Hybrid Methods for Reducing Database Schema Test Suites: Experimental Insights from Computational and Human Studies

Abdullah Alsharif
Saudi Electronic University

Gregory M. Kapfhammer
Allegheny College

Phil McMinn
University of Sheffield

ABSTRACT

Given that a relational database is a critical component of many software applications, it is important to thoroughly test the integrity constraints of a database's schema, because they protect the data. Although automated test data generation techniques ameliorate the otherwise manual task of database schema testing, they often create test suites that contain many, sometimes redundant, tests. Since prior work presented a hybridized test suite reduction technique, called STICCER, that beneficially combined Greedy test suite reduction with a test merging method customized for database schemas, this paper experimentally evaluates a different hybridization. Motivated by prior results showing that test suite reduction with the Harrold-Gupta-Soffa (HGS) method can be more effective than Greedy at reducing database schema test suites, this paper evaluates an HGS-driven STICCER variant with both a computational and a human study. Using 34 database schemas and tests created by two test data generators, the results from the computational study reveal that, while STICCER is equally efficient and effective when combined with either Greedy or HGS, it is always better than the isolated use of either Greedy or HGS. Involving 27 participants, the human study shows that, when compared to test suites reduced by HGS, those reduced by a STICCER-HGS hybrid allow humans to inspect test cases faster, but not always more accurately.

ACM Reference Format:
Abdullah Alsharif, Gregory M. Kapfhammer, and Phil McMinn. 2020. Hybrid Methods for Reducing Database Schema Test Suites: Experimental Insights from Computational and Human Studies. In *IEEE/ACM 1st International Conference on Automation of Software Test (AST '20), October 7–8, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3387903.3389305

1 INTRODUCTION

Many software applications rely on a relational database for critical data storage, thus leading industry experts to advise that they be rigorously tested to ensure their correctness [6, 17]. Developing a relational database involves the challenging task of specifying a schema that has integrity constraints designed to protect the data in the database. Since the incorrect definition of the database schema (i.e., omitting constraints or adding the wrong constraints) can lead to a failure that corrupts the database's state [42], testers can use test data generators, like AVM-D and Domino, to automatically create tests for a schema's integrity constraints [2]. Although these generators speed up the testing process, the test suites that they create may have many tests with numerous, and sometimes similar, database interactions, suggesting the need for test suite reduction.

While there are general-purpose methods for reducing a test suite [59], our prior work presented STICCER, a hybrid method that combined greedy test suite reduction with a merging approach for database schema testing [4]. Yet, since other hybridizations are also possible, this paper presents two empirical studies investigating test suite reduction techniques for relational database schemas:

Computational Study. When comparing two well-known test suite reduction methods, called Greedy [10] and Harrold-Gupta-Soffa (HGS) [20], our prior work showed that HGS achieved an average level of reduction of 46% and 50% for database schema test suites generated by AVM-D and Domino, respectively [4]. This result represents a greater level of reduction than that achieved by the Greedy method, which was 43% and 48% reduction for tests resulting from the same test generators. The first set of research questions posed by the Computational Study in Section 3 of this paper, therefore, explore the theme characterized by the general question "*Are further efficiencies possible if we reduce test suites prior to merging with STICCER using HGS, as opposed to Greedy?*".

Human Study. As we explain in Section 2, the human oracle costs arising from inspecting tests are an important aspect of research in automated test suite generation, that hitherto remains unevaluated in the context of hybrid techniques like STICCER. The second set of research questions studied in this paper, which appear as part of the Human Study in Section 4, concern the matter posed by the general question "*While STICCER produces test suites with fewer test cases and statements overall, does it lower human oracle costs; or are the tests more difficult to understand, therefore increasing costs?*".

Using 34 relational database schemas, two state-of-the-art test data generators, and the two hybridized and two traditional test suite reduction methods, this paper's Computational Study finds that, while the hybridized methods outperform the stand-alone use of either Greedy or HGS, there is, surprisingly, no significant benefit to using HGS instead of Greedy in STICCER. Since this paper's focus is on the benefits that may arise from combining HGS and STICCER, the Human Study asked 27 participants to act as testers who had to manually inspect test suites that had been reduced by either STICCER-HGS or HGS. This paper's Human Study reveals that, compared to those produced by HGS, the reduced test suites made by STICCER-HGS help humans to complete test inspection tasks faster, but not more accurately. Along with confirming the benefits that accrue from hybridizing STICCER with either Greedy or HGS, this paper's two studies suggest that, while test suite reduction may make certain testing tasks — like assessing test suite adequacy through mutation analysis — more efficient, it will not always benefit the humans testers who must inspect the reduced test suites.

AST '20, October 7–8, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7957-1/20/05…$15.00
https://doi.org/10.1145/3387903.3389305

2 BACKGROUND

Relational Database Schemas. A relational database management system (RDBMS), such as SQLite [50] or Postgres [45], is software that hosts and manages one or more relational databases. Each database is defined by a schema through SQL statements, as shown by Figure 1's example. A schema defines one or more *tables*, each involving a set of *columns* (i.e., "`id`" through to "`date_of_birth`" in the example) that describe the data (in this instance, information about an individual person) to be stored in the table's *rows*. Each column has a data type (e.g., `int`, `VARCHAR` — a string with a defined number of characters, and `date` — a day, month, and year). Finally, relational database schemas feature *integrity constraints* that a developer specifies in the definition of an individual column or the wider table. Integrity constraints play a significant role in maintaining the reliability, consistency, and coherency of data [25]. In the example, both the `id` and `email` columns must store distinct values, since they are constrained with `PRIMARY KEY` and `UNIQUE` constraints, respectively. Columns marked with "`NOT NULL`" cannot store `NULL` values. Finally, the `CHECK` constraint declaration only allows one selected value from the list in the `gender` column for each row.

Mistakes made by developers when defining integrity constraints (e.g., omitting constraints or adding the wrong constraints) can manifest themselves in software failures that corrupt data [17]. For instance, not having a `UNIQUE` on the `email` column in the example of Figure 1 would mean different (or duplicate) people in different rows of the database potentially having the same email address. Conversely, a developer unintentionally adding a `UNIQUE` on the first name column would obstruct the database recording information where more than one person has the same first name. Furthermore, different relational database management systems have subtly different, inconsistent interpretations of the SQL standard, of which developers may not necessarily be aware. For example, SQLite allows the insertion of `NULL` into primary keys in certain circumstances [51], yet PostgreSQL forbids this behavior [45]. As such, developers need to test their schemas to check their assumptions. However, the common expectation is that schemas are implicitly correct [5, 6, 13, 46], and as a result, their testing is often neglected. Yet, for all of these reasons, industry experts recommend thorough testing of the integrity constraints in a database schema [17].

Test Generation for Integrity Constraints. McMinn et al. defined a family of coverage criteria for testing the specification of integrity constraints in relational database schemas [42]. These criteria specify test requirements that involve exercising each constraint as *true* and *false* — that is, designing test cases with SQL `INSERT` statements and values to either *satisfy* an integrity constraint (i.e., the RDBMS accepts and stores the data) or *violate* it (i.e., the RDBMS rejects the data). McMinn et al. found that the combination of three different criteria — "Clause-Based Active Integrity Constraint Coverage" (ClauseAICC), "Active Unique Column Coverage" (AUCC), and "Active Null Column Coverage" (ANCC) — was the best at identifying systematically seeded faults [42]. The ClauseAICC criterion requires exercising the roles of individual columns within composite keys when exercising each constraint, as well as the individual clauses of `CHECK` constraints. AUCC exercises all columns with unique and non-unique (i.e., identical) values, while ANCC exercises each column with `NULL` and non-`NULL` values.

```
CREATE TABLE person (
 id int NOT NULL PRIMARY KEY,
 last_name varchar(45) NOT NULL,
 first_name varchar(45) NOT NULL,
 email varchar(45) NOT NULL UNIQUE,
 gender varchar(6) NOT NULL,
 date_of_birth date NOT NULL,
 CHECK (gender IN ('Male', 'Female', 'Other')));
```

Figure 1: An Example of a Relational Database Schema

The *SchemaAnalyst* tool [43] automates the generation of test data to satisfy coverage criteria for integrity constraints, providing two state-of-the-art test generation techniques called AVM-D [28] and Domino [2]. Both of these generators populate a sequence of `INSERT` statements with test data designed to satisfy a test coverage requirement. AVM-D is an implementation of the Alternating Variable Method [19, 29–31, 39], a search-based technique that uses a fitness function to guide a search for test data. AVM-D maintains one test data vector that it modifies throughout the search, initializing it to "default" values (e.g., zero for integers and empty strings for text). Modifications are guided by traditional search-based distance metrics used for predicate testing (e.g., those that appear in branches of a program) [53]. For example, if a test requirement needs two identical values $x = y$ (e.g., to satisfy a `FOREIGN KEY`), a distance metric is formulated as $d = |x - y|$, where d calculates the closeness of the two values x and y, and where $d = 0$ indicates that the search has found identical values. Domino is based on a random test data generation. Domino "tunes" data to a specific test requirement [2], primarily through copying values in an `INSERT` statement for one table column to another. This enables it to generate matching values to satisfy `FOREIGN KEY`s and violate `PRIMARY KEY`s. Conversely, random values are used to generate non-matches (for example, to violate `FOREIGN KEY`s or satisfy `PRIMARY KEY`s), or to satisfy and violate the predicates embedded in `CHECK` constraints.

Since the coverage criteria for testing integrity constraints involve many test requirements, the test suites generated for them may be similarly large in terms of the number of tests [4]. This paper aims to improve the capability of reduction methods to decrease test suite size while still maintaining their level of test coverage.

Test Suite Reduction Methods. Reducing test suites while maintaining coverage is a problem equivalent to that of minimal set cover and as such is NP-complete [24]. However, many techniques exist that are effective at producing approximate solutions. Here, we introduce the ones that we have implemented into *SchemaAnalyst*.

The *Greedy* method (also known as "additional greedy") [59] works by populating an initially empty test suite through iteratively selecting the test cases from the original test suite that cover the most test requirements currently uncovered by the reduced suite.

The *HGS* method developed by Harrold, Gupta, and Soffa [20] works by creating intermediate test suites containing test cases that cover each individual test requirement. HGS starts by adding test cases to the reduced test suite from the intermediate test suites with cardinality 1 (i.e., test cases that cover only that test requirement). HGS then "marks" test suites that also cover these requirements, so that they are no longer considered by further steps of the algorithm. HGS then proceeds to repeatedly select test cases in unmarked test suites of increasing cardinality. In this way, HGS avoids suboptimal traps that the Greedy method can fall into that are caused by selecting test cases that cover many of the same test requirements.

