Squirrel: Testing DBMS with Language Validity and Coverage Feedback

Hong Hu

Database Management Systems (DBMS)

DBMS *is* Popular

- Over one trillion (1×10^{12}) SQLite DBs in use

- **Apple** uses SQLite in many native MacOS/iOS applications
- **Dropbox client** uses SQLite in archiving and sync service
- **Google** uses SQLite in Android OS and Chrome browser
- **Microsoft** uses SQLite as a core component of Windows 10

DBMS Has Severe Bugs

- CVE-2018-20346
- CVE-2018-20505
- CVE-2018-20506
- CVE-2019-5018
- CVE-2019-8600
- CVE-2019-8598
- CVE-2019-8602
- CVE-2019-8577
- CVE-2019-13734
- CVE-2019-13750
- CVE-2019-13751
- CVE-2019-13752
- CVE-2019-13753
- ...

Critical MySQL Vulnerabilities Can Lead to

Patch RCE vulnerability CVE-2020-0618 on your Microsoft SQL-Server!

Posted on 2020-

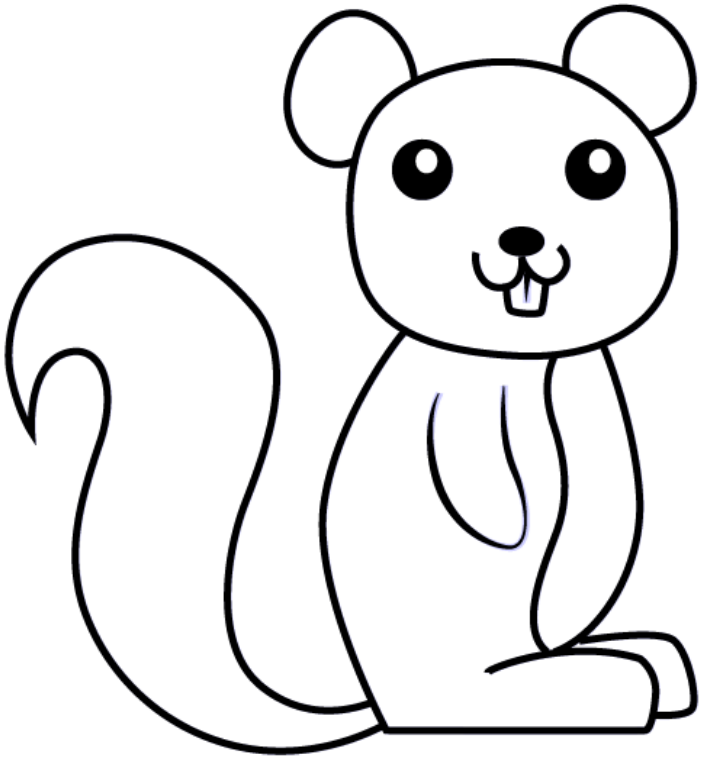
MySQL Remote Root Code Execution / Privilege Escalation (0day Exploit) CVE-2016-6662

legalhackers.com/adviso...

Finding DBMS Bugs *are* Challenging

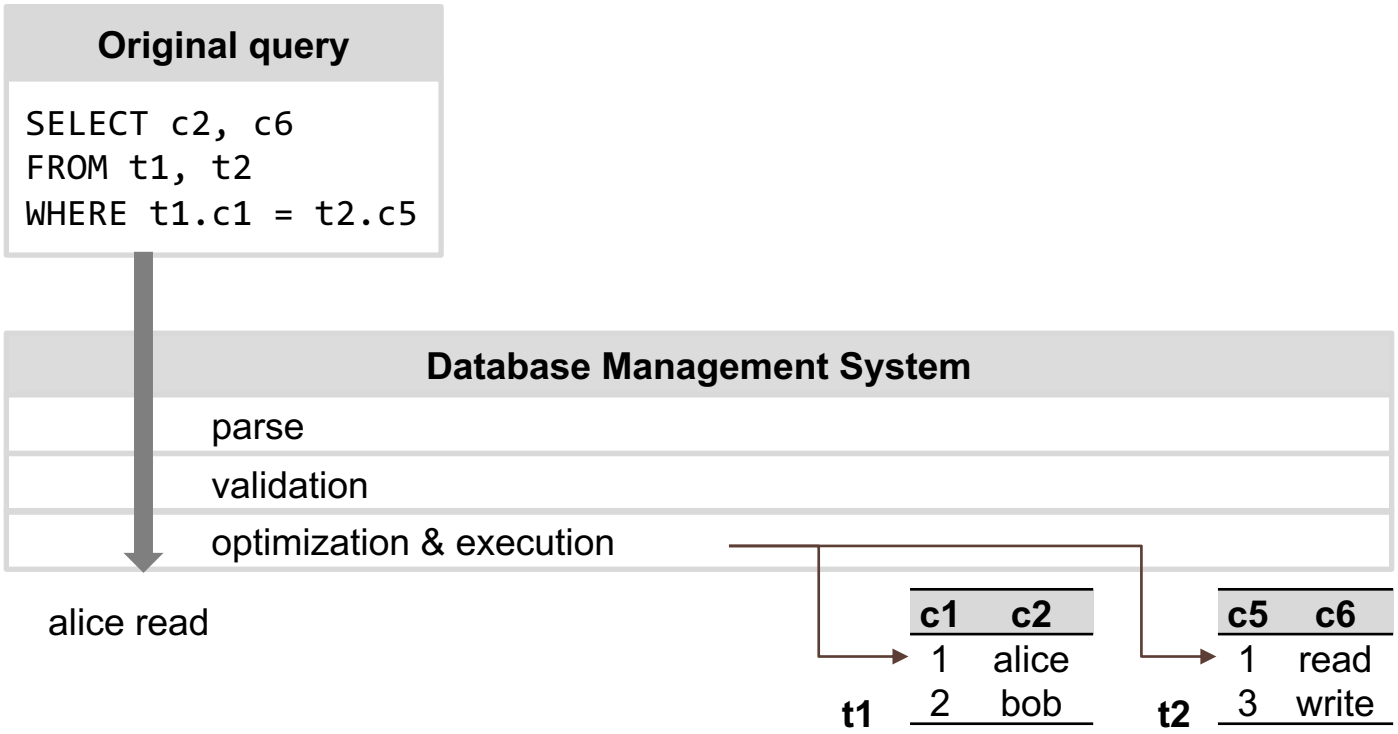
- Generate valid test cases
  - invalid cases are rejected in early stage, and cannot reach deep logic
  - random mutation does not work
- Guide generation for bug finding
  - enumeration is less effective
  - grammar-based generator does not work

