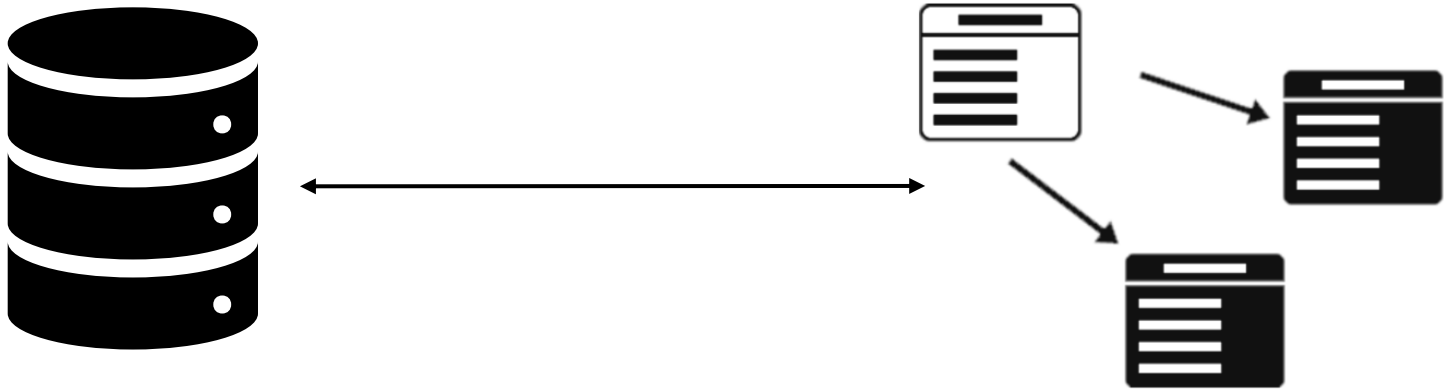
STICCER: Fast and Effective Database Test Suite Reduction Through Merging of Similar Test Cases

by Abdullah Alsharif(a.Alsharif@seu.edu.sa), Gregory M. Kapfhammer, and Phil McMinn

