Driected Random

Abdullah Alsharif

3 January 2017

Directed Random Mechanism

Directed Random works as same as random technique however it is guided by predicates that are checked then fixing a solution. For instance, each INSERT statement must comply to a test requirement that have one or many predicates such as NOT NULL for a specific column, first directed random generate random insert statement then proceed to fixing the insert statement based on a given predicate and if it does not comply to the predicate. If a column is NULL but the predicate require a NOT NULL for this specific column, directed random will fix the insert statement to have NOT NULL. However, Directed random usually fixes one predicate at a time, so if there is many violated predicates in one insert statement it will only fix one then iterate to the next evaluation to fix the other remaining predicates. This means that each evaluation does not fix all predicates or search for optimal solution, however each evaluation checks if the statement is complying with the test requirement.

Predicate Checker

Predicate Fixer

Directed Random Algorithm

```
p <= predicate
n <= insert statement
CHECK method:
  IF n Comply with p THEN
    return true
  ELSE
    return false
END METHOD

FIX method:
    GET non-Complied predicate
    GET c <=column for non-complied predicate
    REPEAT:
      generate random value for column
    UNTIL vaule comply with predicate
END METHOD

generate random values for table insert n
result <= CHECK n aganist p

while result == Ture
  FIX n aganist p
  result <= CHECK n aganist p
END WHILE
```

In our experiments we ran two test data generators AVM and Directed Random (DR), from our experiment we are looking at the performance of the two techniques in regard of test generation timing and mutation

score. This will help us to determine which of the techniques are better in those both factors. Looking at test generation timing will determine which of the two are faster in generating test cases. On the other hand we will look at mutation analysis of the two techniques to determine the strength and the capability of the test suite generated to detect faults.

Experiment Set Up

Our experiment set-up was to run each technique 30 times for each case study using one combined coverage criteria "ClauseAICC+AUCC+ANCC". Each run has different random seed to see the difference of results.

Case Studies

The following table has the case studies that are been used in our experiment:

Schema	Tables	Columns	Total.Columns	Total.Constraints	CHECK.Constraints	FOREIGN.KEY.Co
ArtistSimilarity	2	3	3	3 (0)	0 (0)	2 (0)
ArtistTerm	5	7	7	7 (0)	0 (0)	4 (0)
BankAccount	2	9	9	8 (0)	0 (0)	1 (0)
BookTown	22	67	67	28 (1)	2 (1)	0 (0)
BrowserCookies	2	13	13	10 (4)	2 (1)	1 (1)
Cloc	2	10	10	0 (0)	0 (0)	0 (0)
CoffeeOrders	5	20	20	19 (0)	0 (0)	4 (0)
CustomerOrder	7	32	32	42 (1)	1 (1)	7 (0)
DellStore	8	52	52	39 (0)	0 (0)	0 (0)
Employee	1	7	7	4 (0)	3 (0)	0 (0)
Examination	2	21	21	9 (0)	6 (0)	1 (0)
Flights	2	13	13	10 (4)	1 (1)	1 (1)
FrenchTowns	3	14	14	24 (1)	0 (0)	2 (0)
Inventory	1	4	4	2 (0)	0 (0)	0 (0)
Iso3166	1	3	3	3 (0)	0 (0)	0 (0)
iTrust	42	309	309	134 (15)	8 (8)	1 (0)
JWhoisServer	6	49	49	50 (0)	0 (0)	0 (0)
MozillaExtensions	6	51	51	7 (4)	0 (0)	0 (0)
MozillaPermissions	1	8	8	1 (0)	0 (0)	0 (0)
NistDML181	2	7	7	2 (2)	0 (0)	1 (1)
NistDML182	2	32	32	2 (2)	0 (0)	1 (1)
NistDML183	2	6	6	2 (2)	0 (0)	1 (1)
NistWeather	2	9	9	13 (6)	5 (5)	1 (0)
NistXTS748	1	3	3	3 (0)	1 (0)	0 (0)
NistXTS749	2	7	7	7 (1)	1 (0)	1 (0)
Person	1	5	5	7 (1)	1 (1)	0 (0)
Products	3	9	9	14 (1)	4 (0)	2 (0)
RiskIt	13	57	57	36 (1)	0 (0)	10 (0)
StackOverflow	4	43	43	5 (0)	0 (0)	0 (0)
StudentResidence	2	6	6	8 (0)	3 (0)	1 (0)
UnixUsage	8	32	32	24 (1)	0 (0)	7 (0)
Usda	10	67	67	31 (0)	0 (0)	0 (0)
Total	172	975	975	554 (47)	38 (18)	49 (5)

Results

Test generation Timing

When comparing test generation time we look at how efficient the technique are in regard of the time it takes to generate test suites (ALL AVM and DR has 100% coverage). Figure 1 shows the average test generation timing for each technique for each DBMS, for all runs and schemas. Just By looking at the graph it shows that Directed Random is much faster/efficient compared to AVM in generating test cases nearly 1 second faster for different SQL database engine. ??? Why Postgres takes longer for each AVM and DR compared to other engines ? different semantics or larger engine ?