Our Contribution

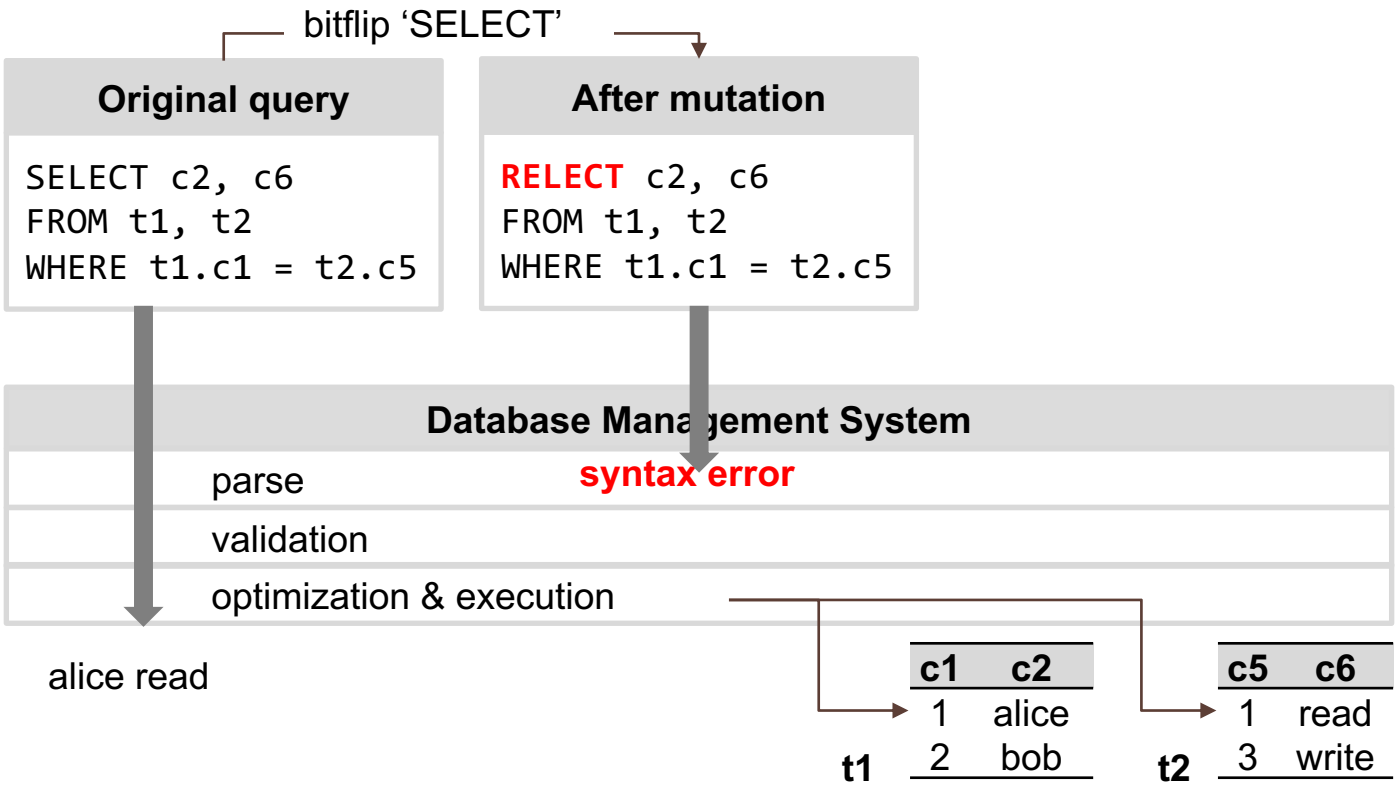


- A general platform to test DBMS systems
  - language validity & coverage feedback
  - support SQLite, MySQL, PostgreSQL, MariaDB
    - easily extensible to other DBMSs
- Bugs in real-world DBMS
  - **51** bugs in SQLite, **7** in MySQL and **5** in MariaDB
  - **52** of the bugs are fixed with **12** CVEs assigned
- https://github.com/s3team/Squirrel