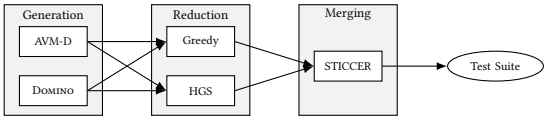

Figure 2: Phases that Generate, Reduce, and Merge Tests

Finally, *STICCER* is a test suite reduction method developed especially for reducing test suites designed to test relational database schema integrity constraints [4]. STICCER stands for **S**chema **T**est **I**ntegrity **C**onstraints **C**ombination for **E**fficient **R**eduction. First, STICCER takes a test suite reduced using, for example, the Greedy method [4]. It then proceeds to merge the tests in the reduced suite by "sticking" sub-sequences of statements from different test cases together to produce a new replacement test case. STICCER produces a "candidate" merged test t_m by first removing the setup steps from t_2, and then appending the remaining test statements to the end of t_1. If t_m has the same coverage as the original two tests t_1 and t_2, then STICCER replaces t_1 and t_2 with t_m in the test suite. By doing this, STICCER not only produces a test suite with fewer test cases, it produces a test suite with a smaller number of total statements overall: by re-using a test t_1 to prepare the database state for another test t_2, it can discard the now redundant "setup" steps in t_2 that essentially performed the same task. Alsharif et al. [4] found that database schema test suites reduced by Greedy and subsequently merged with STICCER were up to 2.5 times faster to run than those test suites reduced by using traditional methods such HGS, or using just Greedy by itself, and 5 times faster than the original test suite. In Section 3, we study a new hybridization of STICCER that uses HGS prior to merging, instead of Greedy.

Human Oracle Costs. Automatically generated tests based on white-box coverage criteria, such as those used in this paper, tend not to be accompanied by a specification or model of correct system behavior. As such, testers often need to perform one of three manual steps: (1) evaluate the outcomes of each test case every time it runs, so as to ascertain whether it passed or failed; (2) manually add `assert` statements to the tests to automatically perform this checking in the future; or (3) check the `assert` statements that the test generation tool may have automatically generated, but which merely reflect current system behavior and may therefore be incorrect. The effort, or cost, associated with this activity — that is tedious and error-prone one for humans when the test suites are of a significant size — is referred to as the "human oracle cost" [1, 7, 18, 41].

Section 4 reports on a Human Study that characterizes the oracle costs associated with HGS and STICCER hybridized with HGS.

3 COMPUTATIONAL STUDY

The Computational Study focuses on a previously unstudied configuration of STICCER that uses HGS — as opposed to the previously studied version that used Greedy [4] — to reduce test suites before merging them. Figure 2 shows the different configurations of the test suite generation and reduction pipeline this section evaluates.

In the first phase, test suite *generation*, we use *SchemaAnalyst* to generate test suites using either AVM-D or Domino. The second stage comprises *reduction* with either the HGS or Greedy method. The third phase involves *merging* test cases with STICCER, resulting in two more test suite reduction techniques. We refer to the configuration of STICCER in the experiments that uses HGS prior to merging, and which forms the focus of this particular study, as "STICCER-HGS". Meanwhile, we refer to the configuration of STICCER that uses Greedy instead, which was the main feature of our previous paper [4], as "STICCER-GRD". Finally, for brevity, we hereafter refer to both of the reduction and merging phases when discussing STICCER as simply "reduction", since both reduce the size of the eventual test suites produced by STICCER-HGS and STICCER-GRD. We aim to answer three research questions:

Table 1: The Relational Database Schemas Studied

			Integrity Constraints					
Schema	Tables	Columns	Check	Foreign Key	Not Null	Primary Key	Unique	Total
ArtistSimilarity	2	3	0	2	0	1	0	3
ArtistTerm	5	7	0	4	0	3	0	7
BankAccount	2	9	0	1	5	2	0	8
BookTown	22	67	2	0	15	11	0	28
BrowserCookies	2	13	2	1	4	2	1	10
Cloc	2	10	0	0	0	0	0	0
CoffeeOrders	5	20	0	4	10	5	0	19
CustomerOrder	7	32	1	7	27	7	0	42
DellStore	8	52	0	0	39	0	0	39
Employee	1	7	3	0	0	1	0	4
Examination	2	21	6	1	0	2	0	9
Flights	2	13	1	1	6	2	0	10
FrenchTowns	3	14	0	2	13	0	9	24
Inventory	1	4	0	0	0	1	1	2
Iso3166	1	3	0	0	2	1	0	3
IsoFlav_R2	6	40	0	0	0	0	5	5
iTrust	42	309	8	1	88	37	0	134
JWhoisServer	6	49	0	0	44	6	0	50
MozillaExtensions	6	51	0	0	0	2	5	7
MozillaPermissions	1	8	0	0	0	1	0	1
NistDML181	2	7	0	1	0	1	0	2
NistDML182	2	32	0	1	0	1	0	2
NistDML183	2	6	0	1	0	0	1	2
NistWeather	2	9	5	1	5	2	0	13
NistXTS748	1	3	1	0	1	0	1	3
NistXTS749	2	7	1	1	3	2	0	7
Person	1	5	1	0	5	1	0	7
Products	3	9	4	2	5	3	0	14
RiskIt	13	57	0	10	15	11	0	36
StackOverflow	4	43	0	0	5	0	0	5
StudentResidence	2	6	3	1	2	2	0	8
UnixUsage	8	32	0	7	10	7	0	24
Usda	10	67	0	0	31	0	0	31
WordNet	8	29	0	0	22	8	1	31
Total	186	1044	38	49	357	122	24	590

RQ1: Reduction Effectiveness. How does STICCER-HGS compare at reducing test suites to STICCER-GRD, HGS, and Greedy?

RQ2: Fault Finding Capability. How does the fault-finding capability of test suites reduced by STICCER-HGS compare to those reduced by STICCER-GRD, HGS, and Greedy?

RQ3: Reduction and Mutation Analysis Runtime. How does the overall time taken to (a) reduce test suites and then (b) perform mutation analysis on them compare when using either STICCER-HGS or STICCER-GRD as the test suite reduction technique?

We began by using the publicly available *SchemaAnalyst* tool [43] to generate test suites with both Domino and AVM-D for each of our subject schemas detailed in Table 1. We configured both test data generators to fulfill the "ClauseAICC+ANCC+AUCC" combination of coverage criteria (introduced in Section 2), with a termination criterion of 100,000 test data evaluations per test requirement (should test data not be found earlier than this limit). Since both Domino and AVM-D are based on random number generation, we used *SchemaAnalyst* to repeat test generation 30 times. We then used *SchemaAnalyst*'s implementations of STICCER-HGS, STICCER-GRD, HGS, and Greedy to reduce each of the test suites, recording the execution time taken. Studying the adequacy assessment process for the reduced test suites, we next used *SchemaAnalyst* to run mutation analysis on each of them, applying Wright et al.'s mutation

operators [57], again recording the time taken. To conduct our experiments, we used a Linux workstation with a quad-core 2.4GHz CPU and 12GB of RAM, running Ubuntu 14.04 with a 3.13.0–44 GNU/Linux 64-bit kernel; generating test data for the SQLite version 3.8.2 RDBMS with the "in-memory" mode setting enabled.

We answer **RQ1** by recording (a) number of test cases and (b) total number of statements (i.e., INSERTs) in each test suite before and after reduction with each of the four reduction techniques. Our motivation behind measuring the total number of statements in a test suite, in addition to its size in terms of test cases, is to rule out the possibility that STICCER simply produces fewer test cases by merely appending them together, and thereby giving a false impression of effectiveness. We calculate reduction using the equation $(1 - |RTS| \div |OTS|) \times 100$; where *RTS* and *OTS* correspond to the reduced test suite and the original test suite, respectively. The numbers we apply in the calculation depend on whether we want to know (a) the reduction in the number of test cases, in which case *RTS* and *OTS* are the number of tests in each respective suite; or (b) the reduction in the number of statements, in which case *RTS* and *OTS* are the total number of statements in each test suite. In our answer, we report the median values for the 30 test suites generated for each schema with each test data generation technique (i.e., AVM-D and DOMINO), reduced by each of the four reduction techniques. We answer **RQ2** by reporting the median mutation scores reported by *SchemaAnalyst* of each set of 30 generated test suites, in both unreduced form and following reduction by either STICCER-HGS, STICCER-GRD, HGS, and Greedy. Finally, we answer **RQ3** by reporting the median time to reduce the 30 test suites, and the median time taken by the mutation analysis of them, comparing it to the median time taken for mutation analysis to execute with the original, unreduced test suites.

Statistical Analysis. As part of our answers to RQs 1–3, Tables 2–5 report if a technique was statistically better or worse than STICCER-HGS by presenting its numerical values in boldface. We further annotate a value with "●" if STICCER-HGS performed significantly *better* when applying the Mann-Whitney U-test with $\alpha = 0.05$ and with "○" if it was significantly *worse*. Finally, we annotate a value with "★" if the underlying distributions of the data for a technique have a large Vargha and Delaney $\hat{A}$ effect size (i.e., $\hat{A} < 0.29$ or > 0.71 [55]) when compared with that of STICCER-HGS.

Threats to Validity. While our set of subjects may not generalize the claims of our study to *all* possible schemas, we made every effort in our previous work [2, 4, 40, 42, 44, 56] to develop as diverse a subject set as possible, featuring different types of integrity constraints and a wide range of complexity (1–42 tables, 3–309 columns, and 1–134 constraints). Other threats include the stochastic behavior of the test data generators, which is subject to the chance effects of random number generation; and the timing of the test generation techniques and mutation analysis, which are subject, to, for example, interference from operating system events. We mitigated both of these possibilities by repeating the experiments 30 times and by using non-parametric statistical tests to analyze the results, since we cannot make any assumptions about the normality of our results' distributions. We mitigated the possibility of defects in the implementation of STICCER-HGS and the code to run our experiments with the *SchemaAnalyst* tool by writing and applying unit tests, which are available at https://github.com/schemaanalyst/schemaanalyst.

Successful Merge of DOMINO Test Cases

	id	last_name	first_name	gender	date_of_birth	
t_1	**-458**	'ada'	'djd'	'Male'	'2008-06-10'	✓
t_2	**0**	'ib'	'edvbewwyg'	'Other'	'1992-03-17'	✓

Unsuccessful Merge of AVM-D Test Cases

	id	last_name	first_name	gender	date_of_birth	
t_1	**0**	''	''	'Male'	'2000-01-01'	✓
t_2	**0**	''	''	'Other'	'2000-01-01'	✗

Figure 3: STICCER's Attempts to Merge Test Cases

It is also possible that the ordering of test cases passed to STICCER for merging could be a considered a potential threat to validity, as this could affect which test cases are merged with one another, and hence the results we obtain. Yet, we experimented with randomizing and reversing the order of test cases passed to STICCER from HGS ("irreplaceable" tests first) and Greedy (most test-requirement-covering tests first), but did not observe significant differences in the results. Therefore, we continued to use the default order of tests provided by the reduction techniques prior to merging.

