Massive Stochastic Testing of SQL

DON SLUTZ @ MICROSOFT RESEARCH, VLDB’98

PRESENTED BY: YUSHAN ZHANG

https://home.cse.ust.hk/~yzhanghw/
Problems

➢ *Gigantic input domain* is hard to cover (test input) => How to effectively explore the input domain?
➢ SQL statements (complex construction)
➢ States of the database system (configurations, versions…)

➢ Difficulty of *test output verification* (oracle) => How to automate output verification?
➢ Extensive human-effort in maintenance

![Diagram of input domain and related concepts]
Intuition

=> How to effectively explore the input domain?
• Greatly enlarge the shaded circle by stochastic SQL generation
• Make all aspects of the generated SQL configurable
• Experiment with configurations to maximize the bug detection rate

=> How to automate output verification?
• Compare execution results from different vendors - system differential testing
RAGS (Random Generation of SQL)

**Stochastic generation:**
- Frequency of different statements (select, insert...)
- Limits (#tables in a join, #entries in a group by...)
- Frequency of features (outer join, where, group by...)

**Output verification:**
- To avoid specific comparisons, the vendor identities are suppressed
- The mapping to actual vendors is changed from test to test.
Example SQLs

SELECT T0.au_id , LTRIM(('cuIfzce' +T0.au_id )))
FROM authors T0
WHERE NOT (NOT ((T0.au_fname )!= ANY (SELECT 'E' FROM discounts T1, authors T2 WHERE NOT (|K'| >= 'tKpc|AV' ) )) )
GROUP BY T0.au_id , T0.au_id

Characteristic:
• 50% of non-error causing selects return rows
Testing Experiences

➢ Multi-user Test
➢ Comparison Tests
➢ Statement Simplification
➢ Visualization

Technical details are not discussed here.
Multi-user (Concurrency) Test

Setup: 10 clients, 2500 statements per client, 5 statements per transaction

Statements: Select, Insert, Update, and Delete statements on a single database system.

<table>
<thead>
<tr>
<th>Item</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of clients</td>
<td>10</td>
</tr>
<tr>
<td>SQL statements per client</td>
<td>2500</td>
</tr>
<tr>
<td>Total number of statements</td>
<td>25000</td>
</tr>
<tr>
<td>Statements per transaction</td>
<td>1 to 9</td>
</tr>
<tr>
<td>Execution Results:</td>
<td></td>
</tr>
<tr>
<td>Number of Selects</td>
<td>6401</td>
</tr>
<tr>
<td>Number of Inserts</td>
<td>6165</td>
</tr>
<tr>
<td>Number of Deletes</td>
<td>6221</td>
</tr>
<tr>
<td>Number of Updates</td>
<td>6213</td>
</tr>
<tr>
<td>Number of Transactions</td>
<td>4997</td>
</tr>
<tr>
<td>Execution with no errors</td>
<td>21518</td>
</tr>
<tr>
<td>Errors - expected:</td>
<td></td>
</tr>
<tr>
<td>Deadlock victim</td>
<td>2715</td>
</tr>
<tr>
<td>Arithmetic error</td>
<td>553</td>
</tr>
<tr>
<td>Character value too long</td>
<td>196</td>
</tr>
<tr>
<td>Errors - not expected (bugs)</td>
<td></td>
</tr>
<tr>
<td>Error code 1</td>
<td>13</td>
</tr>
<tr>
<td>Error code 2</td>
<td>5</td>
</tr>
</tbody>
</table>
Comparison Test

Run the same **2000 random selects** on each of the **four** systems

<table>
<thead>
<tr>
<th>Comparison Case</th>
<th>SYSA</th>
<th>SYSB</th>
<th>SYSC</th>
<th>SYSD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1672</td>
<td>1672</td>
<td>1672</td>
<td>1672</td>
</tr>
<tr>
<td></td>
<td>232</td>
<td>234</td>
<td>241</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>15</td>
<td>12</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>12</td>
<td>5</td>
<td>116</td>
</tr>
<tr>
<td>Error</td>
<td>0</td>
<td>29</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>18</td>
<td>19</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>19</td>
<td>18</td>
<td>113</td>
</tr>
</tbody>
</table>

All four agree 84%

Probably a bug
Statement Simplification

- Remove as many elements of the statement as possible
- Preserving the raising of the original error message
- Does NOT need to produce a statement that is equivalent to the original SQL statement

```sql
SELECT TOP 2 '6c', -(-2), T0.min lvl, ' ^p:'
FROM jobs T0, sales T1 WHERE EXISTS (  
    SELECT DISTINCT TOP 1 T1.ord_date, 'Jul 15 4792 4:16am'
    FROM discounts T2, discounts T3
    ORDER BY 2,1)
```
Visualization

SYSB tested locally
SYSD tested remotely

SYSC two versions: v1 and v2
Extensions (future works)

- Measure code coverage under random SQL testing
- Investigate bugs that reported by the user, but not discovered by RAGS
- Generate equivalent SQLs and compare the result
- Compare optimizer estimations with the measured execution metrics
Testing works by Microsoft