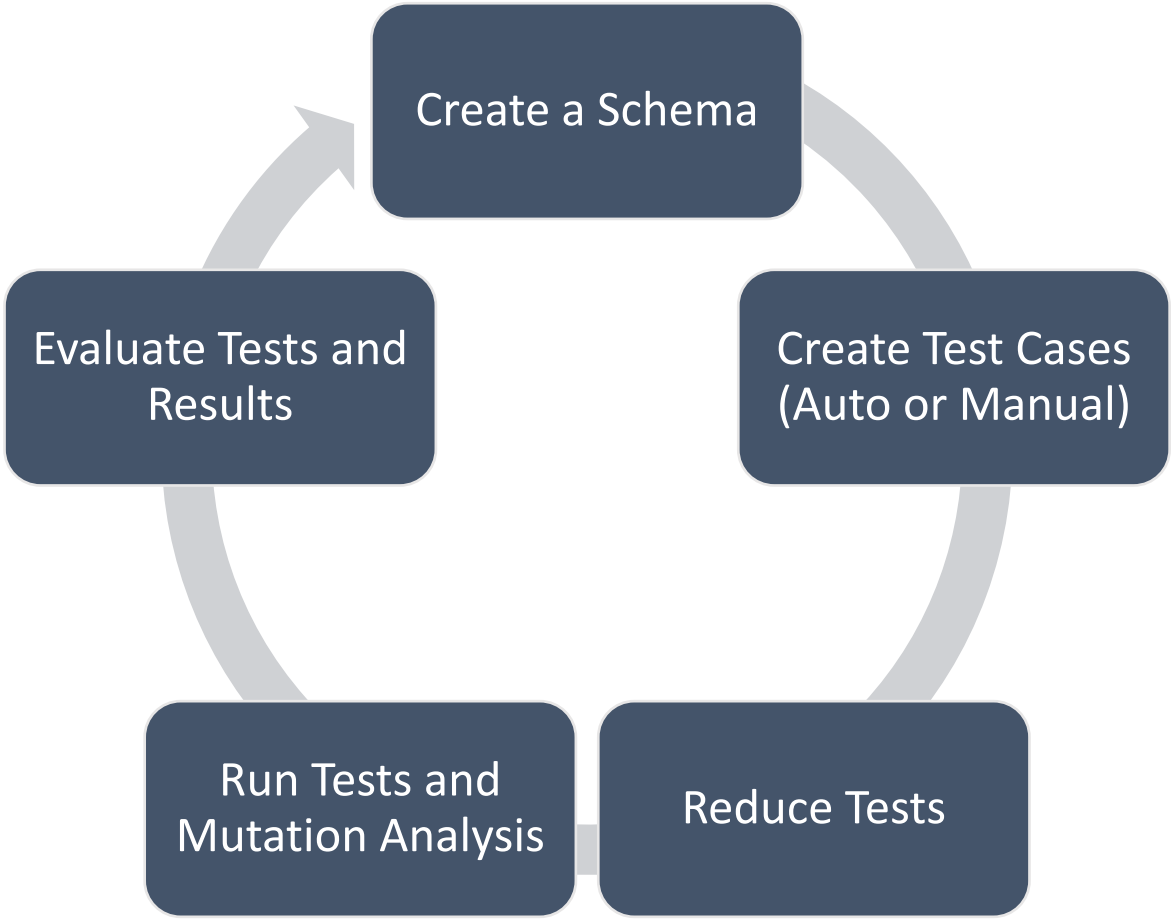
RELATIONAL DATABASES ARE EVERYWHERE AND THE BACKBONE OF MOST SOFTWARE SYSTEMS

Testing Relational Database Schemas



"A good [relational] database **schema** should have many **constraints**. [Therefore], you should **test** them“
Szymon Guz, 2011

The Process


Create a Schema
Create Test Cases (Auto or Manual)
Reduce Tests
Run Tests and Mutation Analysis
Evaluate Tests and Results

Too Many Test Cases

MANY CHANGES CAN INCREASES THE NUMBER OF TESTS

RUNNING TESTS MIGHT CONSUME TIME

INCREASE INSPECTION EFFORT (HUMAN ORACLE COST)

Schema

```sql
CREATE TABLE places (
  host TEXT NOT NULL,
  path TEXT NOT NULL,
  title TEXT,
  visit_count INTEGER,
  fav_icon_url TEXT,
  PRIMARY KEY(host, path)
);

CREATE TABLE cookies (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  value TEXT,
  expiry INTEGER,
  last_accessed INTEGER,
  creation_time INTEGER,
  host TEXT,
  path TEXT,
  UNIQUE(name, host, path),
  FOREIGN KEY(host, path) REFERENCES places(host, path),
  CHECK (expiry = 0 OR expiry > last_accessed),
  CHECK (last_accessed >= creation_time)
);
```

An Example of Faults in a Database Schema

```
CREATE TABLE places (
  host TEXT NOT NULL,
  path TEXT NOT NULL,
  title TEXT,
  visit_count INTEGER,
  fav_icon_url TEXT,
  PRIMARY KEY(host, path)
);

CREATE TABLE cookies (
  id INTEGER NOT NULL,
  name TEXT PRIMARY KEY NOT NULL,
  value TEXT,
  expiry INTEGER,
  last_accessed INTEGER,
  creation_time INTEGER,
  host TEXT,
  path TEXT,
  UNIQUE(name, host, path),
  FOREIGN KEY(host, path) REFERENCES places(host, path),
  CHECK (expiry = 0),
  CHECK (last_accessed >= creation_time)
);
```

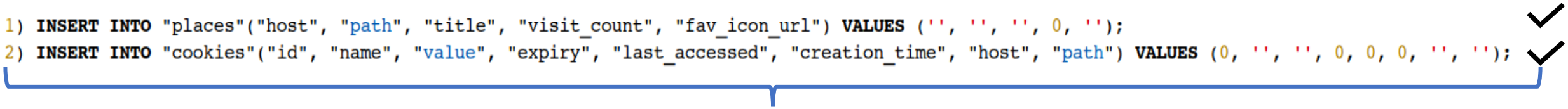
Fault 1:
Adding a PK onto the name column

Fault 2:
expiry = 0 and forgetting to add expriy > last_access

Testing Integrity Constraints

- Test 1 - Violating the PK constraint: `id INTEGER PRIMARY KEY`

```
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
```



1. Prepare an empty database

Testing Integrity Constraints

- Test 1 - Violating the PK constraint: `id INTEGER PRIMARY KEY`

```
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
```