Answer to RQ1: Reduction Effectiveness. Tables 2 and 3 show the median reduction effectiveness of each technique at decreasing the number of test cases for each schema and the total number of statements (i.e., database INSERTs) in the test cases of the test suites, respectively. Both tables report effectiveness for test suites generated by AVM-D and DOMINO, because, as the tables reveal, the reduction techniques vary in performance depending on which test generation technique was initially used. Overall, we see four different trends in the two tables, which we explain next.

Firstly, STICCER-HGS significantly outperforms HGS and Greedy, regardless of initial test generation technique, just as STICCER-GRD did in our previous study [4]. Table 2 shows that STICCER-HGS is significantly better than HGS and Greedy at reducing the number of test cases for all schemas, while Table 3 shows that STICCER-HGS also significantly reduces the total number of statements in the tests suites compared to HGS and Greedy, for all but a few schemas.

Secondly, STICCER-HGS is, overall, more effective at reducing DOMINO-generated test suites than those made by AVM-D. Table 2 shows an overall reduction mean of 72% with DOMINO-generated test suites, compared to 67% with AVM-D-generated suites. As we previously observed in our prior paper [4], the same is true for STICCER-GRD, where the averages are 74% with DOMINO compared to 66% with AVM-D. Our explanation for this phenomenon centers on the data values that each test data generator typically generates. AVM-D repeats "default" values such as empty strings and zero numerical values, aiming to keep test cases as simple as possible. However, this frustrates STICCER's attempts to merge INSERT statements, since the use of the same values across different test cases can inadvertently trigger primary key and UNIQUE constraint violations when two tests are combined. Figure 3 illustrates this phenomenon with an example. One of the test requirements that needs to be preserved by the merged test case in this instance are unique values for the gender field. Yet, the re-use of zero as an id value for the two tests that STICCER is attempting to merge in the AVM-D case results in a primary key violation. As such, the merged test case is not equivalent to the two original test cases, where the database state would have been reset between their execution.

Table 2: Median Test Case Reduction Effectiveness

Refer to Section 3 for the calculation of the higher-is-better test case reduction effectiveness metric, and the meaning of the symbols ●, ○, and ★. The numbers in brackets indicate the median number of statements in the reduced test suite for a schema, followed by the median number of statements in its original test suite, respectively.

	AVM-D								DOMINO							
Schema	STICCER-HGS		STICCER-GRD		HGS		Greedy		STICCER-HGS		STICCER-GRD		HGS		Greedy	
ArtistSimilarity	63%	(7/19)	63%	(7/19)	★●**42%**	(11/19)	★●**32%**	(13/19)	63%	(7/19)	63%	(7/19)	★●**32%**	(13/19)	★●**37%**	(12/19)
ArtistTerm	65%	(15/43)	65%	(15/43)	★●**37%**	(27/43)	★●**30%**	(30/43)	63%	(16/43)	★○**65%**	(15/43)	★●**30%**	(30/43)	★●**30%**	(30/43)
BankAccount	68%	(12/37)	68%	(12/37)	★●**43%**	(21/37)	★●**38%**	(23/37)	70%	(11/37)	★●**70%**	(11/37)	★●**49%**	(19/37)	★●**43%**	(21/37)
BookTown	63%	(100/269)	★●**58%**	(113/269)	★●**46%**	(144/269)	★●**38%**	(167/269)	65%	(94/269)	★○**68%**	(87/269)	★●**49%**	(138/269)	★●**42%**	(156/269)
BrowserCookies	63%	(26/71)	★●**62%**	(27/71)	★●**59%**	(29/71)	★●**56%**	(31/71)	77%	(16/71)	76%	(17/71)	★●**56%**	(31/71)	★●**55%**	(32/71)
Cloc	90%	(4/40)	90%	(4/40)	★●**50%**	(20/40)	★●**48%**	(21/40)	75%	(10/40)	★○**85%**	(6/40)	★●**57%**	(17/40)	★●**51%**	(20/40)
CoffeeOrders	64%	(32/90)	64%	(32/90)	★●**42%**	(52/90)	★●**41%**	(53/90)	70%	(27/90)	★○**74%**	(23/90)	★●**47%**	(48/90)	★●**44%**	(50/90)
CustomerOrder	44%	(71/126)	★●**40%**	(76/126)	★●**37%**	(79/126)	★●**33%**	(84/126)	64%	(45/126)	★●**63%**	(46/126)	★●**39%**	(77/126)	★●**35%**	(82/126)
DellStore	86%	(24/177)	86%	(24/177)	★●**38%**	(110/177)	★●**35%**	(115/177)	67%	(59/177)	★○**71%**	(52/177)	★●**42%**	(102/177)	★●**41%**	(105/177)
Employee	71%	(11/38)	★○**74%**	(10/38)	★●**50%**	(19/38)	★●**50%**	(19/38)	82%	(7/38)	80%	(8/38)	★●**61%**	(15/38)	★●**61%**	(15/38)
Examination	72%	(30/107)	72%	(30/107)	★●**52%**	(51/107)	★●**51%**	(52/107)	85%	(16/107)	★○**87%**	(14/107)	★●**64%**	(38/107)	★●**64%**	(38/107)
Flights	71%	(18/62)	★●**69%**	(19/62)	★●**58%**	(26/62)	★●**58%**	(26/62)	71%	(18/62)	71%	(18/62)	★●**53%**	(29/62)	★●**52%**	(30/62)
FrenchTowns	38%	(33/53)	★○**40%**	(32/53)	★●**34%**	(35/53)	★●**36%**	(34/53)	58%	(22/53)	60%	(21/53)	★●**34%**	(35/53)	★●**32%**	(36/53)
Inventory	72%	(5/18)	★●**67%**	(6/18)	★●**44%**	(10/18)	★●**39%**	(11/18)	72%	(5/18)	78%	(4/18)	★●**56%**	(8/18)	★●**56%**	(8/18)
IsoFlav_R2	76%	(43/177)	★●**75%**	(45/177)	★●**50%**	(88/177)	★●**49%**	(90/177)	78%	(39/177)	★○**81%**	(34/177)	★●**62%**	(66/177)	★●**60%**	(70/177)
Iso3166	58%	(5/12)	58%	(5/12)	★●**33%**	(8/12)	★●**33%**	(8/12)	58%	(5/12)	58%	(5/12)	★●**42%**	(7/12)	★●**33%**	(8/12)
iTrust	58%	(631/1517)	★●**57%**	(646/1517)	★●**44%**	(847/1517)	★●**43%**	(872/1517)	80%	(297/1517)	★○**85%**	(235/1517)	★●**50%**	(754/1517)	★●**49%**	(776/1517)
JWhoisServer	39%	(97/158)	★●**37%**	(100/158)	★●**35%**	(103/158)	★●**33%**	(106/158)	65%	(56/158)	★○**70%**	(48/158)	★●**37%**	(99/158)	★●**35%**	(103/158)
MozillaExtensions	75%	(57/229)	●**75%**	(57/229)	★●**50%**	(115/229)	★●**60%**	(92/229)	82%	(42/229)	★○**85%**	(35/229)	★●**64%**	(83/229)	★●**63%**	(84/229)
MozillaPermissions	73%	(9/33)	73%	(9/33)	★●**48%**	(17/33)	★●**48%**	(17/33)	79%	(7/33)	★○**88%**	(4/33)	★●**64%**	(12/33)	★●**58%**	(14/33)
NistDML181	82%	(7/38)	★●**79%**	(8/38)	★●**53%**	(18/38)	★●**53%**	(18/38)	79%	(8/38)	●**76%**	(9/38)	★●**58%**	(16/38)	★●**55%**	(17/38)
NistDML182	90%	(19/190)	★●**89%**	(20/190)	★●**57%**	(81/190)	★●**57%**	(82/190)	88%	(22/190)	○**89%**	(21/190)	★●**62%**	(73/190)	★●**60%**	(76/190)
NistDML183	82%	(6/34)	82%	(6/34)	★●**53%**	(16/34)	★●**47%**	(18/34)	76%	(8/34)	76%	(8/34)	★●**53%**	(16/34)	★●**50%**	(17/34)
NistWeather	64%	(20/56)	●**64%**	(20/56)	★●**43%**	(32/56)	★●**45%**	(31/56)	80%	(11/56)	★○**82%**	(10/56)	★●**46%**	(30/56)	★●**45%**	(31/56)
NistXTS748	62%	(6/16)	62%	(6/16)	★●**44%**	(9/16)	★●**38%**	(10/16)	75%	(4/16)	69%	(5/16)	★●**50%**	(8/16)	★●**50%**	(8/16)
NistXTS749	63%	(13/35)	★●**57%**	(15/35)	★●**51%**	(17/35)	★●**46%**	(19/35)	69%	(11/35)	69%	(11/35)	★●**49%**	(18/35)	★●**49%**	(18/35)
Person	50%	(10/20)	50%	(10/20)	★●**40%**	(12/20)	★●**40%**	(12/20)	70%	(6/20)	★○**80%**	(4/20)	★●**35%**	(13/20)	★●**30%**	(14/20)
Products	67%	(17/52)	67%	(17/52)	★●**48%**	(27/52)	★●**44%**	(29/52)	70%	(16/52)	69%	(16/52)	★●**46%**	(28/52)	★●**44%**	(29/52)
RiskIt	56%	(110/250)	★●**51%**	(122/250)	★●**48%**	(130/250)	★●**43%**	(142/250)	64%	(91/250)	63%	(92/250)	★●**52%**	(120/250)	★●**48%**	(129/250)
StackOverflow	93%	(12/171)	93%	(12/171)	★●**50%**	(86/171)	★●**48%**	(89/171)	78%	(38/171)	★○**84%**	(28/171)	★●**60%**	(68/171)	★●**60%**	(68/171)
StudentResidence	62%	(13/34)	62%	(13/34)	★●**44%**	(19/34)	★●**41%**	(20/34)	74%	(9/34)	74%	(9/34)	★●**47%**	(18/34)	★●**47%**	(18/34)
UnixUsage	56%	(64/147)	★●**52%**	(70/147)	★●**47%**	(78/147)	★●**43%**	(84/147)	71%	(42/147)	★○**73%**	(40/147)	★●**50%**	(73/147)	★●**48%**	(76/147)
Usda	88%	(30/247)	88%	(30/247)	★●**44%**	(139/247)	★●**40%**	(147/247)	72%	(70/247)	★○**78%**	(54/247)	★●**51%**	(121/247)	★●**49%**	(125/247)
WordNet	59%	(48/118)	★●**58%**	(50/118)	★●**44%**	(66/118)	★●**42%**	(69/118)	60%	(47/118)	★○**64%**	(43/118)	★●**44%**	(66/118)	★●**40%**	(71/118)
Minimum	38%		37%		33%		30%		58%		58%		30%		30%	
Mean	67%		66%		46%		43%		72%		74%		50%		48%	
Maximum	93%		93%		59%		60%		88%		89%		64%		64%	