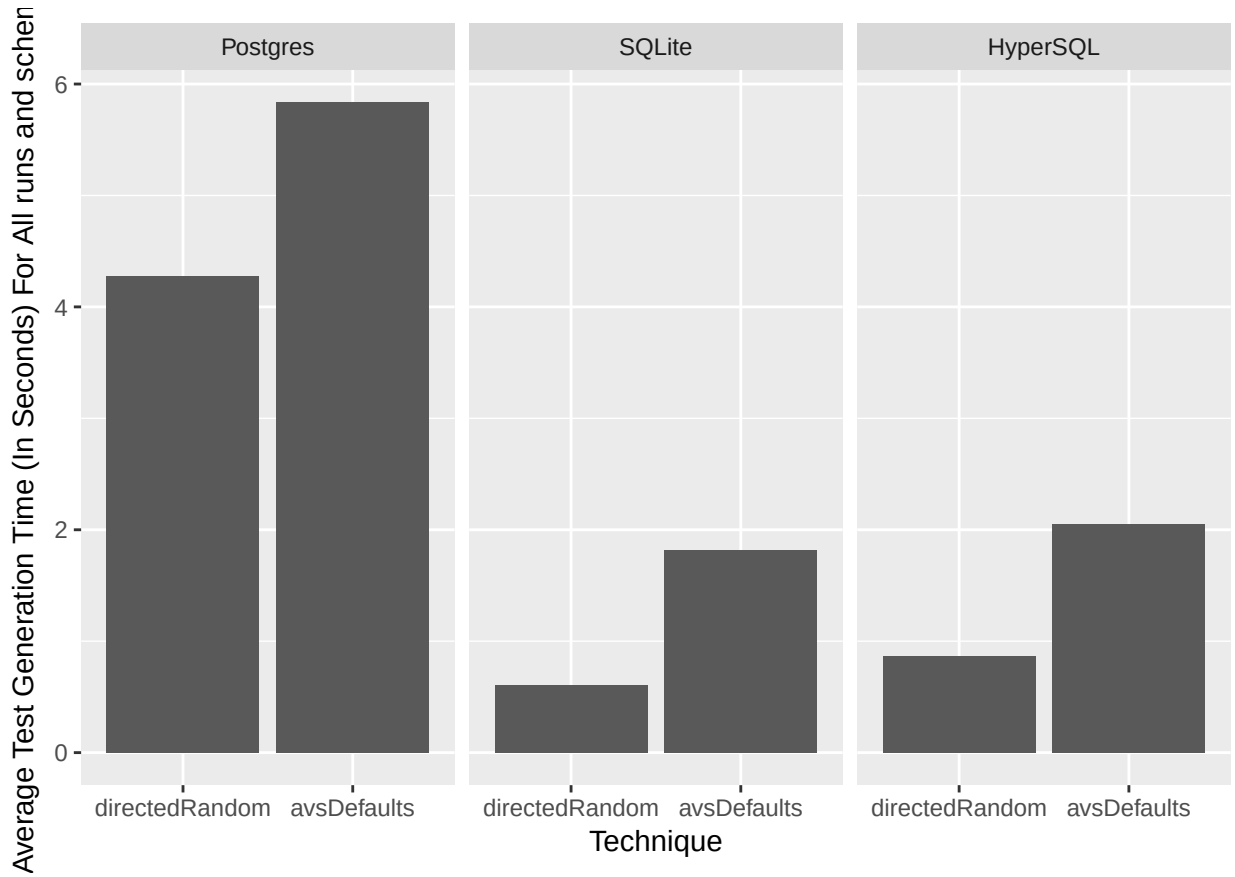

Figure 1: Avrages of Test generation timing - in seconds

To look in more details we split test generation timing analysis for each case study. In Figure 2 we review average test generation timing for each techinque for each schema split by DBMSs and for all runs. We can see that Directed Random still winning for each schema. By looking at all of the results in Figure 2 we can see that DR is better than all AVM even by fractions of seconds.

In Figure 3 I show the spread of values of test generation times in regard of DBMS and technique using a box plot, for all runs and schemas. In this plot we sum all result for each run and spread the values in the box plot, this will help to evaluate the spread of runs for all schema and for each technique split by database engine. As shown in the plot that DR is takes less time compared to AVM in generating test.

In Figure 4 I show the spread of values for test generation timing for each schema, DBMS and technique, for all runs. This will help us seeing the spread of values for each case study and how long it takes to generate test cases.