1. Prepare an empty database

```
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');
```

2. Create Test INSERTs to exercise the PK

Testing Integrity Constraints

- Test 1 - Violating the PK constraint: `id INTEGER PRIMARY KEY`

```
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
```

1. Prepare an empty database

```
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');
```

2. Create Test INSERTs to exercise the PK

Testing Integrity Constraints

- Test 1 - Violating the PK constraint: `id INTEGER PRIMARY KEY`

```
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
```

1. Prepare an empty database

```
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');
```

2. Create Test INSERTs to exercise the PK

- Test 2 - Violating the UNIQUE constraint: `UNIQUE(name, host, path),`

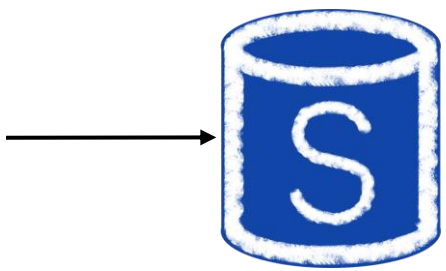
```
1) INSERT INTO places(host, path, title, visit_count, fav_icon_url) VALUES ('', '', '', 0, '')
2) INSERT INTO cookies(id, name, value, expiry, last_accessed, creation_time, host, path) VALUES (0, '', '', 0, 0, 0, '', '')
3) INSERT INTO places(host, path, title,visit_count, fav_icon_url) VALUES ('a', '', '', 0, '')
4) INSERT INTO cookies(id, name, value, expiry, last_accessed, creation_time, host, path) VALUES (1, '', '', 0, 0, 0, '', '')
```

We Can Generate Tests Automatically

```
CREATE TABLE places (
  host TEXT NOT NULL,
  path TEXT NOT NULL,
  title TEXT,
  visit_count INTEGER,
  fav_icon_url TEXT,
  PRIMARY KEY(host, path)
);

CREATE TABLE cookies (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  value TEXT,
  expiry INTEGER,
  last_accessed INTEGER,
  creation_time INTEGER,
  host TEXT,
  path TEXT,
  UNIQUE(name, host, path),
  FOREIGN KEY(host, path) REFERENCES places(host, path),
  CHECK (expiry = 0 OR expiry > last_accessed),
  CHECK (last_accessed >= creation_time)
);
```

We Can Generate Tests Automatically

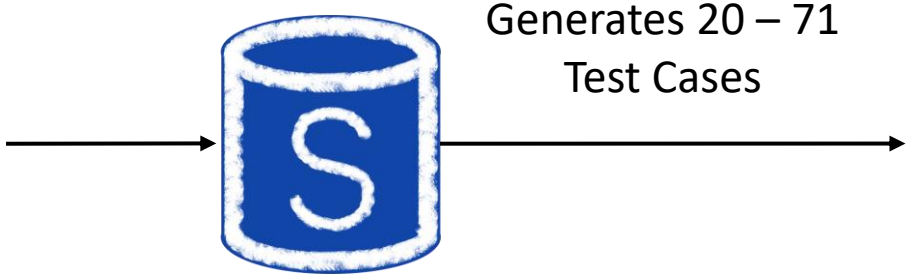
```
CREATE TABLE places (
  host TEXT NOT NULL,
  path TEXT NOT NULL,
  title TEXT,
  visit_count INTEGER,
  fav_icon_url TEXT,
  PRIMARY KEY(host, path)
);

CREATE TABLE cookies (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  value TEXT,
  expiry INTEGER,
  last_accessed INTEGER,
  creation_time INTEGER,
  host TEXT,
  path TEXT,
  UNIQUE(name, host, path),
  FOREIGN KEY(host, path) REFERENCES places(host, path),
  CHECK (expiry = 0 OR expiry > last_accessed),
  CHECK (last_accessed >= creation_time)
);
```



