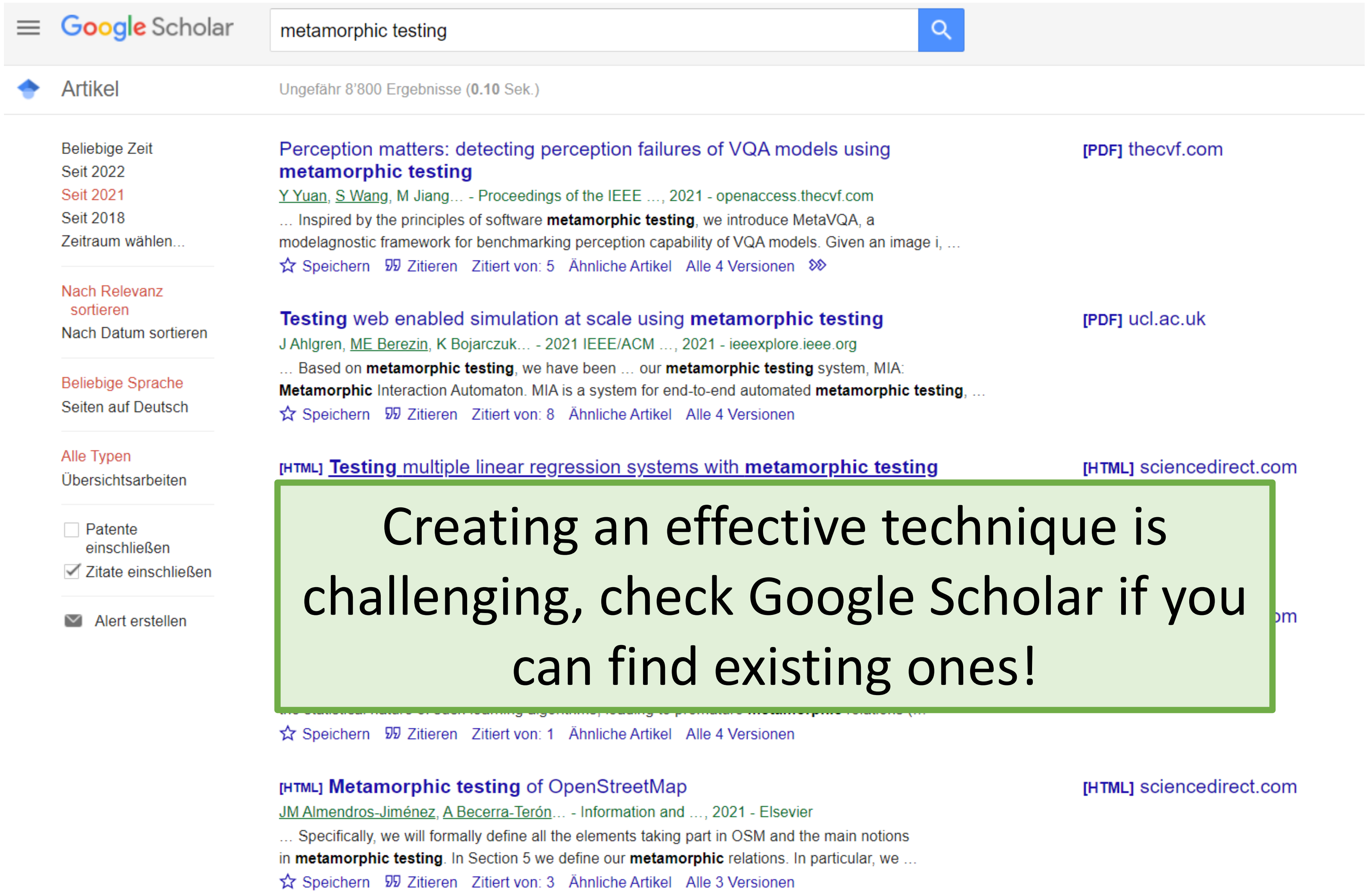


Generalizing the Technique



The general concept is known as **metamorphic testing**

Metamorphic Testing



Google Scholar

metamorphic testing

Artikel

Ungefähr 8'800 Ergebnisse (**0.10** Sek.)

Beliebige Zeit
Seit 2022
Seit 2021
Seit 2018
Zeitraum wählen...