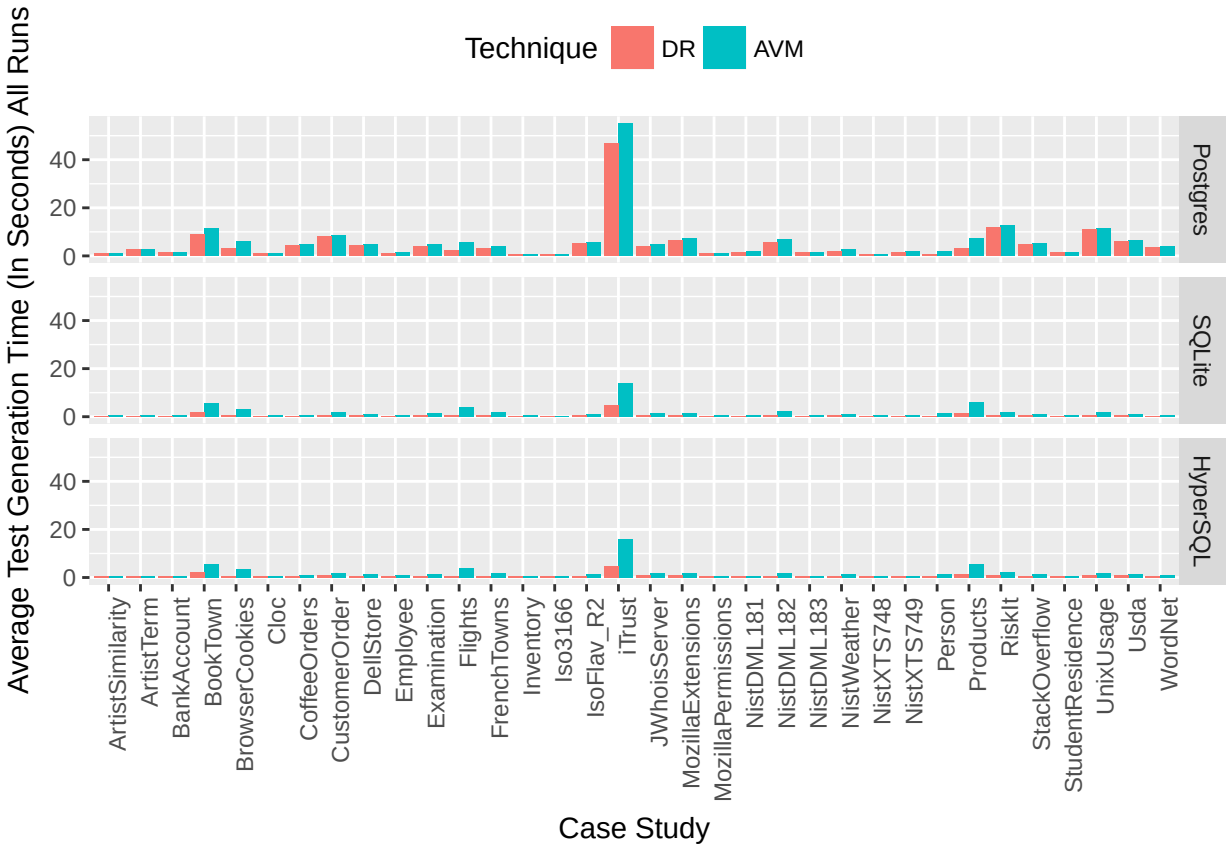

Figure 2: Avrages of Test generation timing for each schema- in seconds

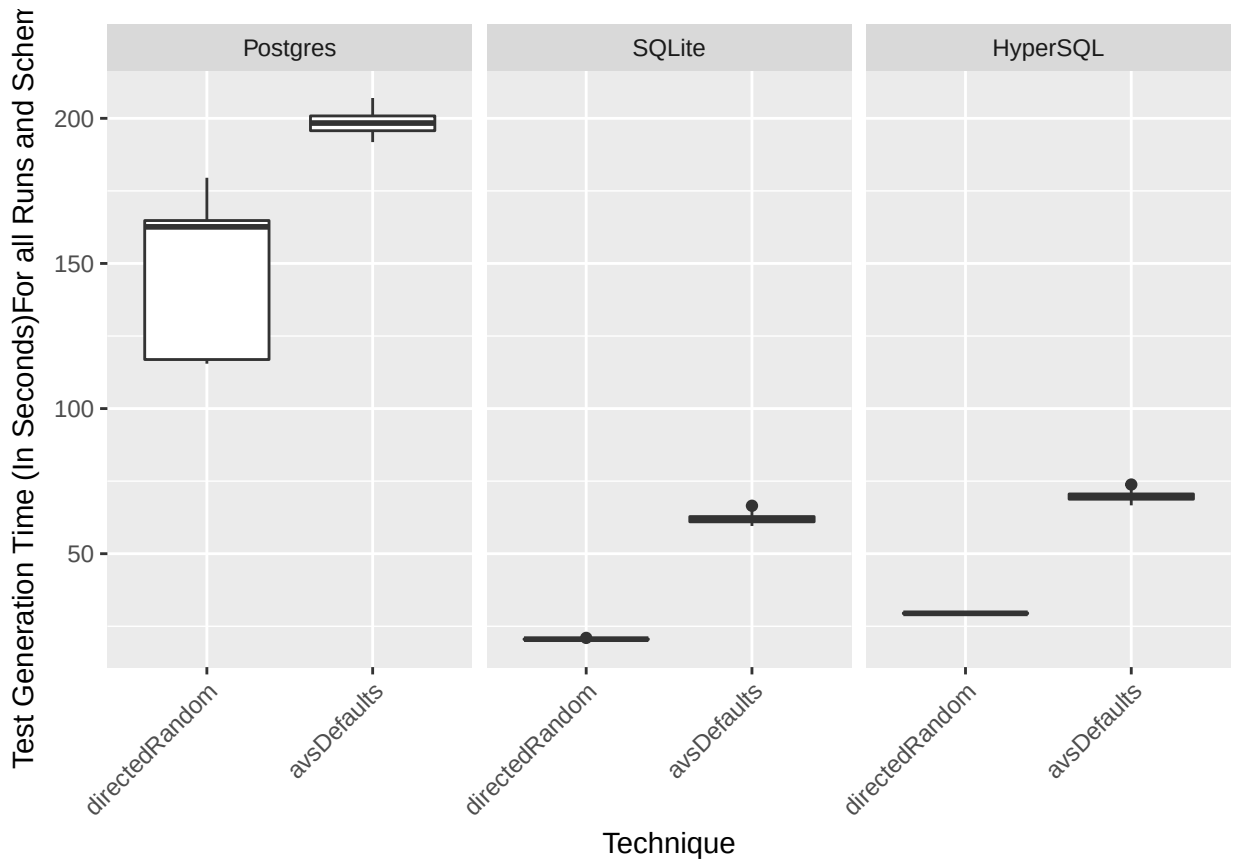

Figure 3: Test generation timing - in seconds

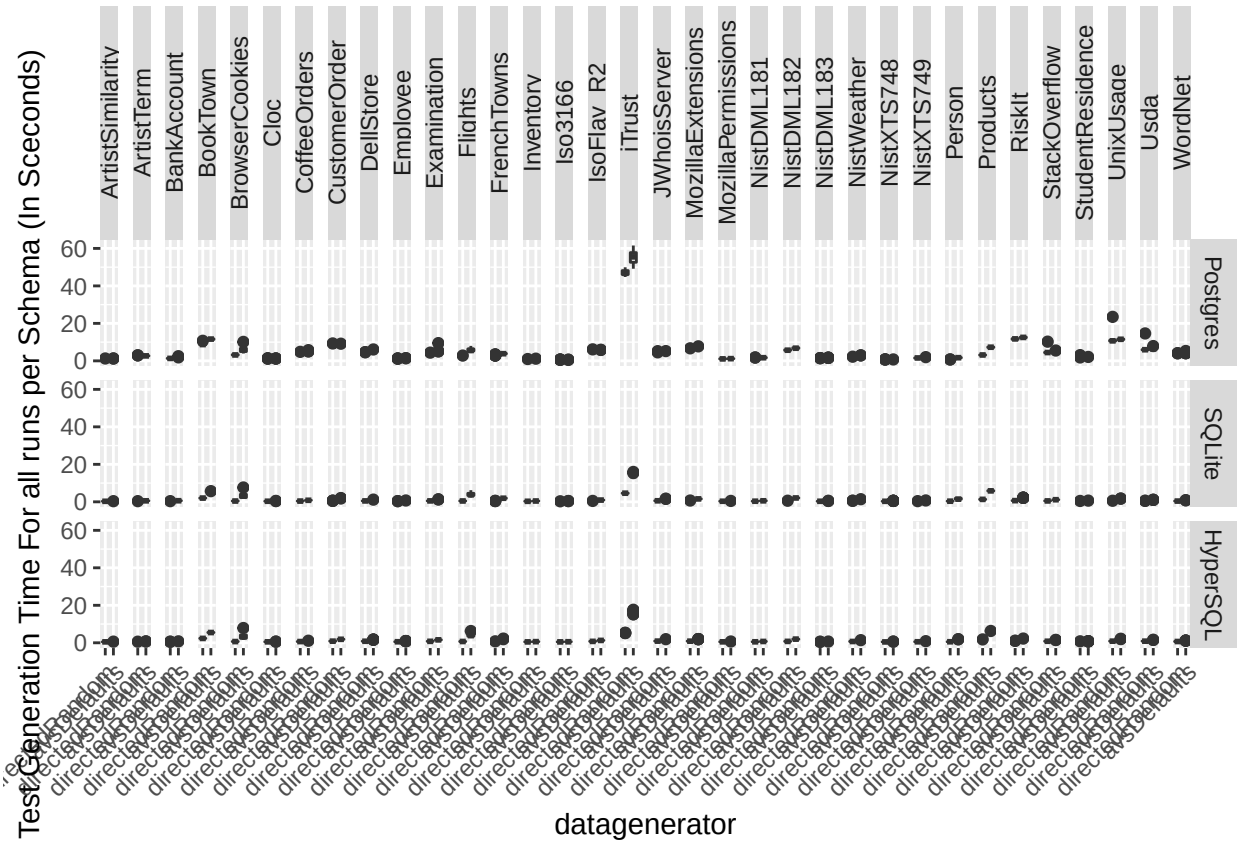

Figure 4: Box plot for mutation scores for DBMS, techniques and schemas - in precentage

Mutation Scores

In Figure 5 I shows the average mutation score for each technique for each DBMS, for all runs and schemas. Just By looking at the graph it shows that Directed Random is batter when compared to AVM in killing more mutants.