SchemaAnalyst

Generating Tests Automatically

```
CREATE TABLE places (
  host TEXT NOT NULL,
  path TEXT NOT NULL,
  title TEXT,
  visit_count INTEGER,
  fav_icon_url TEXT,
  PRIMARY KEY(host, path)
);

CREATE TABLE cookies (
  id INTEGER PRIMARY KEY NOT NULL,
  name TEXT NOT NULL,
  value TEXT,
  expiry INTEGER,
  last_accessed INTEGER,
  creation_time INTEGER,
  host TEXT,
  path TEXT,
  UNIQUE(name, host, path),
  FOREIGN KEY(host, path) REFERENCES places(host, path),
  CHECK (expiry = 0 OR expiry > last_accessed),
  CHECK (last_accessed >= creation_time)
);
```




```
@Test
public void test1() throws SQLException {
    // 2-cookies: UNIQUE[id] for cookies - all cols equal except id - Clause-AICC
    // 28-cookies: id is UNIQUE - AUCC
    // (¬Null[cookies: id] ∧ ¬Null[cookies: name] ∧ (vMatch[≠[cookies: name,host,path]] ∨ Null[cookies: name] ∨ Null[cookies: host] ∨ Null[cookies: pa
    // Result is: true

    // prepare the database state
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"places\"(" +
        "    \"host\", \"path\", \"title\", \"visit_count\", \"fav_icon_url\"" +
        ") VALUES (" +
        "    '', '', '', 0, ''" +
        ");"));
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"cookies\"(" +
        "    \"id\", \"name\", \"value\", \"expiry\", \"last_accessed\", \"creation_time\", \"host\", \"path\"" +
        ") VALUES (" +
        "    0, '', '', 0, 0, 0, '', ''" +
        ");"));

    // execute INSERT statements for the test case
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"places\"(" +
        "    \"host\", \"path\", \"title\", \"visit_count\", \"fav_icon_url\"" +
        ") VALUES (" +
        "    'a', '', '', 0, ''" +
        ");"));
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"cookies\"(" +
        "    \"id\", \"name\", \"value\", \"expiry\", \"last_accessed\", \"creation_time\", \"host\", \"path\"" +
        ") VALUES (" +
        "    1, 'a', '', 0, 0, 0, '', ''" +
        ");"));
}

@Test
public void test2() throws SQLException {
    // 3-cookies: UNIQUE[id] for cookies - all cols equal - Clause-AICC
    // 29-cookies: id is NOT UNIQUE - AUCC
    // (¬Null[cookies: id] ∧ ¬Null[cookies: name] ∧ (vMatch[≠[cookies: name,host,path]] ∨ Null[cookies: name] ∨ Null[cookies: host] ∨ Null[cookies: pa
    // Result is: false

    // prepare the database state
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"places\"(" +
        "    \"host\", \"path\", \"title\", \"visit_count\", \"fav_icon_url\"" +
        ") VALUES (" +
        "    '', '', '', 0, ''" +
        ");"));
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"cookies\"(" +
        "    \"id\", \"name\", \"value\", \"expiry\", \"last_accessed\", \"creation_time\", \"host\", \"path\"" +
        ") VALUES (" +
        "    0, '', '', 0, 0, 0, '', ''" +
        ");"));

    // execute INSERT statements for the test case
    assertEquals(1, statement.executeUpdate(
        "INSERT INTO \"places\"(" +
        "    \"host\", \"path\", \"title\", \"visit_count\", \"fav_icon_url\"" +
        ") VALUES (" +
        "    'a', '', '', 0, ''" +
        ");"));
```

Test data wrapped into INSERTs and into JUnit test cases

The Solution

TO USE TRADITIONAL TEST SUITE REDUCTION TECHNIQUES

Test Suite Reduction Background

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

- We can use the following approaches:
  - Random Reduction – randomly select test case until all the requirements covered
  - Additional Greedy (or called greedy in TSR literature)
  - HGS (an approach by Harrold, Gupta, and Soffa)

Greedy Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

	r1	r2	r3	r4	r5	r6

Greedy Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

	r1	r2	r3	r4	r5	r6
t1	X	X	X			

Greedy Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t3		X			X	

Greedy Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t3		X			X	
t2	X			X		

Greedy Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t3		X			X	
t2	X			X		
t4			X			X

HGS Test Suite Reduction

	r1	r2	r3	r4	r5	r6
t1	X	X	X			
t2	X			X		
t3		X			X	
t4			X			X
t5					X	
	T1	***T2***	***T3***	***T4***	***T5***	***T6***

T	R	tn	Cardinality
T1	r1	{t1, t2}	2
T2	r2	{t1, t3}	2
T3	r3	{t1, t4}	2
T4	r4	{t2}	1
T5	r5	{t3, t5}	2
T6	r6	{t4}	1

