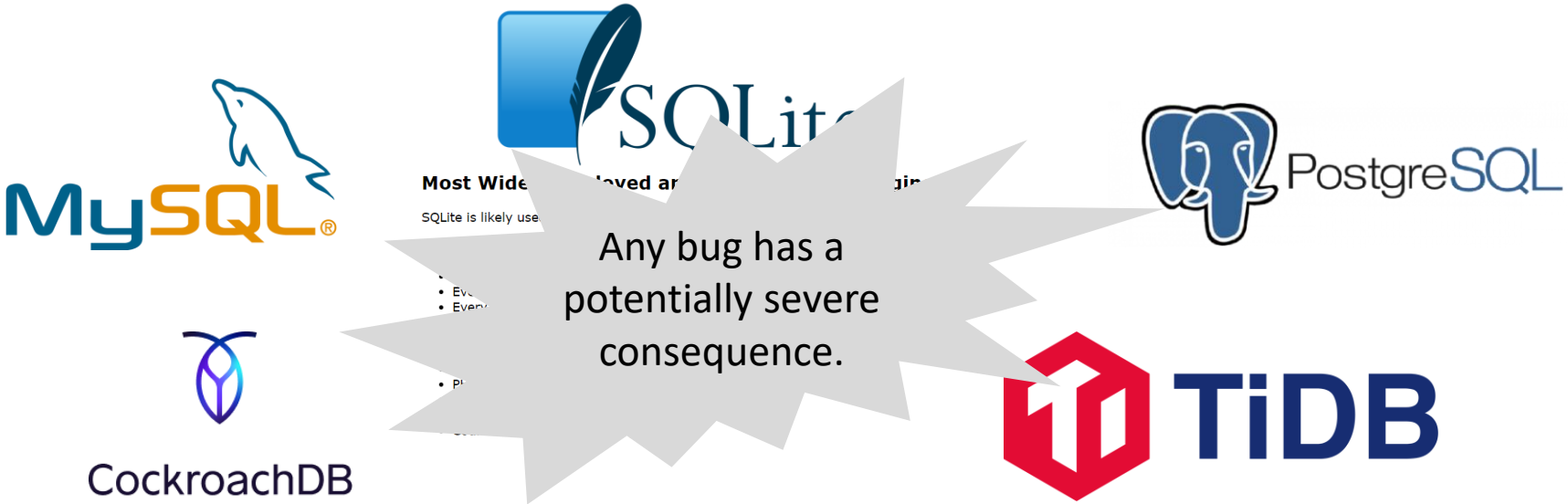
Testing Database Engines via Query Plan Guidance

Jinsheng Ba, Manuel Rigger
National University of Singapore

Testing Database Engines via Query Plan Guidance



Our method finds 53 unique previously-unknown bugs.

*Most Widely Deployed and Used Database Engine: https://www.sqlite.org/mostdeployed.html

Testing Database Engines via Query Plan Guidance

- Test oracles are used to detect bugs.
- For example: Non-optimizing Reference Engine Construction (NoREC)

```
SELECT * FROM t0 WHERE (t0.c0>0);
```