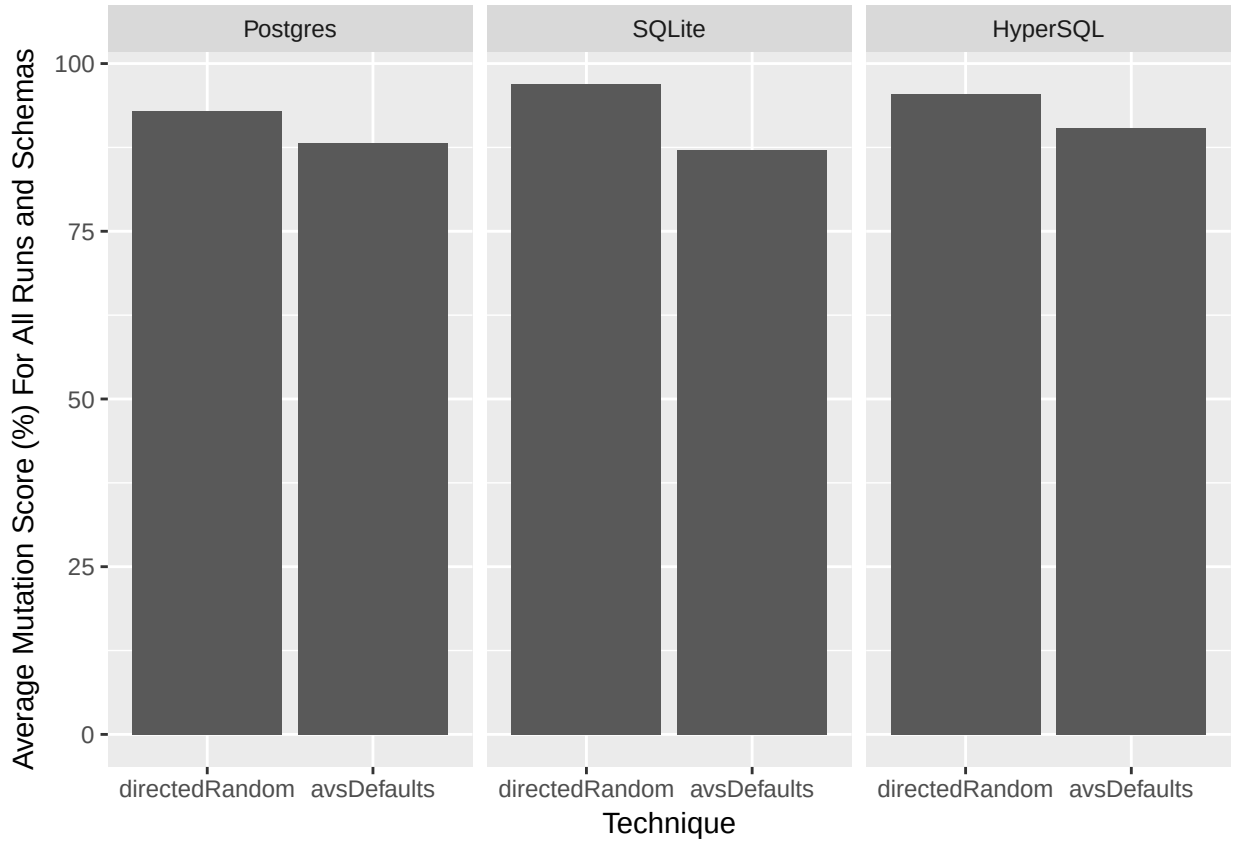

Figure 5: Avrages of Mutation Score - in precentage

In Figure 6 I review average mutation score for each techinque for each schema split by DBMSs, for all runs. We can see that Directed Random have a better or similar scores to AVM however not even one schema has less score comparing to AVM.

In Figure 7 I show the spread of values of mutation score in regard of DBMS and technique using a box plot, for all runs and schemas.

In Figure 8 I show the spread of values for mutation scores for each schema, DBMS and technique, for all runs.

Mutation Scores and Mutation Operators

In Figure 9 I shows the average mutation score for each technique for each DBMS and the mutatan operators, for all runs, schemas and not including Equvilant, Redundant Quasi mutants . Just By looking at the graph it shows that Directed Random is batter when compared to AVM in killing more mutants for two operators the rest shows same percentages.

In Figure 10 I shows a box plot mutation score for each technique for each DBMS and the mutatan operators, this will help us see the spread of values. Just By looking at the graph it shows that Directed Random is batter when compared to AVM in killing more mutants for two operators the rest shows same percentages.