Nach Relevanz sortieren
Nach Datum sortieren

Beliebige Sprache
Seiten auf Deutsch

Alle Typen
Übersichtsarbeiten

Patente einschließen
✓ Zitate einschließen

Alert erstellen

Perception matters: detecting perception failures of VQA models using **metamorphic testing** — [PDF] thecvf.com
Y Yuan, S Wang, M Jiang... - Proceedings of the IEEE ..., 2021 - openaccess.thecvf.com
... Inspired by the principles of software **metamorphic testing**, we introduce MetaVQA, a modelagnostic framework for benchmarking perception capability of VQA models. Given an image i, ...
Speichern Zitieren Zitiert von: 5 Ähnliche Artikel Alle 4 Versionen

Testing web enabled simulation at scale using **metamorphic testing** — [PDF] ucl.ac.uk
J Ahlgren, ME Berezin, K Bojarczuk... - 2021 IEEE/ACM ..., 2021 - ieeexplore.ieee.org
... Based on **metamorphic testing**, we have been ... our **metamorphic testing** system, MIA: **Metamorphic** Interaction Automaton. MIA is a system for end-to-end automated **metamorphic testing**, ...
Speichern Zitieren Zitiert von: 8 Ähnliche Artikel Alle 4 Versionen

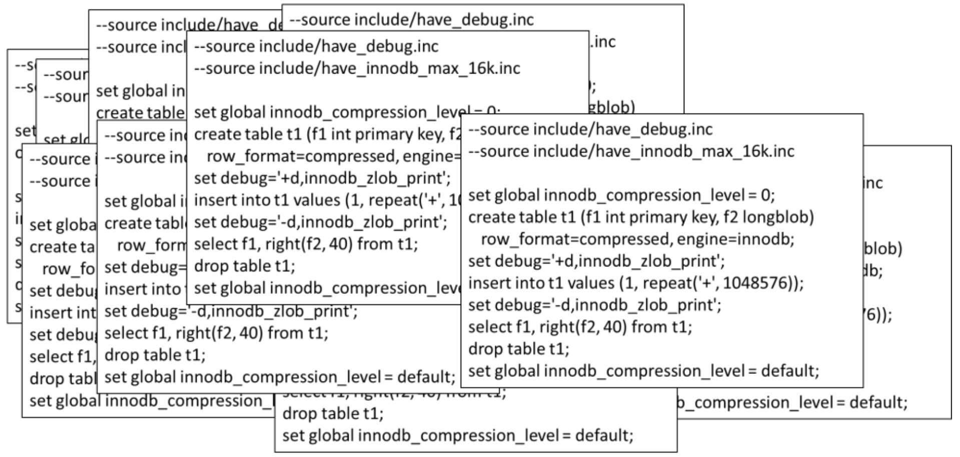
[HTML] **Testing** multiple linear regression systems with **metamorphic testing** — [HTML] sciencedirect.com

Creating an effective technique is challenging, check Google Scholar if you can find existing ones!

Speichern Zitieren Zitiert von: 1 Ähnliche Artikel Alle 4 Versionen

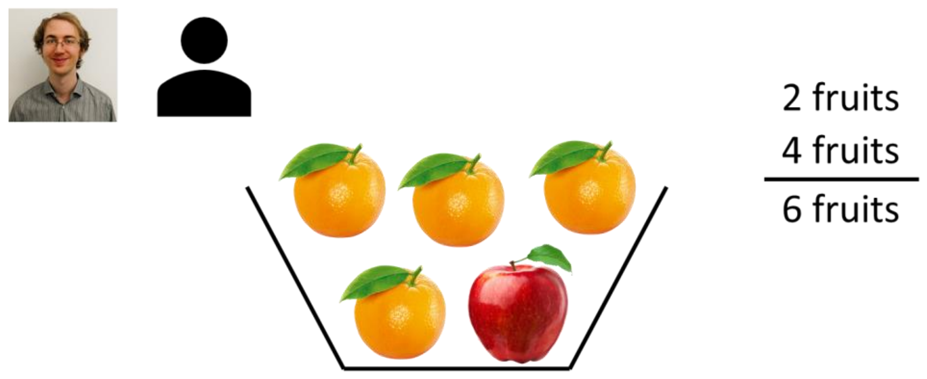
[HTML] **Metamorphic testing** of OpenStreetMap — [HTML] sciencedirect.com
JM Almendros-Jiménez, A Becerra-Terón... - Information and ..., 2021 - Elsevier
... Specifically, we will formally define all the elements taking part in OSM and the main notions in **metamorphic testing**. In Section 5 we define our **metamorphic** relations. In particular, we ...
Speichern Zitieren Zitiert von: 3 Ähnliche Artikel Alle 3 Versionen

Summary & Takeaway



```
--source include/have_debug.inc
--source include/have_innodb_max_16k.inc

set global innodb_compression_level = 0;
create table t1 (f1 int primary key, f2 longblob)
  row_format=compressed, engine=innodb;
set debug='+d,innodb_zlob_print';
insert into t1 values (1, repeat('+', 1048576));
set debug='-d,innodb_zlob_print';
select f1, right(f2, 40) from t1;
drop table t1;
set global innodb_compression_level = default;
```



Manually writing test cases is **time intensive** and requires detailed **domain knowledge**

Coupling **random test case generation** with a **metamorphic test oracle** is effective in finding bugs

@RiggerManuel @sqlancer_dbms

ETH zürich