The issue of test case diversity also helps to explain the third and fourth trends that we observe: STICCER-HGS is better, overall, at reducing AVM-D-generated test suites compared to STICCER-GRD — but conversely, STICCER-GRD is better, overall, at reducing DOMINO-generated test suites. We see both of these phenomena in the summary averages of Tables 2 and 3 — and also when comparing the respective number of schemas STICCER-HGS is significantly better at reducing compared to STICCER-GRD, and vice versa. In the AVM-D case, its choice of repetitious values hinders STICCER's merging, resulting in the ultimate winner being strongly correlated to the effectiveness of the original reduction technique used — that is, HGS in the case of STICCER-HGS, which is more effective than Greedy, used by STICCER-GRD. However, STICCER can work more effectively with the diverse test cases generated by DOMINO, and furthermore, it seems that the larger reduced test suites supplied by Greedy add to this diversity, allowing STICCER's merging to operate more effectively. Hence, STICCER-GRD performs significantly better than STICCER-HGS in more cases than it does not for DOMINO-generated test suites. In the cases that it does not, STICCER-HGS has the advantage of leveraging the more effective reduction provided by HGS. The "lift" of diversity that STICCER-GRD gets from less effective Greedy reduction can be seen for the AVM-D-generated test suites also, resulting in STICCER-HGS not being significantly better for every database schema.

The *BookTown* database schema provides a good illustration of both of these two trends. As shown in Table 2, the unreduced test suite has 269 test cases, which, in the AVM-D case are reduced to 144 and 167 test cases by HGS and Greedy respectively, and then further to 100 and 113 test cases following merging. STICCER can reduce the test suite by more test cases in its merging phase for STICCER-GRD (54, as opposed to 44 achieved by STICCER-HGS), but the initial advantage given to STICCER-HGS by virtue of using HGS for reduction prior to merging is not completely overturned. Conversely, in the DOMINO case, the original test suite size is reduced to 138 and 156 test cases by HGS and Greedy, respectively. However, because of the larger, more diverse pool of test cases produced by DOMINO, the STICCER-GRD technique overturns the initial advantage of HGS, reducing the test suite down to a final size of 87 test cases, as opposed to 94 for STICCER-HGS.

In conclusion for RQ1, like STICCER before it, STICCER-HGS significantly outperforms both HGS and Greedy. The results show that STICCER-HGS is more effective with test suites generated using AVM-D, while STICCER-GRD is more effective for test suites generated with DOMINO. In general, STICCER's merging is more effective with the diverse test data values in DOMINO-generated test cases, and works better with the slightly larger pool of test cases that Greedy tends to provide to the test merging mechanism.

Answer to RQ2: Fault Finding Capability. Table 4 shows the schemas that experienced significant differences in mutation score following test suite reduction. The data shows that test suites generated by DOMINO were immune to significant differences following reduction with any technique. Since RQ1 showed how diverse values were beneficial for reduction, this data would suggest they also protect test suites against the erosion of their mutation score, despite the test suites concerned having fewer test cases and fewer statements overall. AVM-D-generated test suites, without the benefit of the same extent of diversity, did suffer in decreases in mutation score after reduction. AVM-D-generated and STICCER-HGS-reduced test suites received significantly worse mutation scores for seven schemas (each accompanied by a large effect size) than the original test suite, although the differences were not greater than 4%.

Table 3: Median Statement Reduction Effectiveness

Refer to Section 3 for the calculation of the higher-is-better test case reduction effectiveness metric, and the meaning of the symbols ●, ○, and ★. The numbers in brackets indicate the median number of statements in the reduced test suite for a schema, followed by the median number of statements in its original test suite, respectively.

	AVM-D								DOMINO							
Schema	STICCER-HGS		STICCER-GRD		HGS		Greedy		STICCER-HGS		STICCER-GRD		HGS		Greedy	
ArtistSimilarity	52%	(21/44)	●**48%**	(23/44)	★●**45%**	(24/44)	★●**34%**	(29/44)	50%	(22/44)	50%	(22/44)	★●**35%**	(28/44)	★●**39%**	(27/44)
ArtistTerm	59%	(51/124)	●**56%**	(54/124)	★●**45%**	(68/124)	★●**36%**	(79/124)	54%	(56/124)	★○**56%**	(54/124)	★●**36%**	(79/124)	★●**36%**	(79/124)
BankAccount	59%	(33/80)	●**56%**	(35/80)	★●**44%**	(45/80)	★●**38%**	(50/80)	57%	(34/80)	57%	(34/80)	★●**50%**	(40/80)	★●**44%**	(44/80)
BookTown	52%	(194/403)	★●**44%**	(225/403)	★●**47%**	(215/403)	★●**38%**	(250/403)	51%	(197/403)	★●**49%**	(206/403)	★●**48%**	(208/403)	★●**42%**	(233/403)
BrowserCookies	65%	(61/175)	64%	(63/175)	★●**64%**	(63/175)	★●**62%**	(66/175)	69%	(54/175)	69%	(55/175)	★●**60%**	(70/175)	★●**59%**	(71/175)
Cloc	63%	(22/60)	●**62%**	(23/60)	★●**50%**	(30/60)	★●**48%**	(31/60)	58%	(25/60)	★○**61%**	(24/60)	★●**55%**	(27/60)	★●**48%**	(31/60)
CoffeeOrders	61%	(106/273)	●**61%**	(107/273)	★●**48%**	(143/273)	★●**45%**	(149/273)	64%	(98/273)	65%	(96/273)	★●**51%**	(132/273)	★●**49%**	(139/273)
CustomerOrder	43%	(273/475)	★●**39%**	(290/475)	★●**39%**	(288/475)	★●**36%**	(305/475)	59%	(196/475)	★●**56%**	(208/475)	★●**41%**	(279/475)	★●**36%**	(302/475)
DellStore	58%	(118/281)	●**56%**	(123/281)	★●**42%**	(162/281)	★●**39%**	(172/281)	48%	(145/281)	★○**50%**	(141/281)	★●**47%**	(150/281)	★●**44%**	(156/281)
Employee	58%	(22/53)	★○**60%**	(21/53)	★●**47%**	(28/53)	★●**47%**	(28/53)	62%	(20/53)	62%	(20/53)	★●**57%**	(23/53)	★●**57%**	(23/53)
Examination	66%	(77/229)	●**66%**	(78/229)	★●**52%**	(111/229)	★●**50%**	(114/229)	73%	(62/229)	★○**74%**	(60/229)	★●**64%**	(83/229)	★●**63%**	(86/229)
Flights	71%	(40/137)	★●**70%**	(41/137)	★●**66%**	(46/137)	★●**66%**	(46/137)	58%	(57/137)	58%	(58/137)	★●**55%**	(62/137)	★●**55%**	(62/137)
FrenchTowns	47%	(85/161)	★○**50%**	(81/161)	★●**43%**	(91/161)	★●**47%**	(86/161)	53%	(76/161)	52%	(77/161)	★●**43%**	(92/161)	★●**40%**	(96/161)
Inventory	57%	(12/28)	★●**54%**	(13/28)	★●**43%**	(16/28)	★●**39%**	(17/28)	61%	(11/28)	●**57%**	(12/28)	★●**54%**	(13/28)	★●**54%**	(13/28)
IsoFlav_R2	63%	(102/274)	★●**62%**	(104/274)	★●**50%**	(136/274)	★●**50%**	(136/274)	64%	(100/274)	64%	(100/274)	★●**59%**	(111/274)	★●**58%**	(116/274)
Iso3166	47%	(10/19)	47%	(10/19)	★●**37%**	(12/19)	★●**37%**	(12/19)	47%	(10/19)	●**42%**	(11/19)	47%	(10/19)	★●**37%**	(12/19)
iTrust	57%	(946/2204)	★●**56%**	(978/2204)	★●**50%**	(1103/2204)	★●**48%**	(1142/2204)	56%	(978/2204)	★○**57%**	(940/2204)	★●**52%**	(1064/2204)	★●**50%**	(1101/2204)
JWhoisServer	40%	(153/256)	★●**38%**	(158/256)	40%	(153/256)	★●**38%**	(158/256)	45%	(141/256)	★○**47%**	(136/256)	★●**43%**	(145/256)	★●**41%**	(152/256)
MozillaExtensions	63%	(130/356)	○**71%**	(105/356)	★●**50%**	(179/356)	63%	(130/356)	67%	(119/356)	★○**69%**	(110/356)	★●**61%**	(140/356)	★●**60%**	(144/356)
MozillaPermissions	62%	(19/50)	62%	(19/50)	★●**48%**	(26/50)	★●**48%**	(26/50)	62%	(19/50)	★○**66%**	(17/50)	★●**60%**	(20/50)	★●**54%**	(23/50)
NistDML181	68%	(25/78)	★●**68%**	(25/78)	★●**53%**	(37/78)	★●**55%**	(35/78)	68%	(25/78)	●**65%**	(28/78)	★●**56%**	(34/78)	★●**54%**	(36/78)
NistDML182	74%	(100/384)	★●**73%**	(102/384)	★●**61%**	(150/384)	★●**61%**	(151/384)	75%	(97/384)	74%	(98/384)	★●**63%**	(144/384)	★●**61%**	(149/384)
NistDML183	68%	(22/68)	●**65%**	(24/68)	★●**53%**	(32/68)	★●**46%**	(37/68)	65%	(24/68)	62%	(26/68)	★●**51%**	(33/68)	★●**50%**	(34/68)
NistWeather	60%	(48/120)	○**61%**	(47/120)	★●**49%**	(61/120)	★●**52%**	(58/120)	63%	(44/120)	63%	(44/120)	★●**52%**	(58/120)	★●**51%**	(58/120)
NistXTS748	48%	(12/23)	48%	(12/23)	★●**39%**	(14/23)	★●**35%**	(15/23)	57%	(10/23)	52%	(11/23)	★●**48%**	(12/23)	★●**48%**	(12/23)
NistXTS749	60%	(29/73)	★●**53%**	(34/73)	★●**53%**	(34/73)	★●**47%**	(39/73)	59%	(30/73)	58%	(30/73)	★●**49%**	(37/73)	★●**49%**	(37/73)
Person	53%	(14/30)	53%	(14/30)	★●**50%**	(15/30)	★●**50%**	(15/30)	43%	(17/30)	★○**47%**	(16/30)	43%	(17/30)	★●**37%**	(19/30)
Products	67%	(48/144)	●**65%**	(50/144)	★●**56%**	(63/144)	★●**53%**	(68/144)	63%	(53/144)	●**62%**	(54/144)	★●**53%**	(68/144)	★●**51%**	(70/144)
RiskIt	55%	(312/687)	★●**50%**	(342/687)	★●**51%**	(336/687)	★●**47%**	(366/687)	57%	(297/687)	●**54%**	(318/687)	★●**53%**	(322/687)	★●**50%**	(346/687)
StackOverflow	65%	(90/257)	●**64%**	(93/257)	★●**50%**	(129/257)	★●**48%**	(134/257)	62%	(98/257)	★○**66%**	(88/257)	★●**57%**	(110/257)	★●**57%**	(111/257)
StudentResidence	57%	(31/72)	●**56%**	(32/72)	★●**46%**	(39/72)	★●**42%**	(42/72)	60%	(29/72)	60%	(29/72)	★●**49%**	(37/72)	★●**49%**	(37/72)
UnixUsage	61%	(232/595)	★●**58%**	(252/595)	★●**51%**	(293/595)	★●**47%**	(313/595)	69%	(187/595)	★○**70%**	(178/595)	★●**54%**	(274/595)	★●**52%**	(287/595)
Usda	61%	(149/381)	●**59%**	(157/381)	★●**46%**	(206/381)	★●**42%**	(220/381)	55%	(171/381)	★○**59%**	(157/381)	★●**52%**	(181/381)	★●**50%**	(190/381)
WordNet	52%	(93/192)	★●**50%**	(96/192)	★●**49%**	(98/192)	★●**47%**	(102/192)	50%	(96/192)	49%	(97/192)	★●**49%**	(98/192)	★●**45%**	(106/192)
Minimum	40%		38%		37%		34%		43%		42%		35%		36%	
Mean	59%		57%		49%		46%		59%		59%		51%		49%	
Maximum	74%		73%		66%		66%		75%		74%		64%		63%	