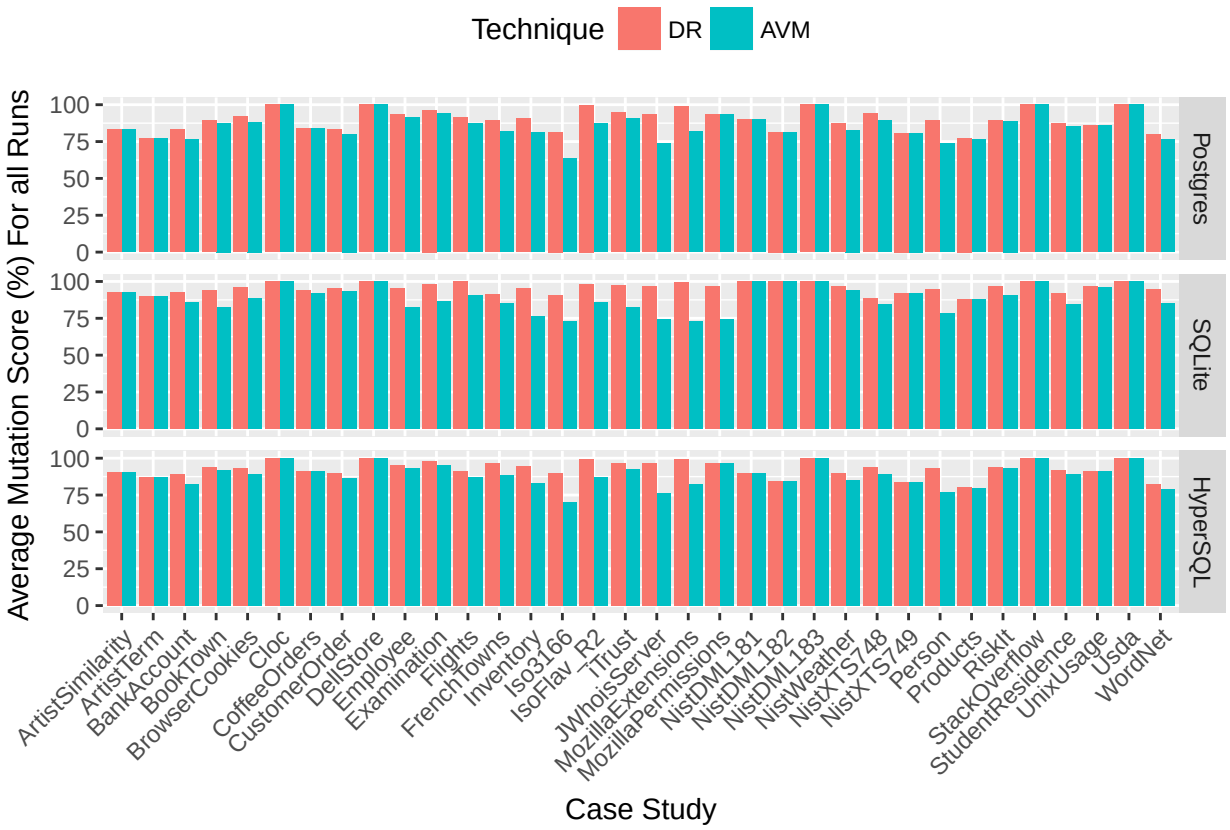

Figure 6: Avrages of Mutation Score for each schema - in precentage

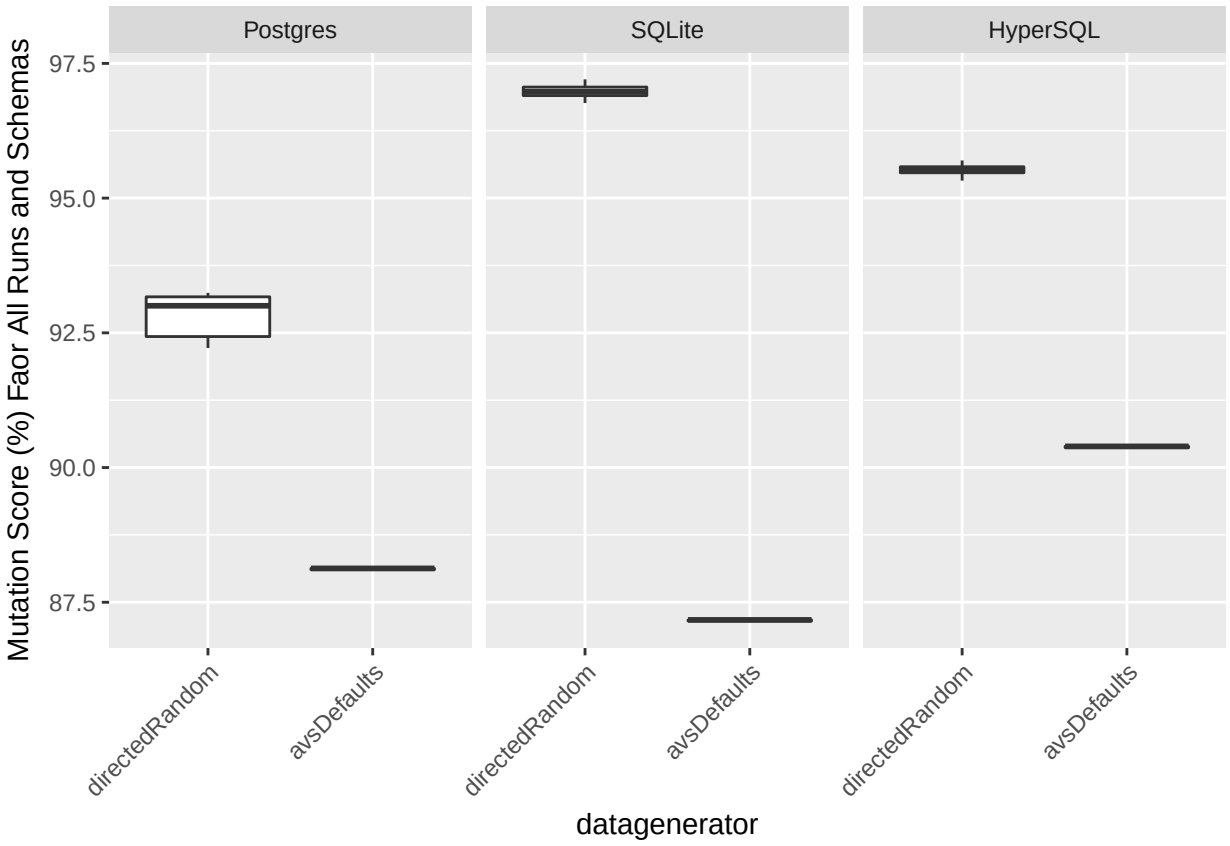

Figure 7: Box plot for mutation scores for DBMSs and techniques - in precentage