→ {|1|2|}

```
SELECT (t0.c0>0) FROM t0;
```

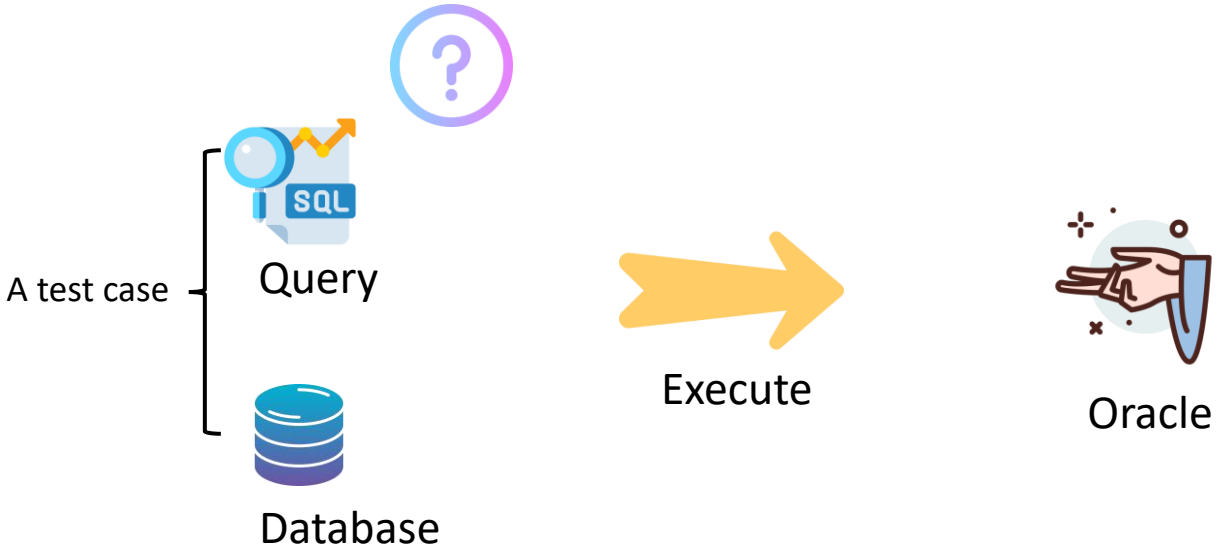
→ {}

Whether the number of TRUEs of the second query equals to the number of rows of the first query

NoREC

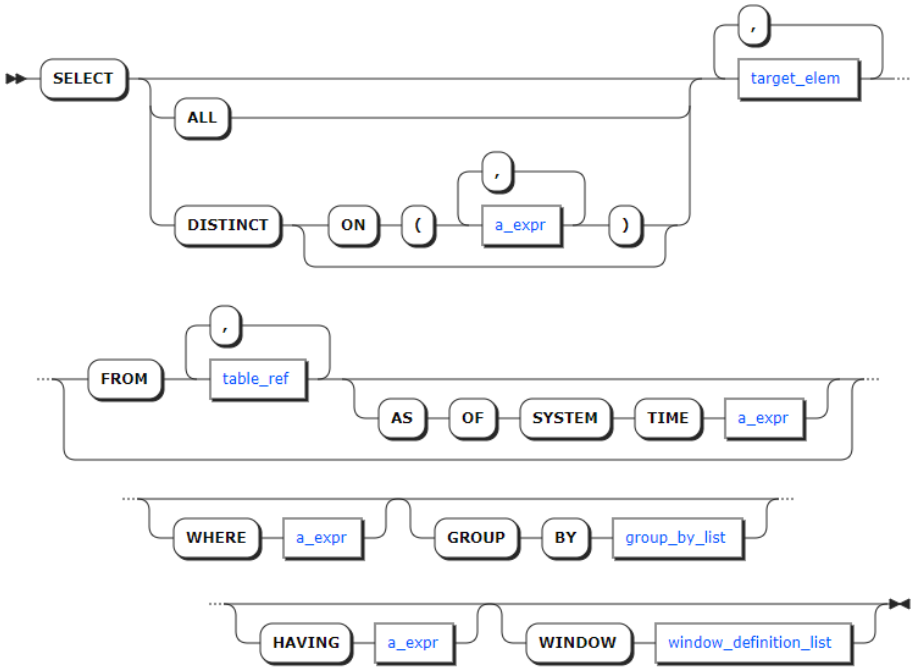
Problem: How To Generate Test Cases?

- Given the test oracles to detect bugs, how do we generate test cases?