In conclusion for RQ2, DOMINO-generated test suites did not change mutation score following reduction. AVM-D-generated suites did incur decreased scores, but only for seven schemas and not > 4%.

Answer to RQ3: Reduction and Mutation Analysis Runtime. Our prior paper [4] established that, on the whole, the time taken for STICCER-GRD to reduce test suites was more than regained in mutation analysis in the majority of cases, thereby decreasing the time needed to conduct mutation analysis overall. Table 5 shows a similar trend for STICCER-HGS, where savings of minutes to several minutes are possible, when comparing reduced test suites with the original test suite. However, although Table 5 reports many significant differences in times recorded for STICCER-HGS and STICCER-GRD, the vast majority only correspond to a couple of seconds, and therefore are almost practically negligible.

The exception to this is the *iTrust* schema, which has the largest original test suite of 1517 test cases. Here, the overheads of the additional algorithmic complexity of HGS compared to Greedy are evident. HGS took a median of 11 minutes to reduce the AVM-D-generated test suites for the *iTrust* schema, compared to only 2 minutes with Greedy reduction. As shown by Table 2, following merging this results in smaller AVM-D-generated test suites on average for STICCER-HGS compared to STICCER-GRD (631 test cases as opposed to 646), but the DOMINO-generated test suites are larger (297 as opposed to 231). Unsurprisingly, mutation analysis times follow the reduced test suite sizes, since the larger the test suite, the more work mutation analysis has to do. Overall, the additional time taken by HGS for the AVM-D-generated test suites is not sufficiently recovered in mutation analysis for the smaller suites of STICCER-HGS, resulting in STICCER-GRD recording a significantly faster time with AVM-D and DOMINO test suites.
In conclusion for RQ3, although our experiments record many significant differences in timing, they are almost negligible in practical terms, except for the largest schema, *iTrust*. For this schema, STICCER-HGS was significantly slower for both AVM-D and the DOMINO-generated test suites. In the AVM-D case, STICCER-HGS produces smaller test suites, but the additional time HGS needs to do this is not recovered in the savings made by mutation analysis.

Table 4: Median Mutation Scores

"S-HGS" is STICCER-HGS, "S-GRD" is STICCER-GRD, and "OTS" is Original Test Suite. For space reasons, we omit the results for schemas that have no significant differences with STICCER-HGS. Refer to Section 3 for the meaning of the symbols ●, ○, and ★.

	AVM-D					DOMINO				
Schemas	S-HGS	S-GRD	HGS	Greedy	OTS	S-HGS	S-GRD	HGS	Greedy	OTS
BrowserCookies	86.0	○**86.5**	86.2	○**86.5**	★○**86.5**	96.6	96.6	96.6	96.6	96.6
FrenchTowns	80.3	80.3	★○**81.1**	80.3	★○**83.3**	95.5	95.5	95.5	95.5	95.5
iTrust	83.6	●**83.6**	83.6	●**83.6**	★○**83.6**	99.1	99.2	99.1	99.2	99.2
NistWeather	93.8	★●**90.6**	93.8	★●**90.6**	93.8	100.0	100.0	100.0	100.0	100.0
NistXTS749	88.0	★○**92.0**	88.0	★○**92.0**	★○**92.0**	94.0	94.0	94.0	94.0	94.0
RiskIt	88.8	★○**89.3**	88.8	★○**89.3**	★○**89.3**	99.5	99.5	99.5	99.5	99.5
UnixUsage	97.3	★○**98.2**	97.3	★○**98.2**	★○**98.2**	100.0	100.0	100.0	100.0	100.0
WordNet	86.3	86.3	86.3	86.3	★○**87.4**	99.0	99.0	99.0	99.0	99.0

Overall Conclusions of the Computational Study. The evidence suggests that STICCER's merging mechanism works better with the diverse DOMINO-generated tests, and the slightly larger set of tests to choose from that arise from using Greedy. Yet, the results for each schema are more nuanced. For some schemas, the more heavily reduced test suites produced by HGS more than outweigh a slightly less efficient secondary merging phase for STICCER-HGS, particularly with those test suites generated by AVM-D.

The results of mutation analysis show a slight degradation of mutation scores for test suites initially generated by AVM-D for all reduction techniques, but no loss of mutant killing power for test suites generated by DOMINO. This evidence suggests that STICCER's merging mechanism does not sacrifice fault-finding capability.

In terms of execution time, we find that STICCER-HGS produced comparable timings to STICCER-GRD for reduction and the subsequent mutation analysis. Timings were marginally faster with

Table 5: Median Reduction and Mutation Times (Seconds)

Lower-is-better median time to reduce test suites (RT), assess the adequacy of the test suite by running mutation analysis, and the total of the two (Total). In this table "OTS" refers to the Original Test Suite. Refer to Section 3 for the meaning of the symbols ●, ○, and ★.