HGS Test Suite Reduction

T	R	tn	Cardinality
T1	r1	{t1, t2}	2
T2	r2	{t1, t3}	2
T3	r3	{t1, t4}	2
~~T4~~	~~r4~~	~~{t2}~~	~~1~~
T5	r5	{t3, t5}	2
~~T6~~	~~r6~~	~~{t4}~~	~~1~~

	r1	r2	r3	r4	r5	r6
t2	X			X		
t4			X			X

HGS Test Suite Reduction

T	R	tn	Cardinality
~~T1~~	~~r1~~	~~{t1, t2}~~	~~2~~
T2	r2	{~~t1~~, t3}	2
~~T3~~	~~r3~~	~~{t1, t4}~~	~~2~~
~~T4~~	~~r4~~	~~{t2}~~	~~1~~
T5	r5	{t3, t5}	2
~~T6~~	~~r6~~	~~{t4}~~	~~1~~

	r1	r2	r3	r4	r5	r6
t2	X			X		
t4			X			X

HGS Test Suite Reduction

T	R	tn	Cardinality
~~T1~~	~~r1~~	~~{t1, t2}~~	~~2~~
T2	r2	{~~t1~~, **t3**}	2
~~T3~~	~~r3~~	~~{t1, t4}~~	~~2~~
~~T4~~	~~r4~~	~~{t2}~~	~~1~~
T5	r5	{**t3**, t5}	2
~~T6~~	~~r6~~	~~{t4}~~	~~1~~

	r1	r2	r3	r4	r5	r6
t2	X			X		
t4			X			X
t3		X			X	

What is missing?

	r1	r2	r3	r4	r5	r6
t2	X			X		
t4			X			X
t3		X			X	

Can we merge similar test cases (decreasing the data restarts)?
Can we decrease the number of INSERTs (decreasing database interactions)?
Can we remove any extra redundancy?

Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit count", "fav icon url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

Equal

Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2

4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

Remove

Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');
3) INSERT INTO "places"("host", "path", "title", "visit count", "fav icon url") VALUES ('a', '', '', 0, '');
4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2



4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

Unnecessary

Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');

4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2



4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

⊃ Remove

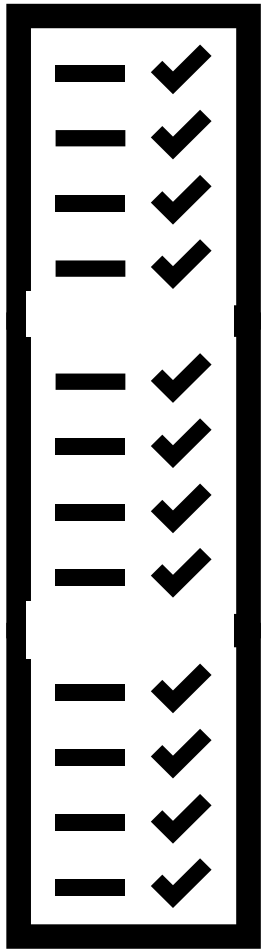
Analyzing Automatically Generated Tests

```
Test 1
1) INSERT INTO "places"("host", "path", "title", "visit_count", "fav_icon_url") VALUES ('', '', '', 0, '');
2) INSERT INTO "cookies"("id", "name", "value", "expiry", "last accessed", "creation time", "host", "path") VALUES (0, '', '', 0, 0, 0, '', '');

4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (0, 'a', '', 0, 0, 0, '', '');

Test 2

4) INSERT INTO "cookies"("id", "name", "value", "expiry", "last_accessed", "creation_time", "host", "path") VALUES (1, '', '', 0, 0, 0, '', '');
```