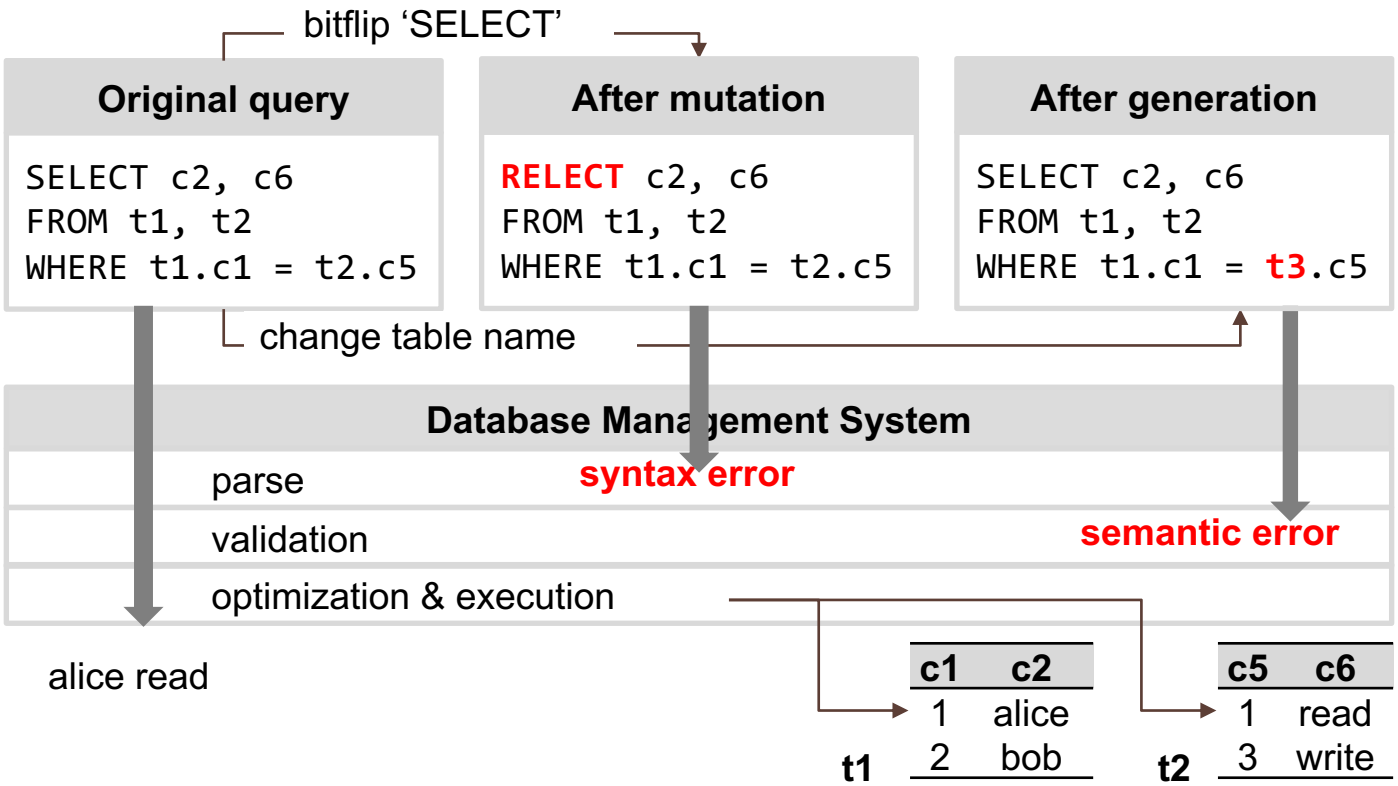
Work Flow of DBMSs



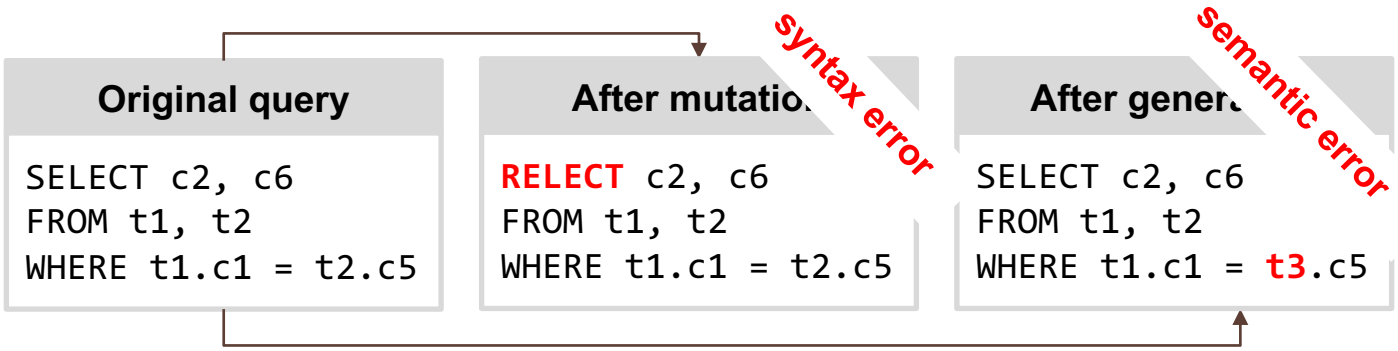
Challenges: Syntax-validity



Challenges: Semantic-validity

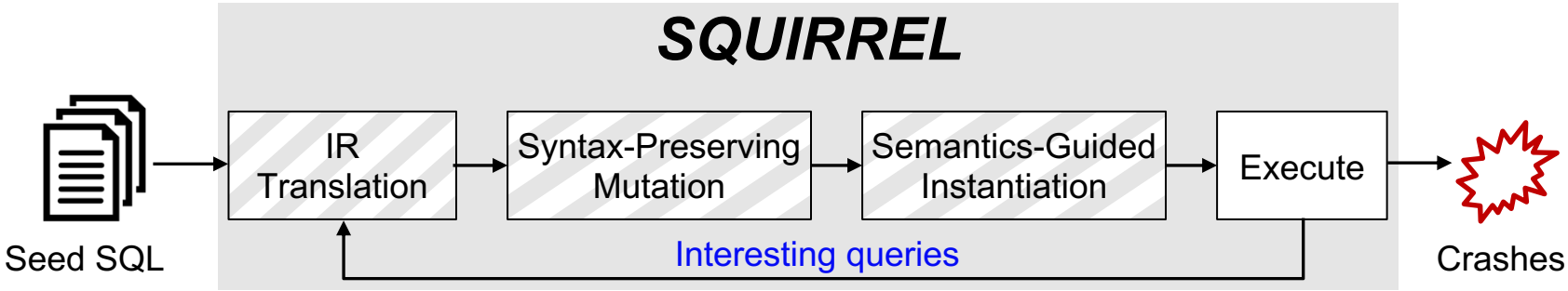


Challenges: Guidance



on the right way?

Our Approach: Squirrel

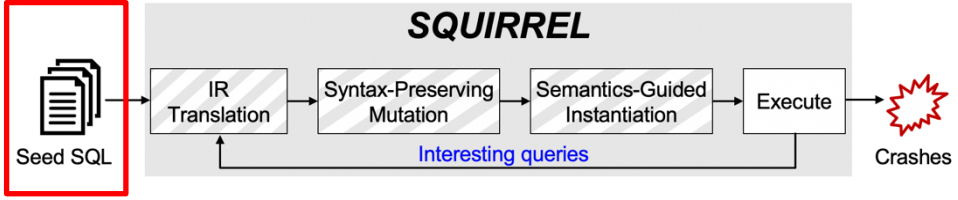


We take advantages of mutation-based and generation-based techniques