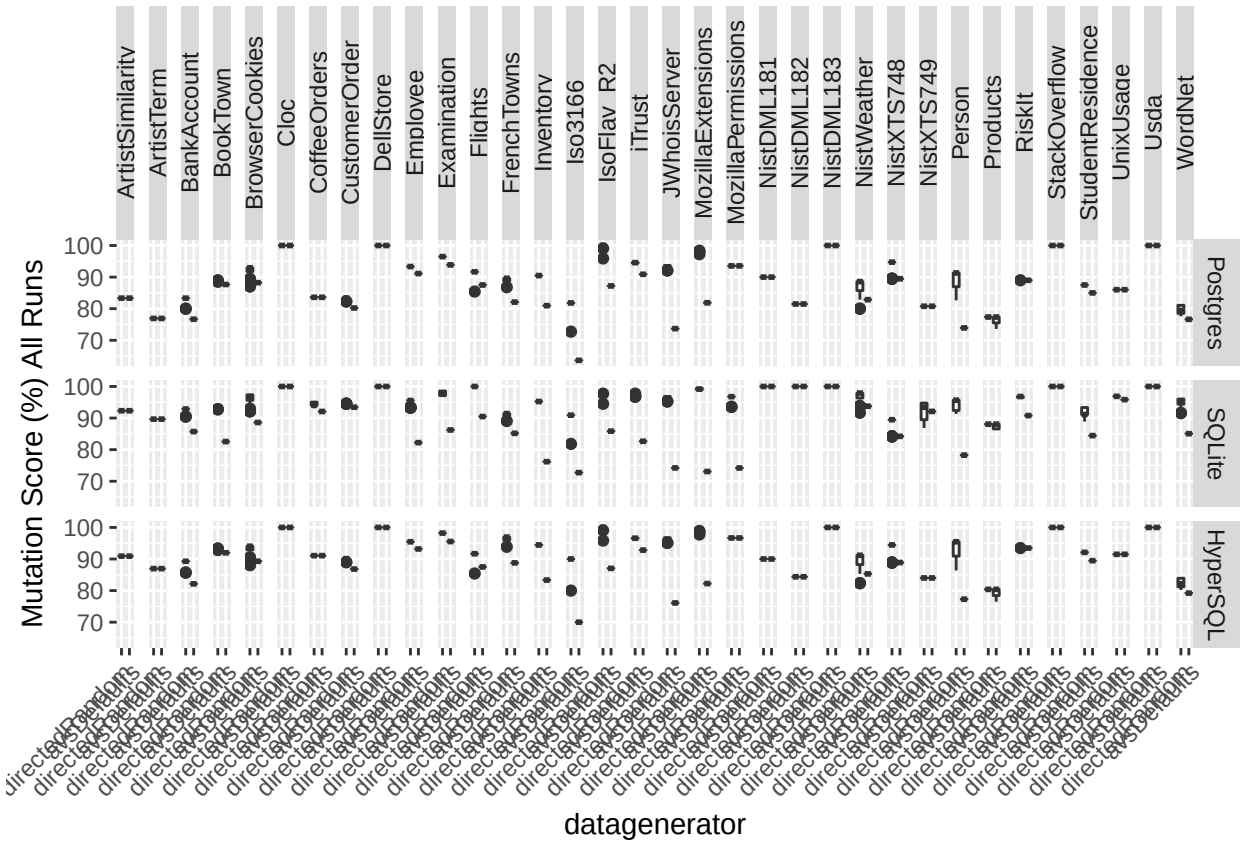

Figure 8: Box plot for mutation scores for DBMS, techniques and schemas - in precentage

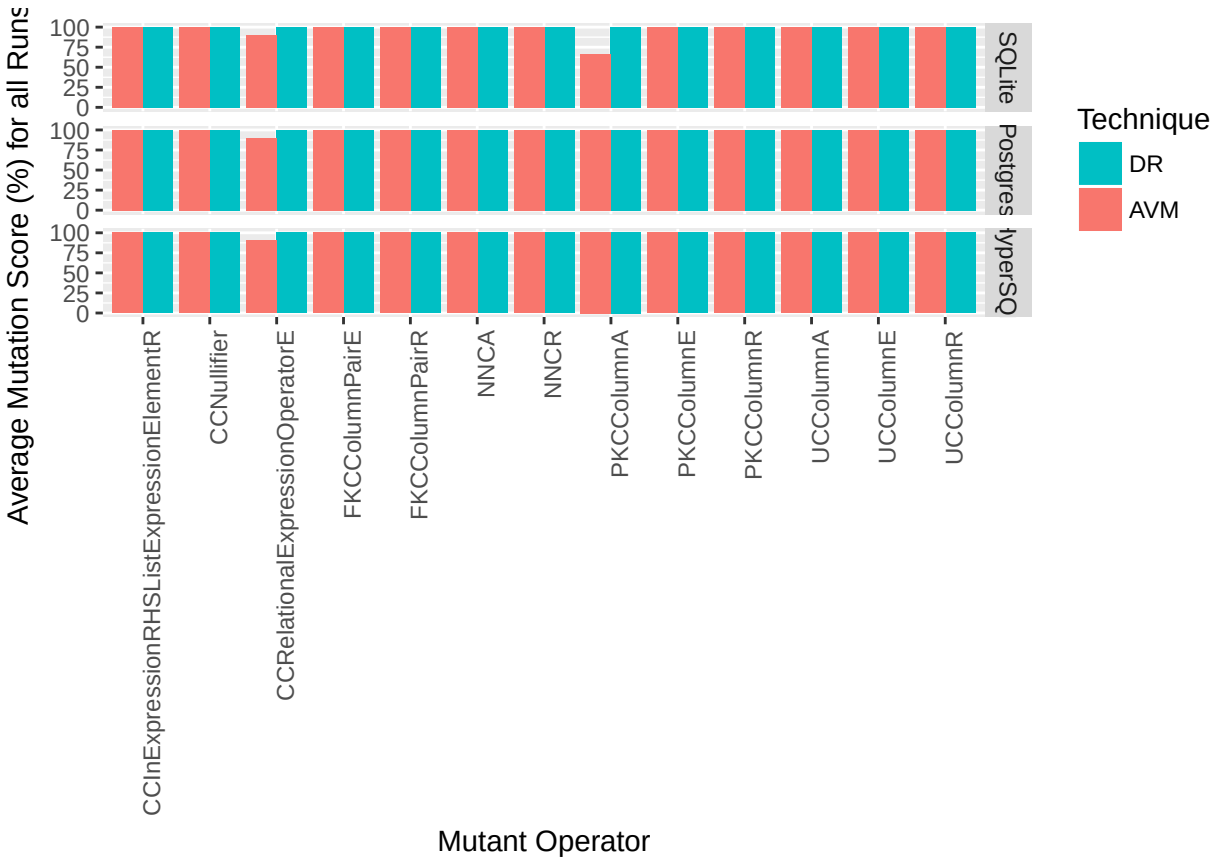

Figure 9: Averages of mutant scores in regard of Mutant Operators and DBMSs - in precentage