Merge

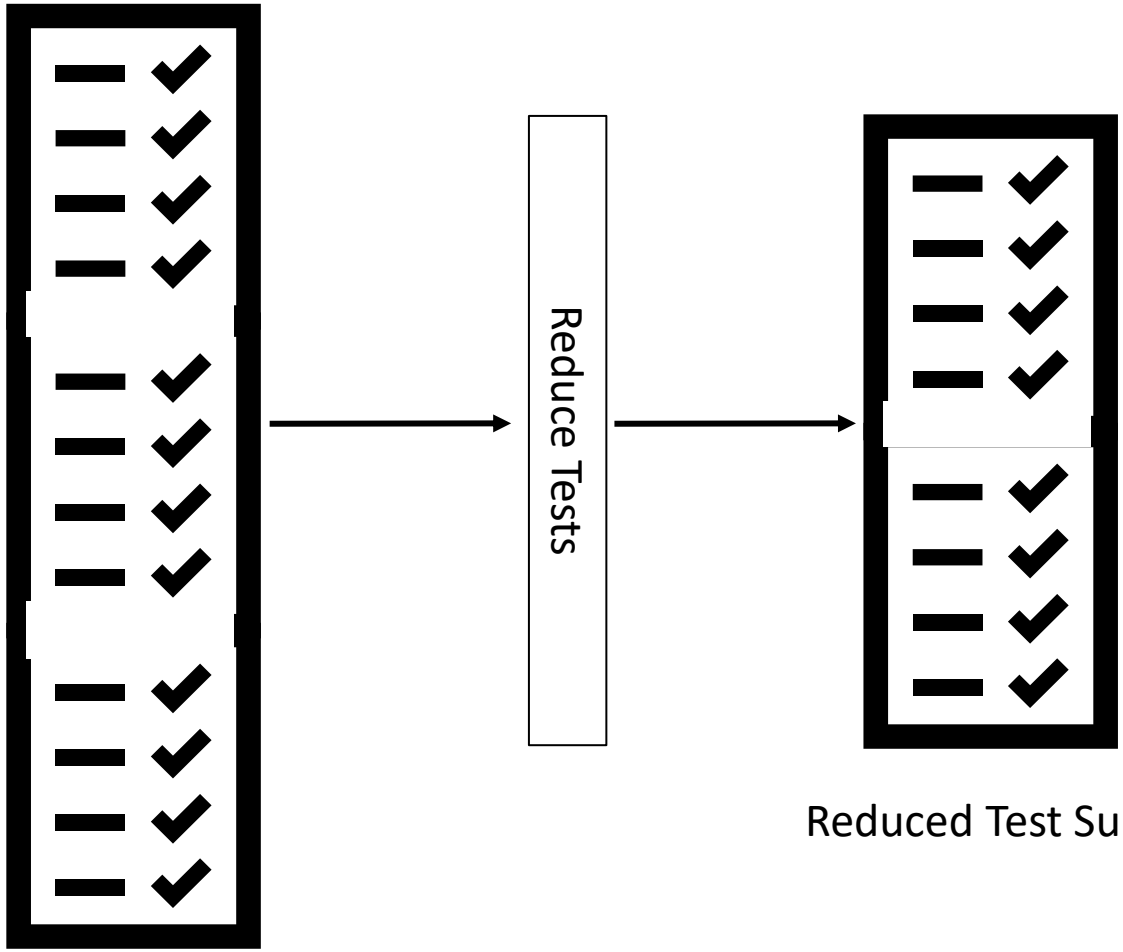
```
Test 1 & Test 2
1) INSERT INTO places(host, path, title, visit_count, fav_icon_url) VALUES ('', '', '', 0, '')
2) INSERT INTO cookies(id, name, value, expiry, last_accessed, creation_time, host, path) VALUES (0, '', '', 0, 0, 0, '', '')
3) INSERT INTO cookies(id, name, value, expiry, last_accessed, creation_time, host, path) VALUES (0, 'a', '', 0, 0, 0, '', '')
4) INSERT INTO cookies(id, name, value, expiry, last_accessed, creation_time, host, path) VALUES (1, '', '', 0, 0, 0, '', '')
```

Schema Test Integrity Constraints Combination for Efficient Reduction (STICCER)