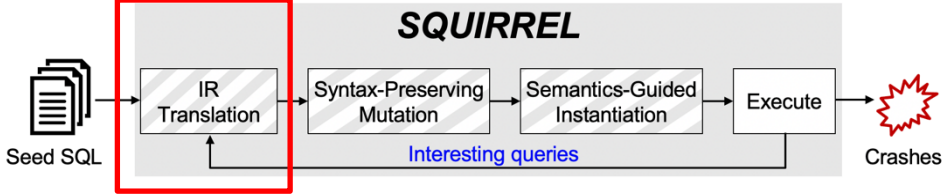
- generate **syntax-correct** queries
- fix **semantic errors**
- adopt **feedback mechanism** to prioritize interesting queries

An Example

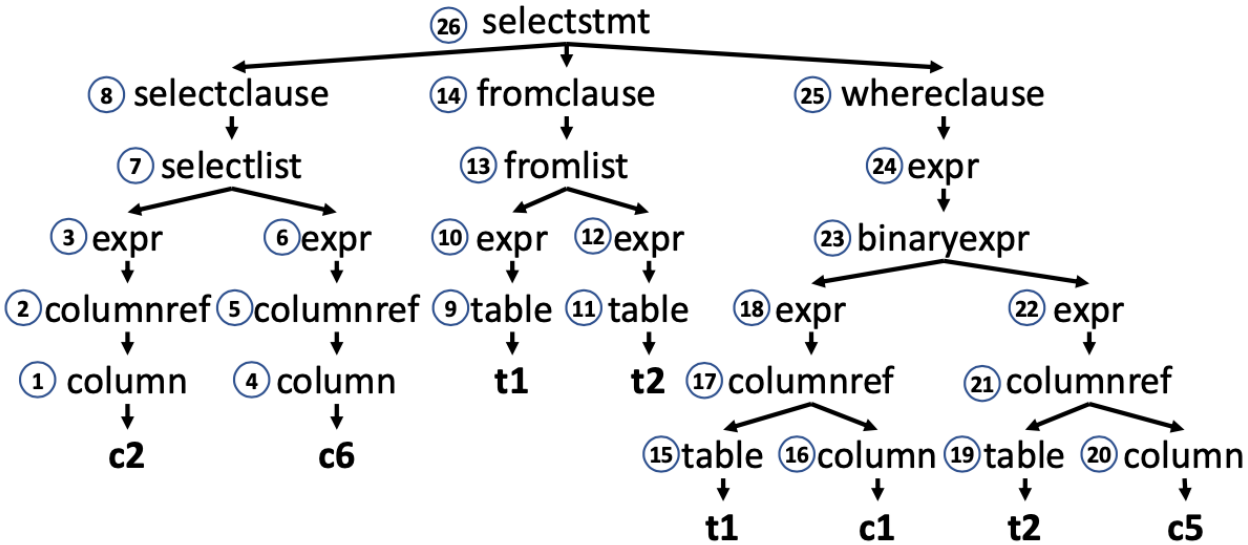
```
SELECT c2, c6
FROM t1, t2
WHERE t1.c1 = t2.c5
```



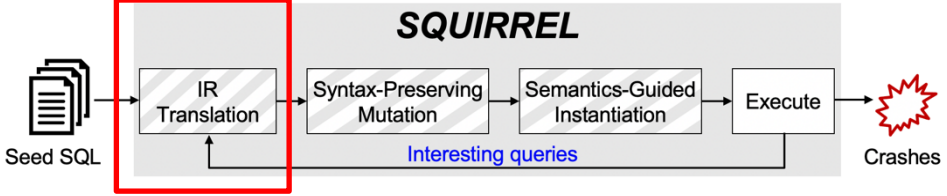
IR Translation




```
SELECT c2, c6
FROM t1, t2
WHERE t1.c1 = t2.c5
```