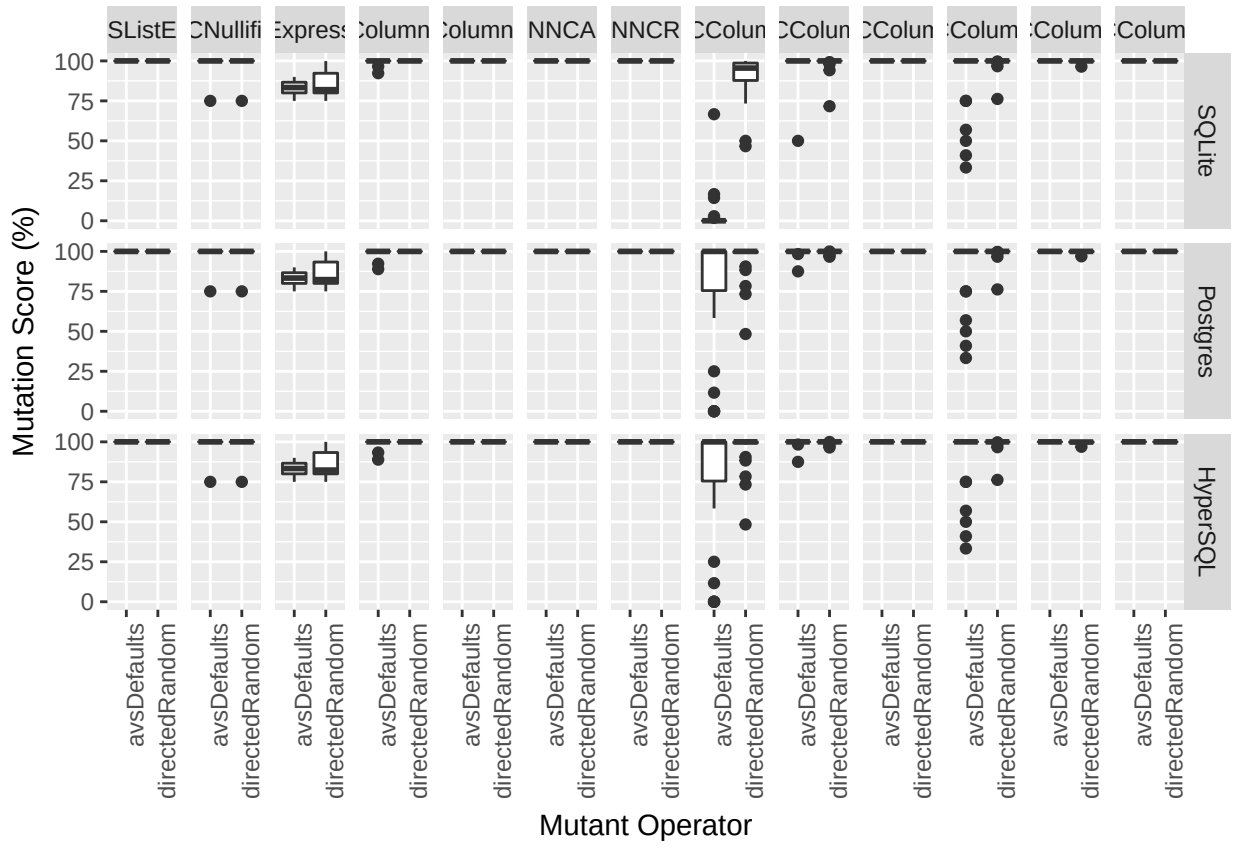

Figure 10: Box Plot of mutant scores in regard of Mutant Operators and DBMSs - in precentage

HeatMap Plot of mutant analysis per technique for each Operator for all schemas

To analysis mutation score in more details we look at the mutant operators that are been killed by both techniques and for each DB engine. In Figure 11 We show that the average percentages of all schemas for each operator that is been killed for each technique. The figure shows that DR has always a higher percentage kill for all operators when compared to AVM in all DB engines which conclude that DR is much superior to AVM technique.

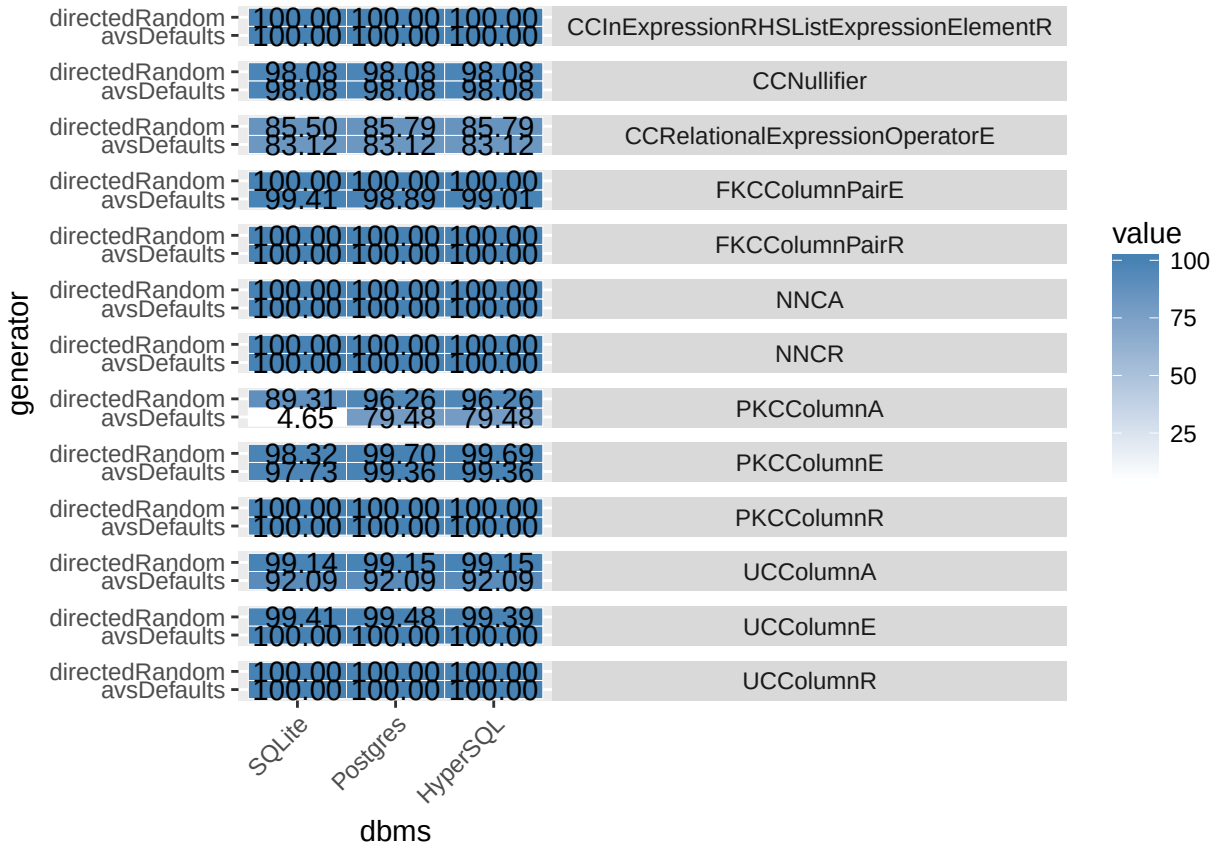

Figure 11: Heat Map of mutant scores in regard of Mutant Operators and DBMSs - in precentage