Previous Test Case Generation Methods

- Generation-based methods.
  - Restricted to the grammar and hard to generate diverse test cases.
- Examples: SQLSmith[2], SQLancer[3]

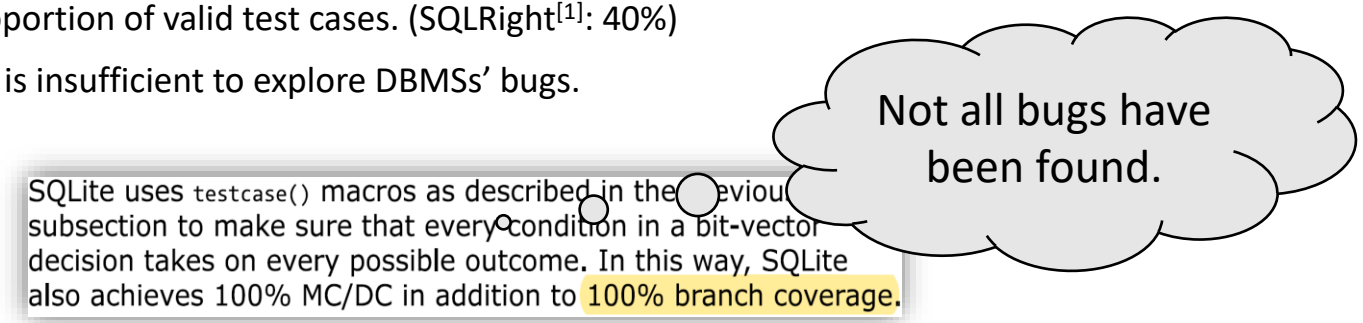


The SQL grammar[1] for CockroachDB.

[1] https://www.cockroachlabs.com/docs/stable/select-clause.html
[2] https://github.com/anse1/sqlsmith
[3] https://github.com/sqlancer/sqlancer

Previous Test Case Generation Methods

- Mutation-based methods (Coverage-guided Grey-box fuzzing).
  - Insufficient proportion of valid test cases. (SQLRight[1]: 40%)
  - Code coverage is insufficient to explore DBMSs' bugs.



SQLite Documents[2].

- Example: SQLRight

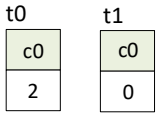
[1] Liang, Y., Liu, S., & Hu, H. (2022). Detecting Logical Bugs of {DBMS} with Coverage-based Guidance. In 31st USENIX Security Symposium (USENIX Security 22) (pp. 4309-4326).
[2] https://www.sqlite.org/testing.html#mcdc

Testing Database Engines via Query Plan Guidance

Idea: Guiding test case generation towards unseen query plans, aiming to cover more behaviours.

What is a Query Plan?

- A query plan is a tree of operations that describes how a SQL statement is executed by a specific DBMS.




```
EXPLAIN SELECT * FROM t0 LEFT
        JOIN t1 WHERE t1.c0==0;
```

Input

```
|--SCAN t0
`--SCAN t1 LEFT-JOIN
   `--Filter
```

Query Plan

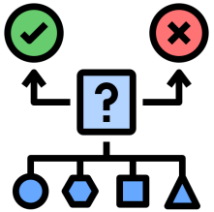
{2|0}

Output

What is a Query Plan?

- A query plan is a tree of operations that describes how a SQL statement is executed by a specific DBMS.
- DBMSs typically expose query plans to users for tuning the performance of queries.

  - How to scan tables? (full scan, partial scan with index…)
  - How to join tables? (hash join, merge join…)
  - Where to apply filter? (after join, after table scan…)



Query Plan Guidance

```
SELECT * FROM t0 LEFT JOIN t1
WHERE t1.c0==0;
```



```
|--SCAN t0
`--SCAN t1 LEFT-JOIN
    `--Filter
```