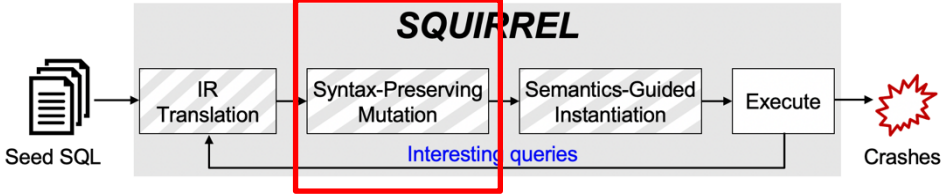
IR Translation




```
SELECT c2, c6
FROM t1, t2
WHERE t1.c1 = t2.c5
```

```
1  // l: left child, r: right child, d: data, t: data type
2  V1 = (Column,     l=0,  r=0,  op=0, d="c2", t=ColumnName);
3  V2 = (ColumnRef,  l=V1, r=0,  op=0, d=0);
4  V3 = (Expr,       l=V2, r=0,  op=0, d=0);
5  V4 = (Column,     l=0,  r=0,  op=0, d="c6", t=ColumnName);
6  V5 = (ColumnRef,  l=V4, r=0,  op=0, d=0);
7  V6 = (Expr,       l=V5, r=0,  op=0, d=0);
8  V7 = (SelectList, l=V3, r=V6, op=0, d=0);
9  // the optional left child can be DINSTRICT
10 V8 = (SelectClause, l=0, r=V6, op.prefix="SELECT", d=0);
11 ...
12 //Unknown type for intermediate IRs
13 Va = (Unknown,    l=V8, r=V14, op=0, d=0);
14 Vb = (Unknown,    l=Va, r=V25, op=0, d=0);
15 // the optional right child can be an ORDER clause
16 V26 = (SelectStmt, l=Vb, r=0,  op=0, d=0);
```

Mutation

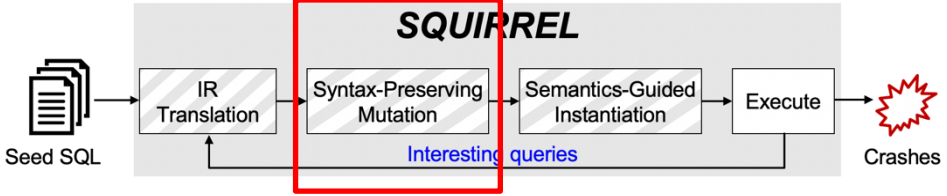



```
SELECT c2, c6
FROM t1, t2
WHERE t1.c1 = t2.c5
```

strip concrete operand

```
SELECT x, x
FROM x, x
WHERE x.x = x.x
```

Mutation




```
SELECT x,x FROM x,x WHERE x.x = x.x;
```

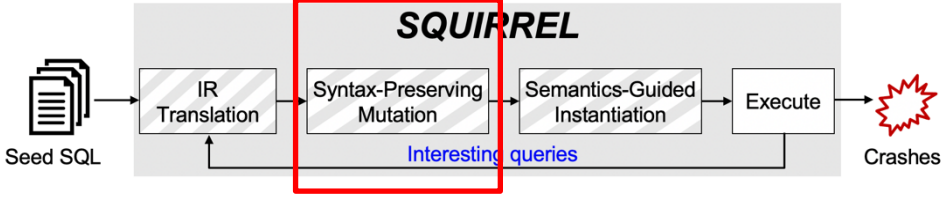
```
V8 = (SelectClause, l=0, r=V6, op.prefix="SELECT”…);
Va = (Unknown, l=V8, r=V14, op=0, d=0);
Vb = (Unknown, l=Va, r=V25, op=0, d=0);
V26 = (SelectStatement, l=Vb, r=0, op=0, d=0);
```

Insertion

```
SELECT x,x FROM x,x WHERE x.x = x.x ORDER BY x;
```

```
Vc = (OrderbyClause, …);
V26 = (SelectStatement, l=Vb, r=Vc, op=0, d=0);
```

Mutation




```
SELECT x,x FROM x,x WHERE x.x = x.x;
V8 = (SelectClause, l=0, r=V6, op.prefix="SELECT”…);
Va = (Unknown, l=V8, r=V14, op=0, d=0);
Vb = (Unknown, l=Va, r=V25, op=0, d=0);
V26 = (SelectStatement, l=Vb, r=0, op=0, d=0);
```

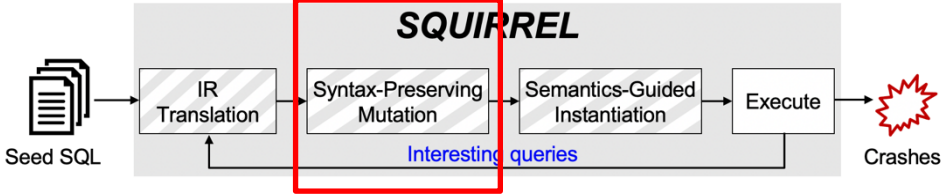
Insertion

```
SELECT x,x FROM x,x WHERE x.x = x.x ORDER BY x;
Vc = (OrderbyClause, …);
V26 = (SelectStatement, l=Vb, r=Vc, op=0, d=0);
```

Replacement

```
SELECT count(x,x) FROM x,x WHERE x.x = x.x;
Vc = (CountClause, …);
V8 = (SelectClause, l=0, r=Vc, op.prefix="SELECT”…);
```