	AVM-D							Domino						
	STICCER-HGS			STICCER-GRD			OTS	STICCER-HGS			STICCER-GRD			OTS
Schemas	RT	MT	Total	RT	MT	Total	Total	RT	MT	Total	RT	MT	Total	Total
ArtistSimilarity	0.04	0.05	0.09	★●**0.12**	0.05	★●**0.17**	0.09	0.05	0.05	0.10	★●**0.13**	0.05	★●**0.18**	0.10
ArtistTerm	0.06	0.22	0.29	★●**0.14**	0.22	★●**0.37**	★●**0.54**	0.05	0.23	0.28	★●**0.14**	0.22	★●**0.36**	★●**0.54**
BankAccount	0.09	0.18	0.27	★●**0.17**	●**0.19**	★●**0.36**	★●**0.45**	0.08	0.20	0.28	★●**0.16**	○**0.19**	★●**0.34**	★●**0.46**
BookTown	0.90	14.20	15.10	0.94	★●**16.17**	★●**17.11**	★●**36.19**	1.04	11.46	12.50	★●**1.07**	★○**10.85**	★○**11.92**	★●**36.50**
BrowserCookies	0.34	0.92	1.26	0.37	0.94	1.31	★●**2.21**	0.38	0.72	1.09	0.37	0.73	1.09	★●**2.34**
Cloc	0.07	0.07	0.14	★●**0.15**	0.07	★●**0.22**	★●**0.27**	0.08	0.12	0.20	★●**0.15**	★○**0.09**	★●**0.25**	★●**0.28**
CoffeeOrders	0.25	1.21	1.46	★●**0.31**	1.20	★●**1.51**	★●**2.96**	0.22	1.14	1.35	★●**0.29**	★○**1.05**	1.34	★●**2.98**
CustomerOrder	0.70	6.22	6.92	★●**0.80**	★●**6.68**	★●**7.47**	★●**9.74**	0.76	4.35	5.10	★●**0.84**	★●**4.56**	★●**5.40**	★●**9.78**
DellStore	1.68	1.86	3.54	★○**1.42**	●**1.93**	★○**3.35**	★●**8.04**	1.75	3.37	5.12	★○**1.46**	★○**3.22**	★○**4.68**	★●**8.40**
Employee	0.15	0.14	0.29	★●**0.20**	★○**0.13**	★●**0.32**	★●**0.34**	0.14	0.11	0.25	★●**0.18**	0.12	★●**0.30**	★●**0.36**
Examination	1.72	1.37	3.09	★○**1.06**	1.39	★○**2.45**	★●**4.42**	1.73	1.06	2.79	★○**1.11**	○**1.00**	★○**2.12**	★●**4.48**
Flights	0.20	0.61	0.81	●**0.30**	0.61	●**0.91**	★●**1.60**	0.37	0.71	1.08	★●**0.41**	○**0.68**	1.09	★●**1.68**
FrenchTowns	0.12	1.51	1.62	★●**0.23**	○**1.46**	1.69	★●**2.43**	0.14	1.23	1.37	★●**0.23**	1.23	★●**1.46**	★●**2.44**
Inventory	0.04	0.04	0.08	★●**0.12**	★●**0.05**	★●**0.17**	★●**0.10**	0.05	0.04	0.09	★●**0.12**	0.04	★●**0.17**	●**0.10**
IsoFlav_R2	0.88	2.82	3.70	★○**0.70**	2.83	★○**3.52**	★●**9.80**	0.96	2.79	3.75	★○**0.75**	2.72	★○**3.47**	★●**10.20**
Iso3166	0.03	0.02	0.06	★●**0.11**	0.02	★●**0.13**	★○**0.04**	0.04	0.03	0.06	★●**0.11**	0.03	★●**0.13**	★○**0.05**
iTrust	653.16	936.61	1589.77	★○**150.17**	★●**959.31**	★○**1109.48**	★●**2297.86**	634.77	522.86	1157.63	★○**157.23**	★○**428.57**	★○**585.80**	★●**2330.02**
JWhoisServer	1.90	5.39	7.29	★○**1.64**	★●**5.62**	7.25	★●**8.91**	2.11	4.28	6.39	★○**1.86**	★○**3.88**	★○**5.74**	★●**9.34**
MozillaExtensions	5.16	6.95	12.12	★○**2.02**	★○**6.62**	★○**8.64**	★●**25.52**	5.26	6.36	11.62	★○**2.38**	★○**5.39**	★○**7.78**	★●**26.61**
MozillaPermissions	0.09	0.10	0.18	★●**0.15**	★○**0.08**	★●**0.23**	★●**0.23**	0.08	0.10	0.18	★●**0.15**	★○**0.07**	★●**0.22**	★●**0.24**
NistDML181	0.07	0.10	0.18	★●**0.15**	★●**0.11**	★●**0.26**	★●**0.36**	0.08	0.12	0.20	★●**0.15**	★●**0.12**	★●**0.28**	★●**0.38**
NistDML182	3.64	2.43	6.07	★○**1.93**	★○**2.22**	★○**4.15**	★●**14.60**	3.66	2.59	6.25	★○**2.00**	★○**2.43**	★○**4.43**	★●**14.99**
NistDML183	0.06	0.09	0.15	★●**0.14**	●**0.09**	★●**0.23**	★●**0.28**	0.07	0.10	0.17	★●**0.14**	0.10	★●**0.25**	★●**0.29**
NistWeather	0.15	0.30	0.45	★●**0.22**	★○**0.29**	●**0.51**	★●**0.74**	0.18	0.26	0.44	★●**0.23**	★○**0.25**	●**0.48**	★●**0.76**
NistXTS748	0.04	0.04	0.08	★●**0.12**	★○**0.04**	★●**0.16**	○**0.08**	0.04	0.04	0.08	★●**0.11**	0.04	★●**0.15**	0.08
NistXTS749	0.10	0.18	0.28	★●**0.17**	★●**0.20**	★●**0.37**	★●**0.39**	0.08	0.17	0.25	★●**0.17**	0.16	★●**0.33**	★●**0.40**
Person	0.03	0.09	0.12	★●**0.10**	0.09	★●**0.19**	●**0.16**	0.08	0.08	0.16	★●**0.15**	★○**0.07**	★●**0.22**	0.16
Products	0.08	0.42	0.50	0.14	0.42	●**0.57**	★●**1.04**	0.12	0.44	0.56	★●**0.18**	0.44	★●**0.62**	★●**1.06**
RiskIt	1.85	21.23	23.08	★○**1.41**	★●**23.15**	★●**24.57**	★●**44.40**	1.93	18.22	20.15	★○**1.59**	★●**18.90**	●**20.49**	★●**45.91**
StackOverflow	1.34	0.93	2.27	★○**0.93**	★●**0.97**	★○**1.90**	★●**5.58**	1.51	1.74	3.25	★○**1.16**	★○**1.42**	★○**2.58**	★●**5.76**
StudentResidence	0.09	0.20	0.29	★●**0.16**	0.20	★●**0.36**	★●**0.43**	0.07	0.17	0.24	★●**0.14**	0.17	★●**0.32**	★●**0.44**
UnixUsage	1.04	5.69	6.73	★○**0.92**	★●**6.21**	★●**7.13**	★●**12.25**	1.07	4.15	5.22	★○**0.95**	★○**3.83**	★○**4.79**	★●**12.34**
Usda	1.88	2.71	4.59	★○**1.39**	★●**2.78**	★○**4.17**	★●**15.55**	2.02	5.31	7.33	★○**1.52**	★○**4.15**	★○**5.68**	★●**16.44**
WordNet	0.34	1.77	2.11	●**0.37**	★●**1.86**	★●**2.23**	★●**3.89**	0.29	1.77	2.07	★●**0.38**	★○**1.69**	2.07	★●**3.93**

STICCER-HGS for smaller database schemas, yet STICCER-GRD had the upper hand with the largest schemas, because of the additional time required by HGS to reduce suites in the first phase.

4 HUMAN STUDY

To investigate the effect of STICCER's test case merging mechanism on human oracle cost, we designed a Human Study in which participants acted as "testers" who had to manually inspect test suites that had been processed by STICCER. As a control, we chose the (unmerged) test suites reduced by HGS, as they, in general, represent the smallest non-merged test suites, thereby making them suitable for the scope of a human study. As such, to allow for a direct comparison, we chose to use STICCER-HGS over STICCER-GRD to study the effect of test merging. A relational database test case attempts to satisfy or violate an integrity constraint with `INSERT` statements that are either accepted or rejected by the DBMS. In our study, therefore, participants had to read a test case and identify the `INSERT` statement(s) that would be rejected. We measured their accuracy and efficiency (i.e., time duration) while they performed this task, with the aim of answering two research questions:

RQ4: Test Inspection Accuracy. How accurate are humans at inspecting the merged and reduced tests produced with STICCER-HGS compared to the reduced and non-merged tests made by HGS?

RQ5: Test Inspection Duration. How long does it take for humans to inspect the merged and reduced tests produced by STICCER-HGS compared to the reduced and non-merged tests made by HGS?

We generated test suites using AVM-D and Domino for the schemas *ArtistSimilarity*, *Inventory*, *NistXTS748*, and *Person*, as listed in Table 1, and applied both HGS and STICCER-HGS. We deliberately picked these schemas to ensure all different types of integrity constraint were represented and a variety of data types, while also ensuring relatively small test suite sizes (i.e., under 30 test cases) so that the test suites used would be feasible for a human to inspect during the study in a reasonable amount of time.

We used *SchemaAnalyst* to generate test suites using the Clause-AICC+ANCC+AUCC coverage criterion combination with the *mutated* versions of each schema. In the study, we asked participants to assess these test suites with respect to the original schemas. We used mutants rather than original schemas for test suite generation to introduce a degree of randomness in the accept/violate pattern of the `INSERT` statements of each suite, enabling a fairer comparison between their merged and reduced versions. We randomly selected the mutant schemas summarized in Table 6 from a pool of mutants generated using the operators of Wright et al. [56].

To measure the accuracy and duration of human inspection, we integrated both the original schema and the mutant's tests into a web questionnaire. Each test case forms an individual "question", where participants are asked to select the `INSERT` statements in each test that the DBMS would reject. If the participant believed that none of the `INSERT`s should be rejected, they could select an option entitled "None of them". If a participant could not decide, then they could select the "I don't know" option. Our thinking behind both options was to prevent random guessing that could negatively influence the results. Furthermore, to prevent confounding results, we also added a mechanism that deselects checkboxes if an option was selected that would contradict another option. For instance, if a participant selected a series of `INSERT`s and then continued to pick either "I don't know" or "None of them" (i.e., they seemingly changed their mind), then the `INSERT`s are deselected, or vice versa.

At the end of questionnaire, participants were presented with an online exit survey that asked about the schemas that they thought to be the easiest and hardest to inspect. The participants could also provide general feedback regarding the questionnaire, ultimately

Table 6: The Mutated Schemas Used in the Human Study

Schema	AVM-D	Domino
ArtistSimilarity	Added a `NOT NULL` to a new column	Added a `NOT NULL` to a new column
Inventory	Changed the column of a `UNIQUE`	Changed the column of a `UNIQUE`
NistXTS748	Added a new `UNIQUE` to a new column	Added a single-column primary key
Person	Removed primary key	Changed primary key to another column

helping us to analyze the results and further characterize a human's perception of the database schemas and their reduced test suites.

In total, we recruited 27 participants from the student body at the University of Sheffield, studying Computer Science (or a related degree) at either the undergraduate or PhD level. We selected participants using a process based on a quiz where individuals had to say whether four `INSERT`s would be accepted or rejected for a table with three constraints. We excluded potential participants if they got more than one wrong answer, thus ensuring that the study only involved people who were likely to be capable of assessing the tests. The participants' level of SQL experience — information that we collected as part of the questionnaire — varied between less than a year for four people to over four years for eight. We financially compensated participants with £5 cash and a £10 book voucher.

The study had two within-subject variables (i.e., the database schemas and the generation techniques) and one between-subject variable (i.e., the specific reduced test suites). We assigned participants randomly to one of four groups, so that there were at least six participants in each group. Each group inspected each schema with each test suite, reduced by either HGS or STICCER-HGS.

To answer **RQ4**, we calculated participants' test inspection accuracy scores based on the number of failing `INSERT` statements correctly selected over all the `INSERT`s (i.e., those that the DBMS accepted or rejected). We report the accuracy score's descriptive statistics (i.e., minimums, maximums, means, and medians).

To answer **RQ5**, we reported the same descriptive statistics for the duration of time that a human took to inspect each test suite.

Unfortunately, due to the small sample of participants and database schemas, we cannot reliably apply statistical significance tests. We leave this as an item for future work, as explained in Section 6.

Threats to Validity. The external validity of the selected schemas and its generated tests may provide results that are not evident for real schemas. We tried to mitigate this by randomly selected four schemas that include common integrity constraints and data types in SQL schemas [46]. We also used an open-source tool to generate the tests, *SchemaAnalyst* [39], with the most effective combination of adequacy criteria [42], thereby ensuring all integrity constraints were thoroughly tested by the generated test suites.

We intentionally selected a relatively small number of small schemas to ensure participants could complete the questionnaire in a reasonable time and to avoid potential fatigue effects affecting our results. This did, however, lead to a small sample size that was insufficient for statistical hypothesis testing, causing us to fall back on descriptive statistics. In the future, we recommend replicating this study with more data points to increase the statistical power.

Another validity threat is that of learning effects, whereby participants become better at answering questions as the questionnaire progresses. We mitigated this concern by randomizing the presentation order for the questions and the database schemas.

Finally, the majority of this study's participants are students. While this can be considered another threat, this approach has been deemed as acceptable and in broad alignment with prior experiments in software engineering by other researchers [22].

Table 7: Descriptive Statistics of Scores and Durations
The accuracy percentage scores and the duration of time in seconds the humans needed to inspect the reduced test suites. Hence, the higher percentage score and the lower duration is better. The variable P denotes the number of participants in each group, T and I are the test suite size and number of `INSERT`s in the test suite, respectively. Finally, S-HGS denotes STICCER-HGS.