{2|0}

Query A

```
|--Filter
|--SCAN t
```

Query B

```
|--SCAN t0
`--USE TEMP B-TREE FOR ORDER BY
```

Query C

```
|--Filter
|--SCAN t0
`--SCAN t1 LEFT-JOIN
```

…

…

Query Plan Study

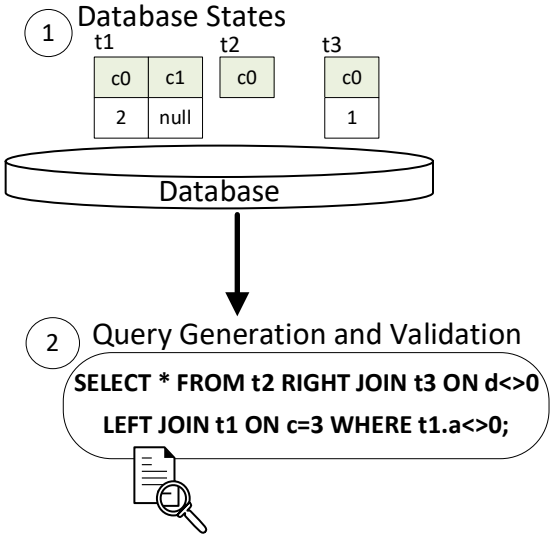
- Query plans of the queries in previously-found bugs are:
  - 1) Diverse.
  - 2) Compact and simple.

DBMS	Bugs	Query Plans		
		Sum	Unique	Length
CockroachDB	68	37	32	3.43
DuckDB	75	59	18	2.00
H2	19	10	7	3.70
MariaDB	7	5	5	1.00
MySQL	40	35	22	1.03
PostgreSQL	31	9	3	2.33
SQLite	193	118	62	2.14
TiDB	62	43	32	5.07
			Unique/Sum=57.28%	**Avg: 2.59**

***https://github.com/sqlancer/bugs: 495 bugs for DBMSs.**

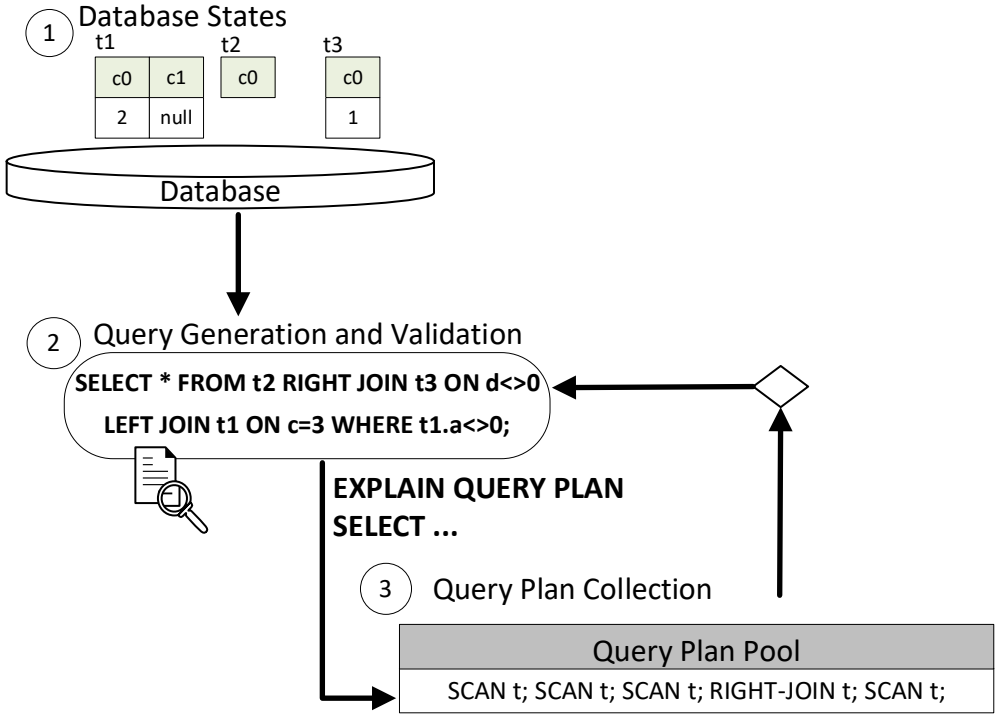
Step 1 & 2: Query Generation and Validation

We reused the existing database and query generation approaches.