Mutation




```
SELECT x,x FROM x,x WHERE x.x = x.x;
V8 = (SelectClause, l=0, r=V6, op.prefix="SELECT”…);
Va = (Unknown, l=V8, r=V14, op=0, d=0);
Vb = (Unknown, l=Va, r=V25, op=0, d=0);
V26 = (SelectStatement, l=Vb, r=0, op=0, d=0);
```

Insertion

```
SELECT x,x FROM x,x WHERE x.x = x.x ORDER BY x;
Vc = (OrderbyClause, …);
V26 = (SelectStatement, l=Vb, r=Vc, op=0, d=0);
```

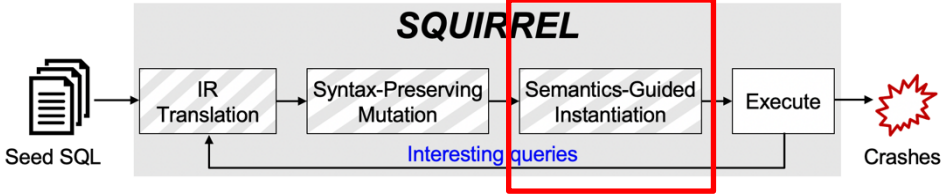
Replacement

```
SELECT count(x,x) FROM x,x WHERE x.x = x.x;
Vc = (CountClause, …);
V8 = (SelectClause, l=0, r=Vc, op.prefix="SELECT”…);
```

Deletion

```
SELECT x,x FROM x,x;
Vb = (Unknown, l=Va, r=0, op=0, d=0);
```

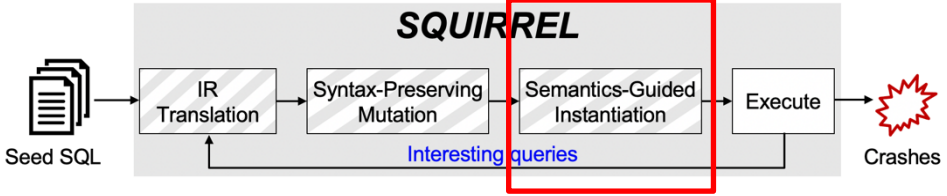
Instantiation




```
CREATE TABLE x1 (x2 INT, x3 INT)
CREATE TABLE x4 (x5 INT, x6 INT)
CREATE TABLE x7 (x8 INT, x9 INT)
SELECT x10, x11 FROM x12, x13 WHERE x14.x15 = x16.x17
```

Build Data Dependency Graph (DDG)

Instantiation

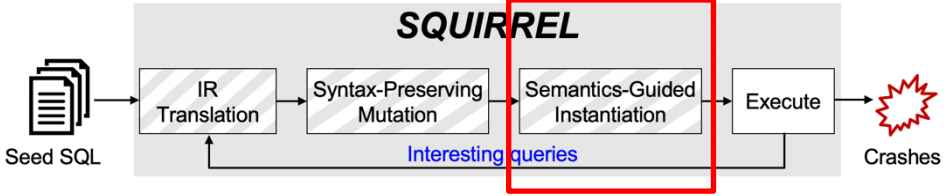



```
CREATE TABLE x1 (x2 INT, x3 INT)
CREATE TABLE x4 (x5 INT, x6 INT)
CREATE TABLE x7 (x8 INT, x9 INT)
SELECT x10, x11 FROM x12, x13 WHERE x14.x15 = x16.x17
```

Data	Type
x1, x4, x7	CreateTable
x2, x3	CreateColumn
x5, x6	CreateColumn
x8, x9	CreateColumn
x10, x11	UseFromColumn
x12, x13	UseAnyTable
x14, x16	UseFromTable
x15	UseTableColumn
x17	UseTableColumn

Build Data Dependency Graph (DDG)

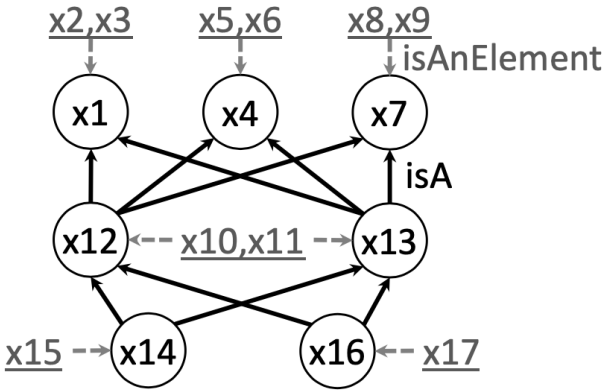
Instantiation




```
CREATE TABLE x1 (x2 INT, x3 INT)
CREATE TABLE x4 (x5 INT, x6 INT)
CREATE TABLE x7 (x8 INT, x9 INT)
SELECT x10, x11 FROM x12, x13 WHERE x14.x15 = x16.x17
```

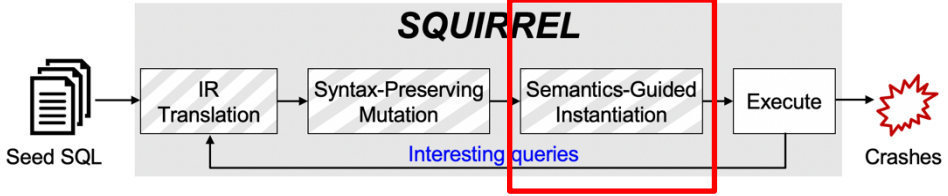
Data	Type
x1, x4, x7	CreateTable
x2, x3	CreateColumn
x5, x6	CreateColumn
x8, x9	CreateColumn
x10, x11	UseFromColumn
x12, x13	UseAnyTable
x14, x16	UseFromTable
x15	UseTableColumn
x17	UseTableColumn