						Score (%)				Duration (Minutes)			
Schema	Generator	Reduction	P	T	I	Min	Mean	Median	Max	Min	Mean	Median	Max
ArtistSimilarity	AVM-D	HGS	7	10	22	70.0	87.4	90.0	100.0	1.6	4.0	3.2	6.5
		S-HGS	6	8	20	79.0	88.0	91.0	91.0	2.6	4.9	4.0	8.0
	Domino	HGS	7	10	22	80.0	91.0	90.0	100.0	3.7	5.7	4.5	10.3
		S-HGS	7	8	20	37.0	79.5	87.0	100.0	1.9	4.1	3.6	6.5
Inventory	AVM-D	HGS	7	10	16	70.0	88.6	85.0	100.0	2.7	5.7	5.4	9.6
		S-HGS	7	5	12	40.0	73.4	72.0	93.0	2.1	3.2	2.7	5.7
	Domino	HGS	7	7	12	78.0	94.6	100.0	100.0	1.7	3.4	3.5	5.8
		S-HGS	6	6	12	25.0	83.3	100.0	100.0	1.6	2.6	2.1	5.1
NistXTS748	AVM-D	HGS	6	9	14	66.0	86.7	88.5	100.0	1.1	3.0	3.3	4.6
		S-HGS	7	5	11	58.0	86.9	95.0	100.0	1.8	4.1	4.6	6.0
	Domino	HGS	7	8	12	37.0	83.7	87.0	100.0	1.5	2.4	1.9	4.7
		S-HGS	7	6	12	75.0	84.1	83.0	91.0	2.2	4.6	3.7	9.3
Person	AVM-D	HGS	7	11	13	31.0	85.9	90.0	100.0	1.8	4.7	3.5	10.6
		S-HGS	7	3	12	88.0	95.7	100.0	100.0	1.9	4.1	2.7	9.1
	Domino	HGS	6	14	19	53.0	90.8	100.0	100.0	3.8	4.9	5.0	6.3
		S-HGS	7	6	18	33.0	88.3	97.0	100.0	2.8	4.3	4.0	6.5
All Schemas	AVM-D	HGS	27	40	55	31.0	87.1	90.0	100.0	1.2	4.4	4.2	10.6
		S-HGS	27	21	55	40.0	85.9	91.0	100.0	1.8	4.1	3.3	9.1
	Domino	HGS	27	39	65	37.0	90.0	95.0	100.0	1.5	4.1	4.0	10.3
		S-HGS	27	26	62	25.0	83.8	91.0	100.0	1.6	3.9	3.5	9.3

Measuring a human's understanding of tests is subjective and a threat to the construct validity that we addressed by determining how successful human testers were at identifying which `INSERT`s are rejected by the database for violating an integrity constraint.

Another threat is that the participants might not be accustomed to the questionnaire interface's to determine the outcome of a schema test case. We addressed this concern by providing a tutorial prior to the completion of the actual questionnaire, showing concepts about testing integrity constraints and the study's procedure.

It is also possible that testers might have better knowledge of a database schema that they designed than the participants in the Human Study. To address this concern, we allowed participants the opportunity to study each schema to properly understand it before having to answer the questions about the schema's test suite.

Answer to RQ4: Test Inspection Accuracy. Table 7 shows the descriptive statistics for the accuracy scores of and time duration by the participants for each test suite that they evaluated. On average, participants were more accurate with the test suites reduced by HGS compared to STICCER-HGS. The mean difference in accuracy, however, for test suites was only as large as 15.2% (for the *Inventory* schema with the test suite generated by AVM-D), with the largest median difference as 13.0% (again for *Inventory* with the test suite generated by AVM-D). Overall, no clear pattern emerges, and it would seem that the smaller test suites that were reduced and merged by STICCER-HGS do not give it an advantage over HGS. This suggests that testers prefer smaller, focused test cases as much, if not more than, fewer but potentially more complex test cases.

In conclusion for RQ4, the smaller test suites reduced and merged by STICCER-HGS give it no clear advantage over test suites reduced by HGS only, suggesting that fewer, but longer, tests do not necessarily improve the accuracy of humans when they inspect test cases.

Answer to RQ5: Test Inspection Duration. Table 7 shows the duration descriptive statistics of each test suite inspection. For 10 of the 16 schema-test generator combinations, the participants

were faster with test suites reduced and merged with STICCER-HGS, as opposed to simply being reduced with HGS. This table also shows that that the overall mean and median averages favor STICCER-HGS. These results suggest that participants can process the smaller number of test cases offered by STICCER-HGS more quickly, on the whole, even if they cannot do it more accurately. Given that STICCER-HGS test cases are longer, due to the merging, it would seem that there is more opportunity for participants to make mistakes, and/or become over-confident in their analysis.

In conclusion for RQ5, the evidence suggests that, compared to durations with tests from HGS, participants were faster at inspecting the smaller test suites reduced and merged by STICCER-HGS.

Overall Conclusions of the Human Study. The results from this study suggest that, while human testers are not more accurate at analyzing a smaller number of longer tests, there is some evidence that they are faster. One explanation for this is that a tester may subconsciously spend the same amount of time on a test, regardless of its length, therefore being faster overall with smaller test suites. Yet, this constant amount of time is a disadvantage for comparatively longer tests, as there is more to inspect, and as such aspects of these test cases may be overlooked, leading to mistakes. Although interesting, these results suggest the need for a large-scale study.

5 RELATED WORK

Test suite reduction methods aim to make regression testing more cost-effective [26]. As noted by Yoo and Harman [59], many reduction methods (e.g., additional greedy and HGS [9, 11, 20]) adopt some form of a greedy heuristic. Prior experimental studies found that, by removing redundant tests, both HGS and Greedy decreased the size [10, 60] and execution cost [23, 35, 36] of the test suite. Other work has shown the benefits of using, for instance, integer linear programming [8, 21] and evolutionary algorithms [38, 58] to reduce a test suite. Notably, unlike this paper's focus on reducing tests for database schemas, all the aforementioned work considered test suite reduction for traditional programs.

Although efficient regression testing is important [26], the cost for humans to evaluate test outcomes is also a critical consideration. For instance, prior studies found that, while manually written tests are hard to understand [33], they often are more readable than automatically generated ones [16]. Others characterized the difficulty that testers face when understanding and maintaining automated tests [14, 47]. Given these results, recent methods seek to minimize tester effort through test documentation [32, 33, 37], readability improvement [1, 14, 15], and test visualization [12, 48]. Yet, unlike this paper, none of the aforementioned approaches focused on the costs that humans incur when inspecting database schema tests.

This paper's hybrid test suite reduction method, called STICCER-HGS, was inspired by STICCER, our recent hybrid test suite reduction method for database schemas [4]. Moreover, while our previous study involving humans experimentally determined which type of automatically generated test data best supported testers [3], this prior paper did not, unlike the current one, involve humans in the study of reduced test suites. Finally, there are several prior methods for the regression testing of database applications, including a greedy approach for test suite reduction [27]. Other papers presented greedy methods for reducing the database application test suites comprised of SQL SELECT queries [54]. Notably, neither of the two aforementioned papers employed human testers to study the benefits of the presented test suite reduction techniques.

6 CONCLUSIONS AND FUTURE WORK

Since many software applications interact with a database that has a difficult-to-test schema, testers may use automated test data generation techniques, like DOMINO and AVM-D, to create a schema test suite. Although these generators obviate the need for manual testing, the test suites that they produce often have many tests with numerous, and sometimes similar, database interactions, suggesting the need for test suite reduction. Since our prior work proposed STICCER, a hybrid method that combined Greedy test suite reduction with a merging approach for database schema testing [4], this paper presents both a computational and a human study investigating a new hybridization that combines STICCER-based merging with test suite reduction by the Harrold-Gupta-Soffa method.

Considering four test suite reduction methods (i.e., Greedy, HGS, STICCER-GRD, and STICCER-HGS), two test data generators (i.e., AVM-D and DOMINO), and 34 database schemas, this paper's Computational Study answered three research questions. Focused on assessing the capability of these reduction methods to quickly decrease a test suite's size while preserving its mutation adequacy, the Computational Study reveals that, while there are benefits to using either Greedy or HGS in combination with STICCER, neither STICCER-GRD nor STICCER-HGS are a strictly dominant method. That is, although there was prior evidence showing that HGS was superior to Greedy at reducing database schema test suites, the surprising conclusion of this study is that there is no significant benefit to hybridizing STICCER with HGS instead of Greedy.

Incorporating 27 participants who had to manually inspect reduced test suites and answer questions about their behavior, the Human Study investigated the influence that STICCER's test case merging mechanism has on human oracle costs. Since this paper's focus is on the benefits attributable to HGS, this study compared HGS to STICCER-HGS, answering two research questions. This paper's Human Study reveals that, compared to those produced by HGS, the reduced test suites of STICCER-HGS may help humans to perform test inspection faster, but not always more accurately.

Along with confirming the benefits from hybridizing STICCER with either Greedy or HGS, this paper's two studies suggest that, while test suite reduction may make some schema testing tasks (e.g., test adequacy assessment with mutation analysis) more efficient, it may not always benefit the humans testers who inspect the reduced test suites. Given these results, we plan to investigate new STICCER hybridizations that leverage alternative test suite reduction methods (e.g., [34, 49, 52, 60]) that consider, for instance, both the requirement coverage and execution time of a test. We will also conduct new experiments with both additional database schemas and more human subjects, including testers from industry who have experience with relational databases. To ensure that any follow-on studies involving human testers yield results amenable to statistical analysis, we also intend to incorporate a greater variety of schema mutations. Building on the insights from this paper's studies, our ultimate goal is to develop fast reduction methods for database schema test suites that decrease suite size and runtime while both maintaining test adequacy and supporting humans testers.

REFERENCES

[1] Sheeva Afshan, Phil McMinn, and Mark Stevenson. 2013. Evolving Readable String Test Inputs Using a Natural Language Model to Reduce Human Oracle Cost. In *Proc. of ICST*.

[2] Abdullah Alsharif, Gregory M. Kapfhammer, and Phil McMinn. 2018. DOMINO: Fast and Effective Test Data Generation for Relational Database Schemas. In *Proc. of ICST*.

[3] Abdullah Alsharif, Gregory M. Kapfhammer, and Phil McMinn. 2019. What Factors Make SQL Test Cases Understandable for Testers? A Human Study of Automated Test Data Generation Techniques. In *Proc. of ICSME*.

[4] Abdullah Alsharif, Gregory M. Kapfhammer, and Phil McMinn. 2020. STICCER: Fast and Effective Database Test Suite Reduction Through Merging of Similar Test Cases. In *Proc. of ICST*.