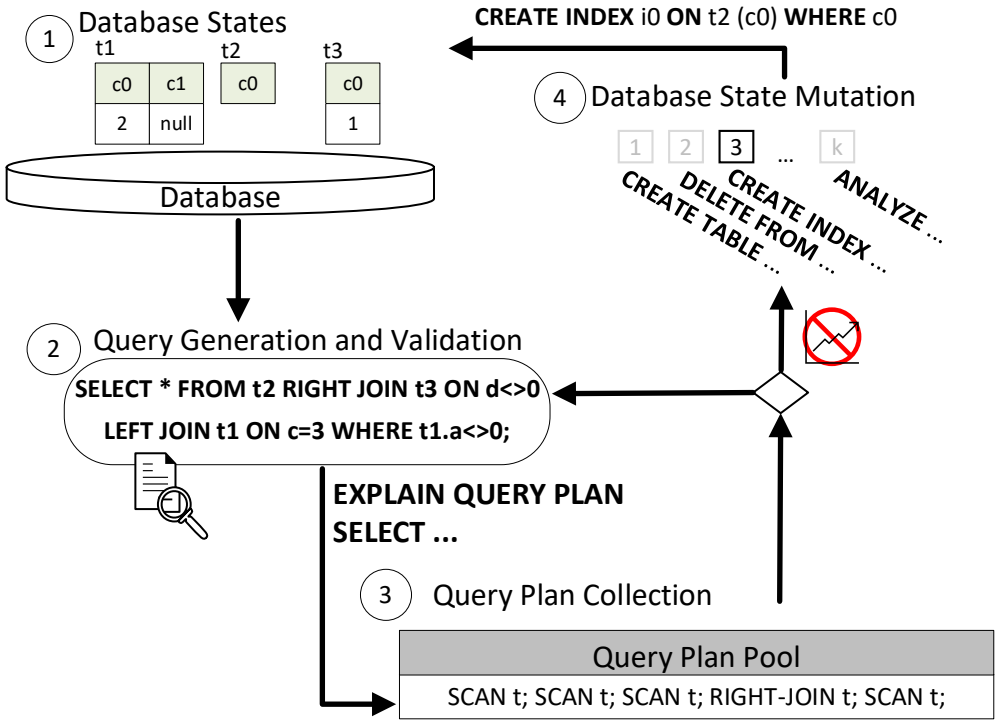
Step 3: Query Plan Collection

Insert unique query plans to the query plan pool



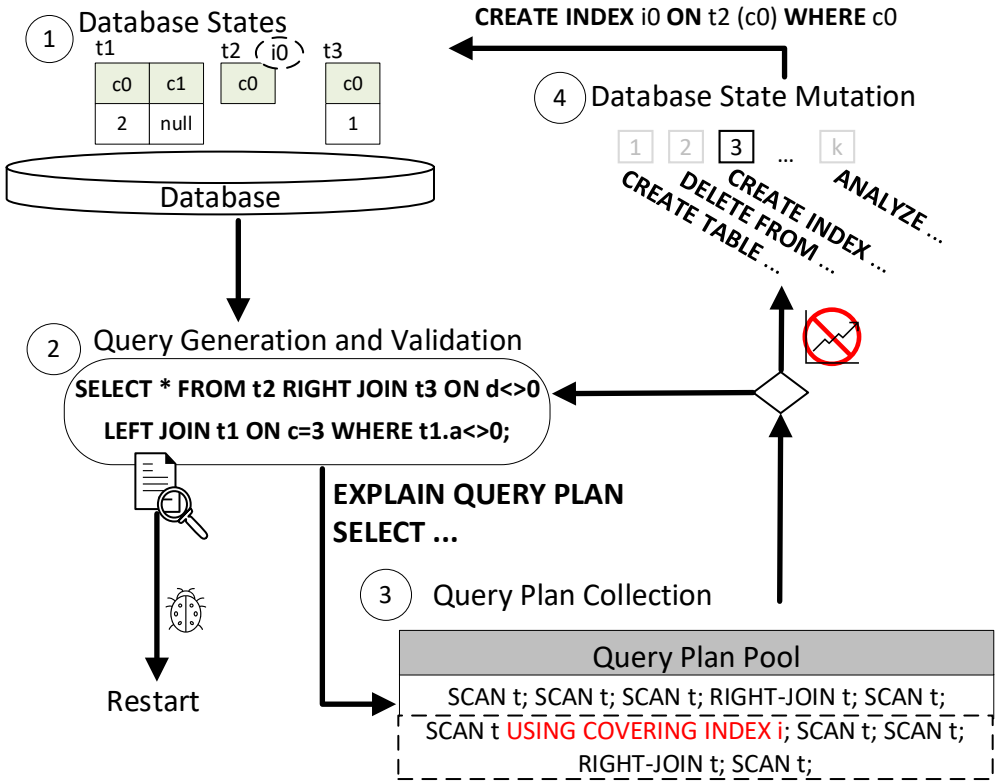
Step 4: Database State Mutation

Mutate the database state if no query plan has been observed for a certain number of iterations



Approach Overview

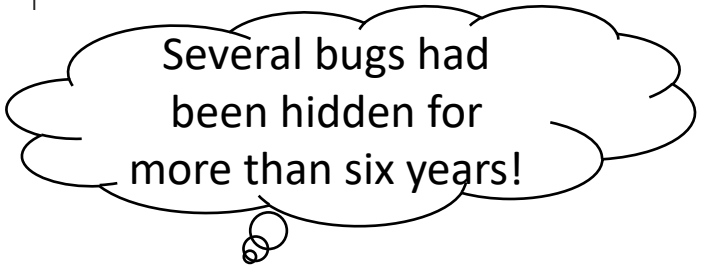
New query plans are able to be observed, and new bugs may be found