Relation

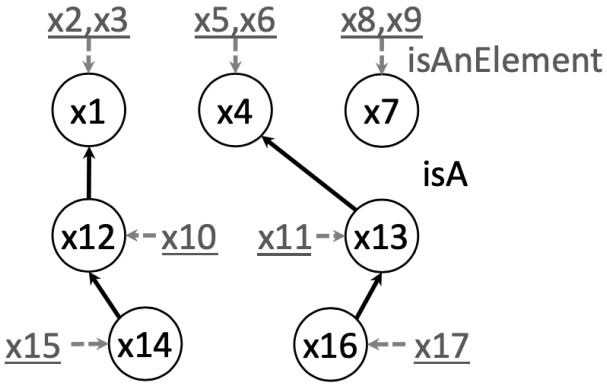


Build Data Dependency Graph (DDG)

Instantiation

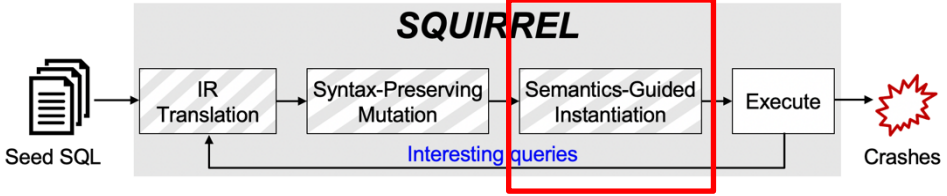


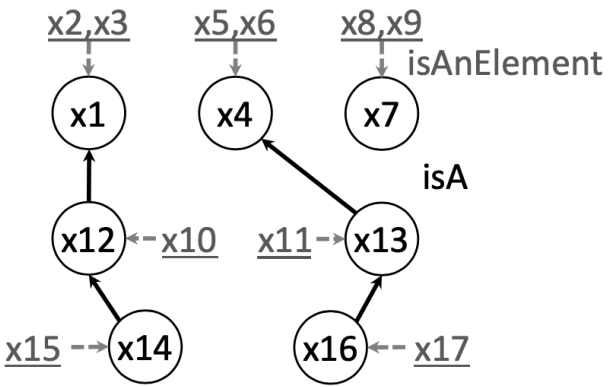
One possible relation



Pick concrete data to create query

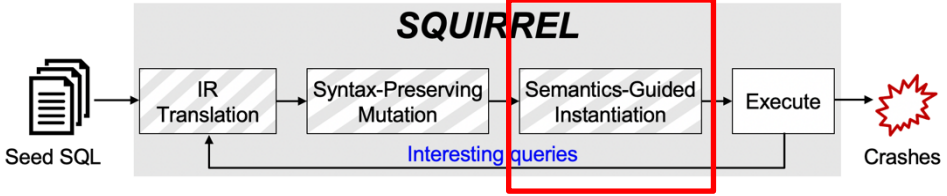
Instantiation

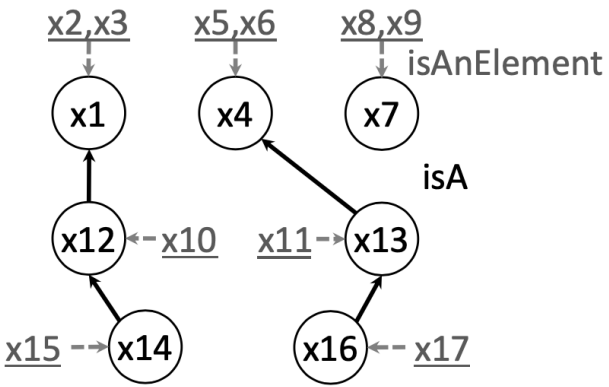


One possible relation	Data	Value
	x1, x12, x14	v1
	x2, x15	v2
	x3, x10	v3
	x4, x13, x16	v4
	x5, x17	v5
	x6, x11	v6
	x7	v7
	x8	v8
	x9	v9

Pick concrete data to create query

Instantiation



One possible relation	Data	Value
	x1, x12, x14	v1
	x2, x15	v2
	x3, x10	v3
	x4, x13, x16	v4
	x5, x17	v5
	x6, x11	v6
	x7	v7
	x8	v8
	x9	v9


```
CREATE TABLE v1 (v2 INT, v3 INT)
CREATE TABLE v4 (v5 INT, v6 INT)
CREATE TABLE v7 (v8 INT, v9 INT)
SELECT v3, v6 FROM v1, v4 WHERE v1.v2 = v4.v5
```

Pick concrete data to create query

Evaluation: New Bugs

Ran Squirrel for 40 days on one 16-core server.

Bugs found in SQLite, MySQL and MariaDB

- **63** unique bugs found & confirmed
- **52** bugs fixed
- **12** CVEs assigned

New Bugs

UAF: use-after-free. **BOF**:buffer overflow of Global (**G**), Heap (**H**), and Stack (**S**).
BUF: buffer underflow. **AF**: assertion failure. **OOM**: out of memory. **UB**: undefined behavior.