[5] Scott Ambler. 2019. Database Testing: How to Regression Test a Relational Database. http://www.agiledata.org/essays/databaseTesting.html.

[6] Scott Ambler and Pramod J. Sadalage. 2006. *Refactoring Databases: Evolutionary Database Design*.

[7] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *Trans. on Software Engineering* 41, 5 (2015).

[8] Jennifer Black, Emanuel Melachrinoudis, and David Kaeli. 2004. Bi-Criteria Models for All-Uses Test Suite Reduction. In *Proc. of ICSE*.

[9] Tsong Yueh Chen and Man Fai Lau. 1998. A New Heuristic for Test Suite Reduction. *Information and Software Technology* (1998).

[10] Tsong Yueh Chen and Man Fai Lau. 1998. A Simulation Study on Some Heuristics for Test Suite Reduction. *Information and Software Technology* (1998).

[11] Vasek Chvatal. 1979. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research* (1979).

[12] Bas Cornelissen, Arie Van Deursen, Leon Moonen, and Andy Zaidman. 2007. Visualizing Testsuites to Aid in Software Understanding. In *Proc. of CSMR*.

[13] Carlo A. Curino, Letizia Tanca, Hyun J. Moon, and Carlo Zaniolo. 2008. Schema Evolution in Wikipedia: Toward a Web Information System Benchmark. In *Proc. of ICEIS*.

[14] Ermira Daka, José Campos, Gordon Fraser, Jonathan Dorn, and Westley Weimer. 2015. Modeling Readability to Improve Unit Tests. In *Proc. of FSE*.

[15] Ermira Daka, José Miguel Rojas, and Gordon Fraser. 2017. Generating Unit Tests with Descriptive Names or: Would You Name Your Children Thing1 and Thing2?. In *Proc. of ISSTA*.

[16] Giovanni Grano, Simone Scalabrino, Harald C Gall, and Rocco Oliveto. 2018. An Empirical Investigation on the Readability of Manual and Generated Test Cases. In *Proc. of ICPC*.

[17] Szymon Guz. 2011. Basic Mistakes in Database Testing. http://java.dzone.com/articles/basic-mistakes-database.

[18] Mark Harman, Sung Gon Kim, Kiran Lakhotia, Phil McMinn, and Shin Yoo. 2010. Optimizing for the Number of Tests Generated in Search Based Test Data Generation with an Application to the Oracle Cost Problem. In *Proc. of SBST*.

[19] M. Harman and P. McMinn. 2010. A Theoretical and Empirical Study of Search-Based Testing: Local, Global and Hybrid Search. *Trans. of Software Engineering* 36, 2 (2010).

[20] Mary Jean Harrold, Rajiv Gupta, and Mary Lou Soffa. 1993. A Methodology for Controlling the Size of a Test Suite. *Trans. on Software Engineering and Methodology* 2, 3 (1993).

[21] J Hartmann and DJ Robson. 1989. Revalidation During the Software Maintenance Phase. In *Proc. of ICSME*.

[22] Martin Höst, Björn Regnell, and Claes Wohlin. 2000. Using Students as Subjects: A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering* 5, 3 (2000).

[23] Dennis Jeffrey and Neelam Gupta. 2007. Improving Fault Detection Capability by Selectively Retaining Test Cases During Test Suite Reduction. *Trans. on Software Engineering* (2007).

[24] James A. Jones and Mary Jean Harrold. 2003. Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage. *Trans. on Software Engineering* 29, 3 (2003).

[25] Gregory M. Kapfhammer. 2007. *A Comprehensive Framework for Testing Database-Centric Applications.* Ph.D. Dissertation. University of Pittsburgh.

[26] Gregory M. Kapfhammer. 2010. Regression Testing. In *The Encyclopedia of Software Engineering*.

[27] Gregory M. Kapfhammer. 2012. Towards a Method for Reducing the Test Suites of Database Applications. In *Comp. of ICST*.

[28] Gregory M. Kapfhammer, Phil McMinn, and Chris J. Wright. 2013. Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems. In *Proc. of ICST*.

[29] Joseph Kempka, Phil McMinn, and Dirk Sudholt. 2013. A Theoretical Runtime and Empirical Analysis of Different Alternating Variable Searches for Search-Based Testing. In *Proc. of GECCO*.

[30] Joseph Kempka, Phil McMinn, and Dirk Sudholt. 2015. Design and Analysis of Different Alternating Variable Searches for Search-Based Software Testing. *Theoretical Computer Science* (2015).

[31] B. Korel. 1990. Automated Software Test Data Generation. *Trans. on Software Engineering* 16, 8 (1990).

[32] Boyang Li, Christopher Vendome, Mario Linares-Vásquez, and Denys Poshyvanyk. 2018. Aiding Comprehension of Unit Test Cases and Test Suites with Stereotype-Based Tagging. In *Proc. of ICPC*.

[33] Nan Li, Yu Lei, Haider Riaz Khan, Jingshu Liu, and Yun Guo. 2016. Applying Combinatorial Test Data Generation to Big Data Applications. In *Proc. of ASE*.

[34] Chu-Ti Lin, Kai-Wei Tang, Cheng-Ding Chen, and Gregory M. Kapfhammer. 2012. Reducing the cost of regression testing by identifying irreplaceable test cases. In *Proc. of ICGEC*.

[35] Chu-Ti Lin, Kai-Wei Tang, and Gregory M. Kapfhammer. 2014. Test Suite Reduction Methods that Decrease Regression Testing Costs by Identifying Irreplaceable Tests. *Information and Software Technology* 56, 10 (2014).

[36] Chu-Ti Lin, Kai-Wei Tang, Jiun-Shiang Wang, and Gregory M. Kapfhammer. 2017. Empirically Evaluating Greedy-Based Test Suite Reduction Methods at Different Levels of Test Suite Complexity. *Science of Computer Programming* 150 (2017).

[37] Mario Linares-Vásquez, Boyang Li, Christopher Vendome, and Denys Poshyvanyk. 2016. Documenting Database Usages and Schema Constraints in Database-Centric Applications. In *Proc. of ISSTA*.

[38] Nashat Mansour and Khalid El-Fakih. 1999. Simulated Annealing and Genetic Algorithms for Optimal Regression Testing. *Journ. of Software Maintenance: Research and Practice* 11, 1 (1999).

[39] Phil McMinn and Gregory M. Kapfhammer. 2016. AVMf: An Open-Source Framework and Implementation of the Alternating Variable Method. In *Proc. of SSBSE*.

[40] Phil McMinn, Gregory M. Kapfhammer, and Chris J. Wright. 2016. Virtual Mutation Analysis of Relational Database Schemas. In *Proc. of AST*.

[41] Phil McMinn, Mark Stevenson, and Mark Harman. 2010. Reducing Qualitative Human Oracle Costs associated with Automatically Generated Test Data. In *Proc. of STOV*.

[42] Phil McMinn, Chris J. Wright, and Gregory M. Kapfhammer. 2015. The Effectiveness of Test Coverage Criteria for Relational Database Schema Integrity Constraints. *Trans. on Software Engineering and Methodology* 25, 1 (2015).

[43] Phil McMinn, Chris J. Wright, Cody Kinneer, Colton J. McCurdy, Michael Camara, and Gregory M. Kapfhammer. 2016. SchemaAnalyst: Search-based Test Data Generation for Relational Database Schemas. In *Proc. of ICMSE*.

[44] Phil McMinn, Chris J. Wright, Colton J. McCurdy, and Gregory M. Kapfhammer. 2019. Automatic Detection and Removal of Ineffective Mutants for the Mutation Analysis of Relational Database Schemas. *Trans. on Software Engineering* 45, 5 (2019).

[45] PostgreSQL Project. 2012. About PostgreSQL. http://www.postgresql.org/about/.

[46] Dong Qiu, Bixin Li, and Zhendong Su. 2013. An Empirical Analysis of the Co-evolution of Schema and Code in Database Applications. In *Proc. of FSE*.

[47] José Miguel Rojas, Gordon Fraser, and Andrea Arcuri. 2015. Automated Unit Test Generation During Software Development: A Controlled Experiment and Think-Aloud Observations. In *Proc. of ISSTA*.

[48] Adam M. Smith, Joshua J. Geiger, Gregory M. Kapfhammer, Manos Renieris, and G. Elisabeta Marai. 2009. Interactive Coverage Effectiveness Multiplots for Evaluating Prioritized Regression Test Suites. In *Comp. of InfoVis*.

[49] Adam M. Smith and Gregory M. Kapfhammer. 2009. An Empirical Study of Incorporating Cost into Test Suite Reduction and Prioritization. In *Proc. of SAC*.

[50] SQLite Developers. 2012. Most Widely Deployed SQL Database Engine. http://www.sqlite.org/mostdeployed.html.

[51] SQLite Developers. 2019. SQL as Understood by SQLite. https://sqlite.org/lang_createtable.html.

[52] Sriraman Tallam and Neelam Gupta. 2005. A Concept Analysis Inspired Greedy Algorithm for Test Suite Minimization. In *Proceedings of the 6th Workshop on Program Analysis for Software Tools and Engineering*.

[53] Nigel Tracey, John Clark, Keith Mander, and John McDermid. 1998. An Automated Framework for Structural Test-Data Generation. In *Proc. of ASE*.

[54] Javier Tuya, Claudio de la Riva, Maria Jose Suarez-Cabal, and Raquel Blanco. 2016. Coverage-Aware Test Database Reduction. *Trans. on Software Engineering* 42, 10 (2016).

[55] András Vargha and Harold D. Delaney. 2000. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journ. of Educational and Behavioral Statistics* 25, 2 (2000).

[56] Chris J. Wright, Gregory M. Kapfhammer, and Phil McMinn. 2013. Efficient Mutation Analysis of Relational Database Structure Using Mutant Schemata and Parallelisation. In *Proc. of Mutation*.

[57] Chris J Wright, Gregory M Kapfhammer, and Phil McMinn. 2014. The Impact of Equivalent, Redundant and Quasi Mutants on Database Schema Mutation Analysis. In *Proc. of QSIC*.

[58] Shin Yoo and Mark Harman. 2010. Using Hybrid Algorithm for Pareto Efficient Multi-Objective Test Suite Minimisation. *Journ. of Systems and Software* (2010).

[59] Shin Yoo and Mark Harman. 2012. Regression Testing Minimization, Selection and Prioritization: A Survey. *Software Testing, Verification and Reliability* 22, 2 (2012).

[60] Hao Zhong, Lu Zhang, and Hong Mei. 2006. An Experimental Comparison of Four Test Suite Reduction Techniques. In *Proc. of ICSE*.