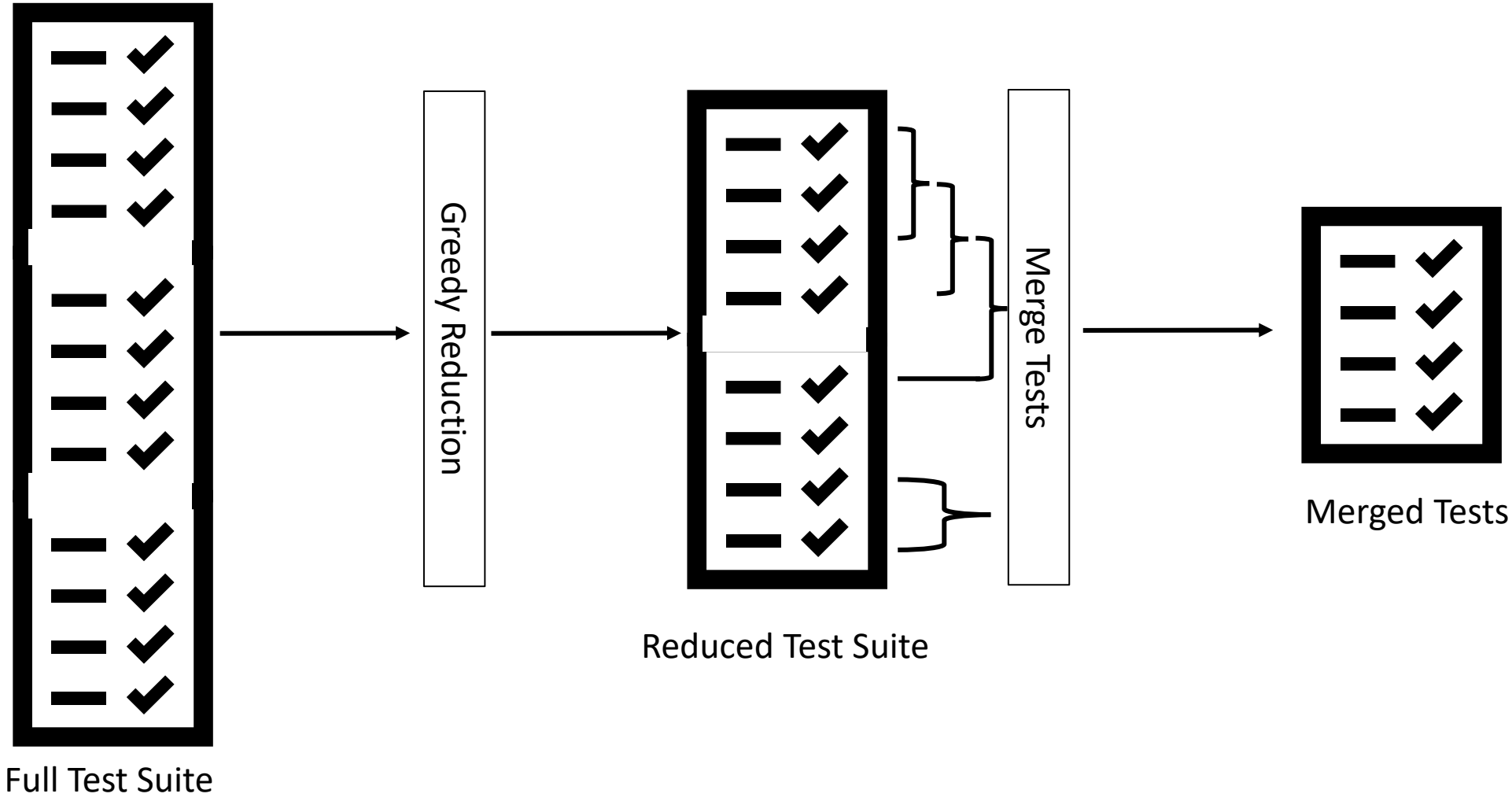


Full Test Suite

Schema Test Integrity Constraints Combination for Efficient Reduction (STICCER)



Schema Test Integrity Constraints Combination for Efficient Reduction (STICCER)



Research Questions

RQ1: Reduction Effectiveness - How effective is STICCER at ***reducing the number of test cases and INSERTs?***

RQ2: Impact on Fault Finding Capability - How is the ***fault-finding capability*** of the test suites affected?

RQ3: Impact on Test Suite and Mutation Analysis Runtime - How are the ***running times*** of the reduced test suites on mutation analysis affected?

Methodology

34 schemas

1 – 42 tables

3 – 309 columns

1 – 134 integrity constraints

Two test data generators

30 runs

Four reduction techniques

Mutation analysis

RQ1: Reduction Effectiveness Results

- iTrust schema includes 42 tables, 309 columns, 134 Integrity Constraints
  - Highest merge count = 539 merges.

RQ1: Reduction Effectiveness Results

- iTrust schema includes 42 tables, 309 columns, 134 Integrity Constraints
  - Highest merge count = 539 merges.

Metric	OTS	STICCER	Random	Greedy	HGS
Test Cases	1517	85% (235)	44% (849)	49% (776)	50% (754)
INSERTs	2204	57% (940)	45% (1212)	50% (1101)	52% (1064)

RQ1: Reduction Effectiveness Results

- iTrust schema includes 42 tables, 309 columns, 134 Integrity Constraints
  - Highest merge count = 539 merges.