ID	Type	Function	Status	Severity†	Reference
SQLite v3.30.1, 300K LoC					
1	BOF	`PRAGMA integrity_check`	Fixed	Critical	CVE-2019-19646
2	NP	lookupName	Fixed	Critical	CVE-2019-19317
3	UAF	`WITH`	Fixed	High	CVE-2019-20218
4	BOF	exprListAppendList	Fixed	High	CVE-2019-19880
5	BOF	`ZipFile` extension	Fixed	High	CVE-2019-19959
6	NP	zipfileUpdate	Fixed	High	CVE-2019-19925
7	NP	`parser`	Fixed	High	CVE-2019-19926
8	NP	`LEFT JOIN` optimization	Fixed	High	CVE-2019-19923
9	SBOF	`ALTER TABLE`	Fixed	Medium	CVE-2019-19645
10	NP	`JOIN INDEX`	Fixed	Medium	CVE-2019-19242
11	NP	`parser`	Fixed	Medium	CVE-2019-19924
12	BOF	propagateConstantExprRewrite	Fixed	Medium	CVE-2020-6405
13	UB	fopen/fopen64	Fixed	-	0c4f820
14	GBOF	sqlite3VdbeMemPrettyPrint	Fixed	-	5ca0632
15	AF	sqlite3GenerateConstraintChecks	Fixed	-	ad5f157
16	AF	`IN` expression optimization	Fixed	-	b97f353
17	AF	whereLoopAddOr	Fixed	-	9a1f2e4
18	AF	`WHERE` with `OR` opt.	Fixed	-	a4b2df5
19	AF	wherePathSatisfiesOrderBy	Fixed	-	77c9b3c
20	AF	Bytecode `OP_DeferredSeek`	Fixed	-	be3da24
21	AF	`WHERE`	Fixed	-	4adb1d0
22	AF	`WHERE` flag setting	Fixed	-	118efd1
23	AF	Bytecode `OP_ResultRow` release	Fixed	-	02ff747
24	AF	sqlite3SelectReset	Fixed	-	aa328b6
25	AF	Bytecode `OP_SCopy`	Fixed	-	629b88c
26	AF	`scalar subquery`	Fixed	-	629b88c
27	AF	Bytecode `OP_ResultRow`	Fixed	-	02ff747
28	AF	`SELECT`	Fixed	-	fbb6e9f
29	AF	`WHERE`	Fixed	-	f1bb31e
30	AF	`PRAGMA encoding`	Fixed	-	b5f0e40
SQLite v3.31 (under development), 304K LoC					
31	GBOF	`ZipFile` extension	Fixed	-	8d7f44c
32	HBOF	`ZipFile` extension	Fixed	-	a194d31
33	HBUF	`ZipFile` extension	Fixed	-	8d7f44c
34	UAF	sqlite3GenerateConstraintChecks	Fixed	-	6d67aff
35	NP	`VTable`	Fixed	-	c7a5ff4
36	NP	`ORDER BY` Windows Function	Fixed	-	73bacb7
37	NP	`SF_Aggregate` flag setting	Fixed	-	9e10f9a
38	NP	`USING`	Fixed	-	0824d5b
39	NP	`ZipFile` extension	Fixed	-	0d21eae
40	NP	`LEFT JOIN` uses values from `IN`	Fixed	-	74ebaad
41	AF	`WHERE`	Fixed	-	b592d47
42	AF	`NEVER` marco can be true	Fixed	-	78b5220
43	AF	impliesNotNullRow	Fixed	-	aef8167
44	AF	`Code Generator` for inline function	Fixed	-	25c4296
45	AF	scalar `SELECT` w/ `WINDOW`	Fixed	-	4ea562e
46	AF	`Code Generator` for sub query	Fixed	-	fc705da
47	AF	AND optimization	Fixed	-	2b6e670
48	AF	Bytecode `OP_Move`	Fixed	-	4cbd847
49	AF	Bytecode `OP_Copy-coalesce` opt.	Fixed	-	9099688
50	AF	sqlite3ExprCodeIN	Fixed	-	f6ea97e
51	AF	whereTermPrint	Fixed	-	6411d65
MySQL v8.0, 4250K LoC					
52	OOM	`WITH` optimization	Verified	Critical	ID98190
53	NP	`JOIN` optimization	Fixed	Serious	ID98119
54	NP	`JOIN` optimization	Verified	?	ID99438
55	NP	`UPDATE` optimization	Verified	?	ID99424
56	AF	`SELECT`	Verified	?	ID99420
57	AF	`INDEX`	Verified	?	ID99421
58	AF	`CREATE TABLE`	Verified	?	ID99454
MariaDB v10.5.3, 3641K LoC					
59	BOF	`UPDATE`	Verified	?	MDEV22464
60	BOF	`UPDATE`	Verified	?	MDEV22476
61	AF	`JOIN`	Verified	?	MDEV22461
62	AF	`SELECT`	Verified	?	MDEV22462
63	AF	`Array OOB`	Verified	?	MDEV22463

Evaluation: Compared With Existing Tools

Compared with SQLSmith, Angora, GRIMOIRE, QSYM, AFL.

- up to **20.9x** higher syntax correctness.
- up to **243.9x** higher semantic correctness.
- up to **10.9x** more edges.

Evaluation: Contributions of Different Aspects

[Squirrel w/o semantic
Squirrel w/o feedback
Squirrel w/o semantic_syntax]

- Syntax correctness helps achieve up to **1.5x** more new edges
- Semantic correctness helps achieve up to **1.7x** more new edges
- Feedback helps achieve **2.0x** more new edges

Summary of Squirrel

- A general DBMS testing framework
  - generate high-quality SQL test cases
- Discovered bugs in popular DBMSs
  - 63 bugs confirmed
  - 12 CVEs assigned
- https://github.com/s3team/Squirrel