Evaluation: New Bugs

(2) By Richard Hipp (drh) on 2022-07-15 12:59:59 in reply to 1 [link] [source]

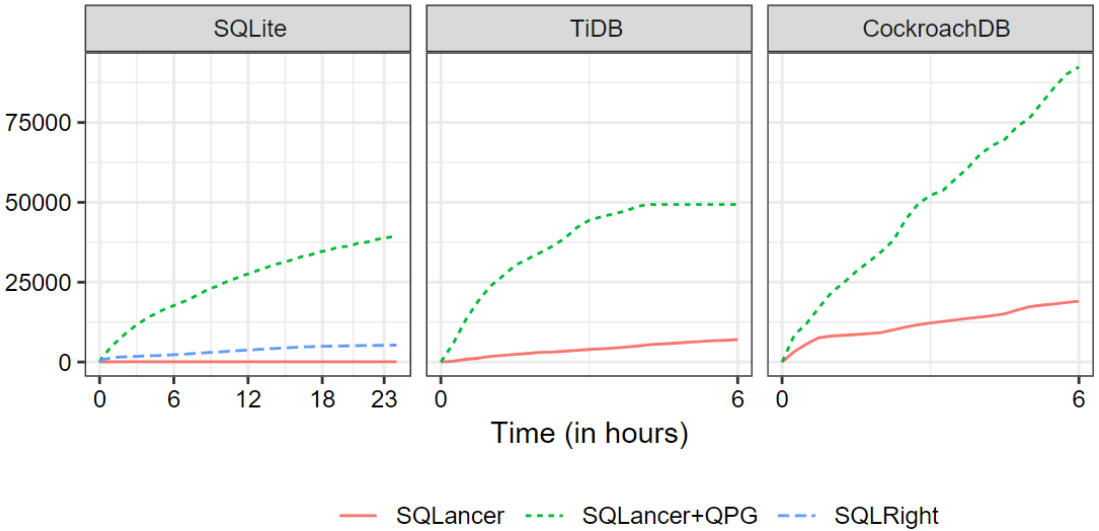
This bug goes back almost 8 years to check-in ddb5f0558c445699 on 2016-09-07, ve



DBMS	Crash	Internal Error	Logic	All
SQLite	0	5	23	28
TiDB	2	4	3	9
CockroachDB	3	11	2	16
Sum:	**5**	**20**	**28**	**53**

With the help of QPG, we found 53 unique, previously unknown bugs.