Metric	OTS	STICCER	Random	Greedy	HGS
Test Cases	1517	85% (235)	44% (849)	49% (776)	50% (754)
INSERTs	2204	57% (940)	45% (1212)	50% (1101)	52% (1064)

- On Average:

Metric	STICCER	Random	Greedy	HGS
Test Cases	74%	42%	48%	50%
INSERTs	59%	43%	49%	51%

- No loss of coverage

RQ1: Reduction Effectiveness Results

- iTrust schema includes 42 t… 30… …ns, 134 Integrity Constraints
  - Highest merg…

Metric	…	…
Test C…	…	…754)
INSERTs	…	…1)

- On A…

…	…	…
Test Case…	…	50%
INSERTs	…	51%

- No loss of co…

STICCER is the most effective at reducing the number of test cases and the overall number of INSERT statements in a test suite

RQ2: Impact on Fault Finding Capability Results

	AVM-D					DOMINO				
Schemas	OTS	STICCER	Random	Greedy	HGS	OTS	STICCER	Random	Greedy	HGS
BrowserCookies	86.5	**▼86.5**	86.5	**▼86.5**	**▼86.5**	96.6	96.6	96.6	96.6	96.6
FrenchTowns	83.3	*▼**80.3**	*▼**80.3**	*▼**80.3**	*▼**81.8**	95.5	95.5	95.5	95.5	95.5
iTrust	83.6	*▼**83.6**	*▼**83.6**	*▼**83.6**	*▼**83.6**	99.2	99.2	99.2	99.2	99.1
NistWeather	93.8	*▼**90.6**	93.8	*▼**90.6**	93.8	100.0	100.0	100.0	100.0	100.0
NistXTS749	92.0	92.0	**▼92.0**	92.0	*▼**88.0**	94.0	94.0	94.0	94.0	94.0
RiskIt	89.3	89.3	**▼89.3**	89.3	*▼**88.8**	99.5	99.5	99.5	99.5	99.5
UnixUsage	98.2	98.2	98.2	98.2	*▼**97.3**	100.0	100.0	100.0	100.0	100.0
WordNet	87.4	*▼**86.3**	**▼87.4**	*▼**86.3**	*▼**86.3**	99.0	99.0	99.0	99.0	99.0

- AVM-D generated and reduced test case impacted the Fault-Finding Capability
- 5 test suites were impacted by STICCER reduction
- Maximum impact was only 3.2% compared to OTS

RQ2: Impact on Fault Finding Capability Results

Schemas	AV… TICCER	Ran…	DOMI… ICCE…	…n	Greedy	HGS
BrowserCookie…				96.6	96.6	96.6
FrenchTowns						
iTrust						100.0
					94.0	94.0
				5	99.5	99.5
					100.0	100.0
Word…					99.0	99.0

Mutation scores of the test suites were preserved following reduction. While some test suites experienced a non-substantial drop in mutation score (3.2% maximum)

…t-Finding Capability

- 5 test…
- Maxim…

RQ3: Impact on Test Suite and Mutation Analysis Runtime Results

- Mutation analysis runtime:
  - STICCER test suites were 5X faster than the original test suite
  - 2.5X faster than other traditional reduction techniques
- Example: iTrust test suites and running mutation analysis

Unit	OTS	STICCER	Random	Greedy	HGS
Minutes	38	7 (+2 reduction)	21	19	18.5

RQ3: Impact on Test Suite and Mutation Analysis Runtime Results

- Mutation an[illegible]ysis runtime
  - STICCER te[illegible] we[illegible] [illegible]inal test suite
  - 2.5X faster
- Exam[illegible] [illegible]lysis

In general, STICCER reduced test suites ran faster compared to the OTS and those reduced by other techniques.

Conclusions and Future Work

- STICCER = Reduce + Merge
- Outperforms other reduction techniques and maintains coverage
- A maximum of 3.2% loss of fault-finding capabilities (mutation)
- Mutation analysis execution:
  - 5X faster than the original test suite
  - 2.5X faster than other traditional reduction techniques
- Future Work:
  - Integrate STICCER within the test data generator
  - Enhance STICCER with multi-objective test data generators
  - Adapt STICCER into traditional programs that manipulate complex state in other formats

github.com/schemaanalyst/schemaanalyst