Analyse Wilcox Rank and Effect Size for AVM and Directed Random

To statistically analyze the the new technique we conducted tests for significance with the nonparametric Wilcoxon rank-sum test, using the sets of 30 execution times obtained with a specific DBMS and all techniques **Hitchhicker Guide Ref**. A p-value that less than 0.05 is considered significant. To review practical are significance tests we use the nonparametric A12 statistic of Vargha and Delaney **REF** was used to calculate effect sizes. The A12 determine the average probability that one approach beats another, or how superior one technique compared to the other. We followed the guidelines of Vargha and Delaney in that an effect size is considered to be "large" if the value of A12 is $ < 0:29$ or > 0.71, "medium" if A12 is < 0.36 or > 0.64 and "small" if A12 is < 0.44 or > 0.56. However, is the values of A12 close to the 0.5 value are viewed as there no effect.

When comparing AVM and Directed Random techniques in regard of time we used Mann-Whitney U-test and the A-hat effect size calculations. As Shown in in the following two tables that there is statistically significat differance between Driected Random and AVM, $p \leq 0.05$. Therefore, we reject the null hypothesis that there is no difference between AVM and Directed Random. As p-value near zero, that directed random

is faster than AVM in a statistically significant test. Moreover, the A-12 shows that Directed Random has a large effect size when it comes to test generation timing. Which menas that Directed Random is the winner in regard of test generation timing.

p.value	dbms	vs
0.93791	Postgres	AVM vs Directed Random Coverage
1.00000	SQLite	AVM vs Directed Random Coverage
1.00000	HyperSQL	AVM vs Directed Random Coverage
0.00000	Postgres	AVM vs Directed Random Mutation Score
0.00000	SQLite	AVM vs Directed Random Mutation Score
0.00000	HyperSQL	AVM vs Directed Random Mutation Score
0.00000	Postgres	AVM vs Directed Random Number Of evaluations
0.00000	SQLite	AVM vs Directed Random Number Of evaluations
0.00000	HyperSQL	AVM vs Directed Random Number Of evaluations
0.00000	Postgres	AVM vs Directed Random Test Generation Time
0.00000	SQLite	AVM vs Directed Random Test Generation Time
0.00000	HyperSQL	AVM vs Directed Random Test Generation Time

HeatMap Plot of mutant analysis per technique for each Operator

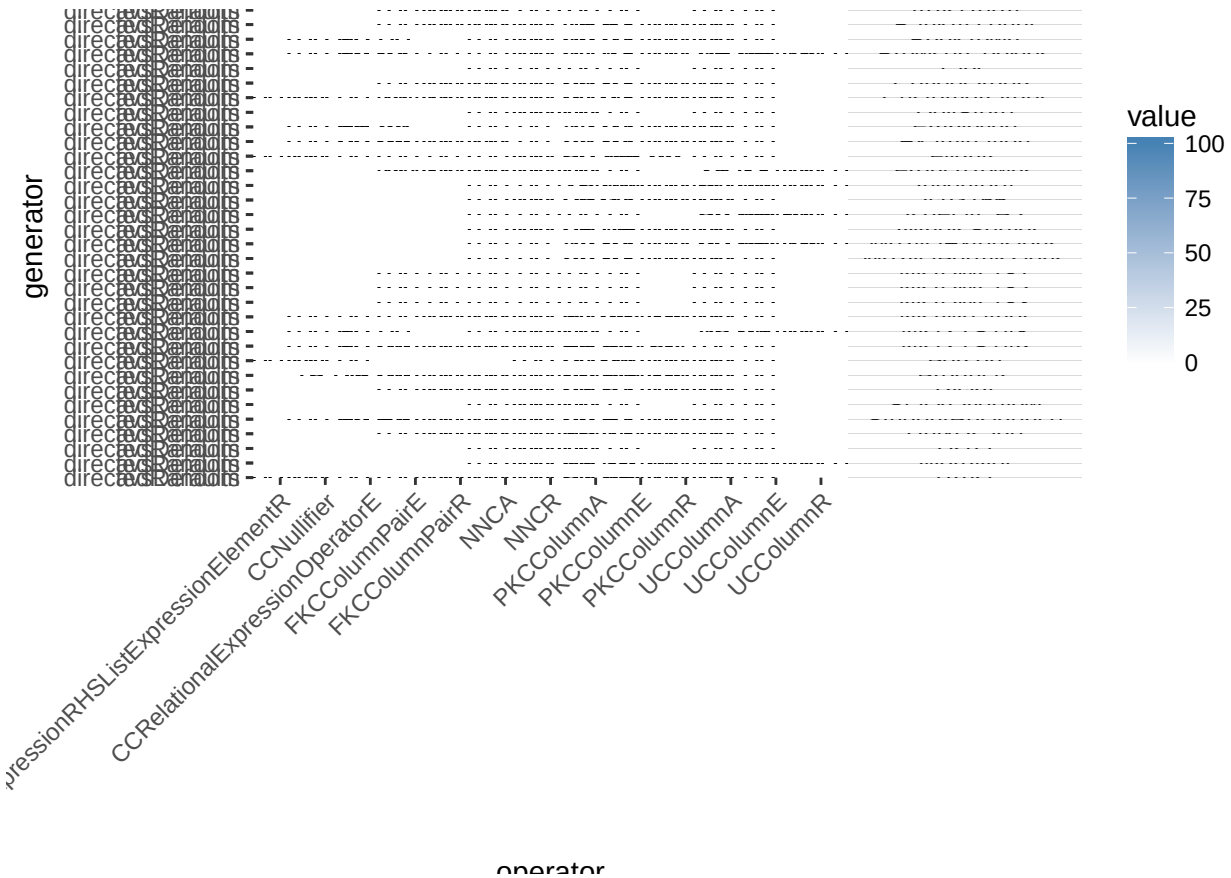

Figure 12: Heat Map of mutant scores in regard of Mutant Operators and DBMSs - in precentage