Evaluation: Covering unique query plans



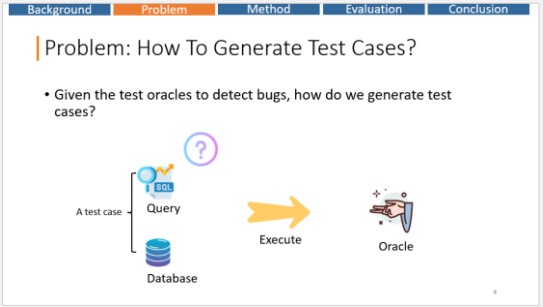
The average number of unique query plans across 10 runs in 24 hours.

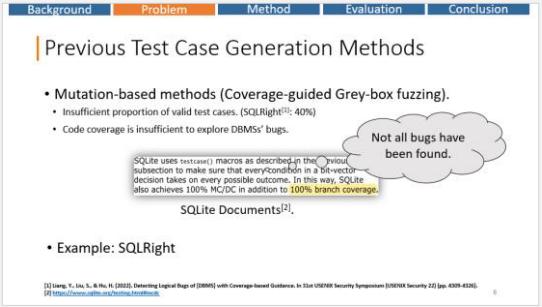
QPG exercises
4.85–408.48× more unique query plans than a naive random generation method (SQLancer),
7.46× more than a code-coverage guidance method (SQLRight).

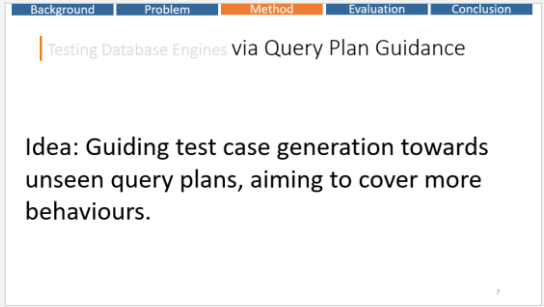
Conclusion

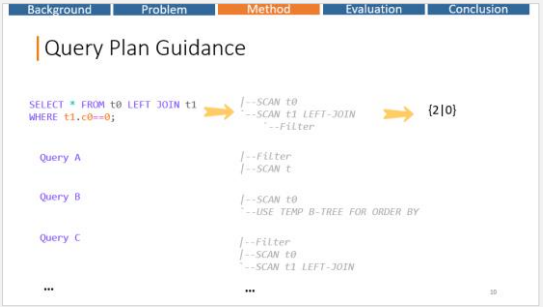
Github

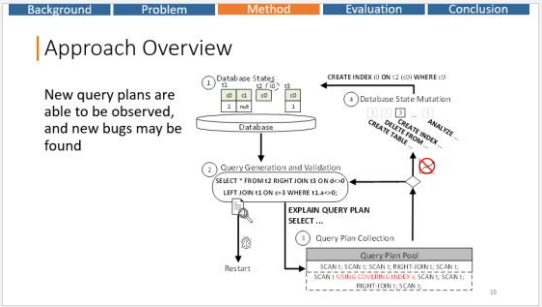
Author

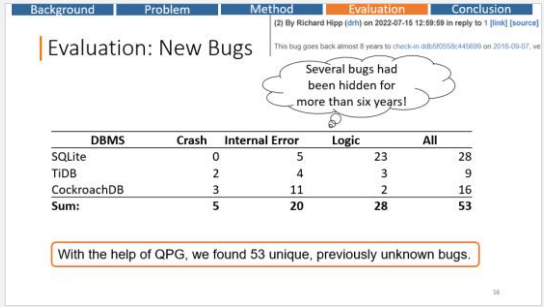












Query plans can efficiently guide test-case generation in a black-box manner.