Towards Correct and Reliable Data-centric Systems

Manuel Rigger

National University of Singapore

Example: MySQL

```
SELECT * FROM t0, t1;
```

t0.c0	t1.c0
0.0	-0.0

ϕ

```
SELECT * FROM t0, t1 WHERE t0.c0=t1.c0
UNION ALL
SELECT * FROM t0, t1 WHERE NOT (t0.c0=t1.c0)
UNION ALL
SELECT * FROM t0, t1 WHERE (t0.c0=t1.c0) IS NULL;
```

t0.c0	t1.c0

≠

https://bugs.mysql.com/bug.php?id=99122

Architecture

DuckDB TLP Oracle	MySQL TLP Oracle	…
DuckDB Query Generator	MySQL Query Generator	…
DuckDB Database Generator	MySQL Database Generator	…
SQLancer Base (logging, thread handling, …)		

The DBMS-specific components are large and share less code than they should

DatabaseProvider

```
public class DuckDBProvider extends SQLProviderAdapter<DuckDBGlobalState, DuckDBOptions> {
    public enum Action implements AbstractAction<DuckDBGlobalState> {


        INSERT(DuckDBInsertGenerator::getQuery), //
        CREATE_INDEX(DuckDBIndexGenerator::getQuery), //
        VACUUM((g) -> new SQLQueryAdapter("VACUUM;")), //
        ANALYZE((g) -> new SQLQueryAdapter("ANALYZE;")), //
        DELETE(DuckDBDeleteGenerator::generate), //
        UPDATE(DuckDBUpdateGenerator::getQuery), //
        CREATE_VIEW(DuckDBViewGenerator::generate), //
        …
    });
}
```

The DatabaseProvider subclasses are the main entry points for a DBMS implementation

Statement Generators and Expected Errors

```
private SQLQueryAdapter generate() {
    sb.append("INSERT INTO ");
    DuckDBTable table = globalState.getSchema().getRandomTable(t -> !t.isView());
    List<DuckDBColumn> columns = table.getRandomNonEmptyColumnSubset();
    sb.append(table.getName());
    sb.append("(");
    sb.append(columns.stream().map(c -> c.getName()).collect(Collectors.joining(", ")));
    sb.append(")");
    sb.append(" VALUES ");
    insertColumns(columns);
    DuckDBErrors.addInsertErrors(errors);
    return new SQLQueryAdapter(sb.toString(), errors);
}
```

```
errors.add("PRIMARY KEY or
UNIQUE constraint violated");
```

Some semantic errors are difficult to prevent, while others might be unexpected (e.g., database corruptions)

IgnoreMeException

```
public A getRandomTableOrBailout() {
    if (databaseTables.isEmpty()) {
        throw new IgnoreMeException();
    } else {
        return Randomly.fromList(getDatabaseTables());
    }
}
```

In some context it's easier to bail out rather than first checking whether all preconditions for an action are met

Expression Generators

```
public final class DuckDBExpressionGenerator extends UntypedExpressionGenerator<Node<DuckDBExpression>, DuckDBColumn> {



    private enum Expression {
        UNARY_POSTFIX, UNARY_PREFIX, BINARY_COMPARISON, BINARY_LOGICAL, BINARY_ARITHMETIC, CAST, FUNC, BETWEEN, CASE, IN, COLLATE,
LIKE_ESCAPE
    }


    @Override
    protected Node<DuckDBExpression> generateExpression(int depth) {
    if (depth >= globalState.getOptions().getMaxExpressionDepth() || Randomly.getBoolean()) {
        return generateLeafNode();
    }
    Expression expr = Randomly.fromOptions(Expression.values());
    switch (expr) {
        case UNARY_PREFIX:
            return new NewUnaryPrefixOperatorNode<DuckDBExpression>(generateExpression(depth + 1), DuckDBUnaryPrefixOperator.getRandom());
        case UNARY_POSTFIX:
            return new NewUnaryPostfixOperatorNode<DuckDBExpression>(generateExpression(depth + 1), DuckDBUnaryPostfixOperator.getRandom());
        case BINARY_COMPARISON:
            Operator op = DuckDBBinaryComparisonOperator.getRandom();
            return new NewBinaryOperatorNode<DuckDBExpression>(generateExpression(depth + 1), generateExpression(depth + 1), op);

    }
}
```

Untyped Expression Generators

```
CREATE TABLE t0(c0 INT);
INSERT INTO t0 VALUES ('I am an int');
SELECT * FROM t0 WHERE c0 > 'Hello';
```

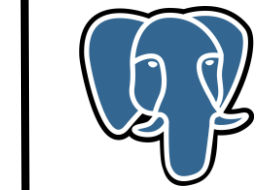

```
'I am an int'
```


https://www.db-fiddle.com/f/kiodqyYUDMvLV96Tf7jnnX/1

Typed Expression Generators

Schema Error: error: invalid input syntax for type integer: "I am an int"

```
CREATE TABLE t0(c0 INT);
INSERT INTO t0 VALUES ('I am an int');
SELECT * FROM t0 WHERE c0 > 'Hello';
```



Query Error: error: invalid input syntax for type integer: "Hello"

https://www.db-fiddle.com/f/8mswd2wKnNCTNYWJfnYnQ1/0

Test Oracle Example

```
public class DuckDBQueryPartitioningWhereTester extends DuckDBQueryPartitioningBase {

    @Override
    public void check() throws SQLException {
        super.check();
        select.setWhereClause(null);
        String originalQueryString = DuckDBToStringVisitor.asString(select);
        List<String> resultSet = ComparatorHelper.getResultSetFirstColumnAsString(originalQueryString, errors, state);
        boolean orderBy = Randomly.getBooleanWithRatherLowProbability();
        if (orderBy) {
            select.setOrderByExpressions(gen.generateOrderBys());
        }
        select.setWhereClause(predicate);
        String firstQueryString = DuckDBToStringVisitor.asString(select);
        select.setWhereClause(negatedPredicate);
        String secondQueryString = DuckDBToStringVisitor.asString(select);
        select.setWhereClause(isNullPredicate);
        String thirdQueryString = DuckDBToStringVisitor.asString(select);
        List<String> combinedString = new ArrayList<>();
        List<String> secondResultSet = ComparatorHelper.getCombinedResultSet(firstQueryString, secondQueryString,
thirdQueryString, combinedString, !orderBy, state, errors);
        ComparatorHelper.assumeResultSetsAreEqual(resultSet, secondResultSet, originalQueryString, combinedString, state,
ComparatorHelper::canonicalizeResultValue);
    }

}
```

Options

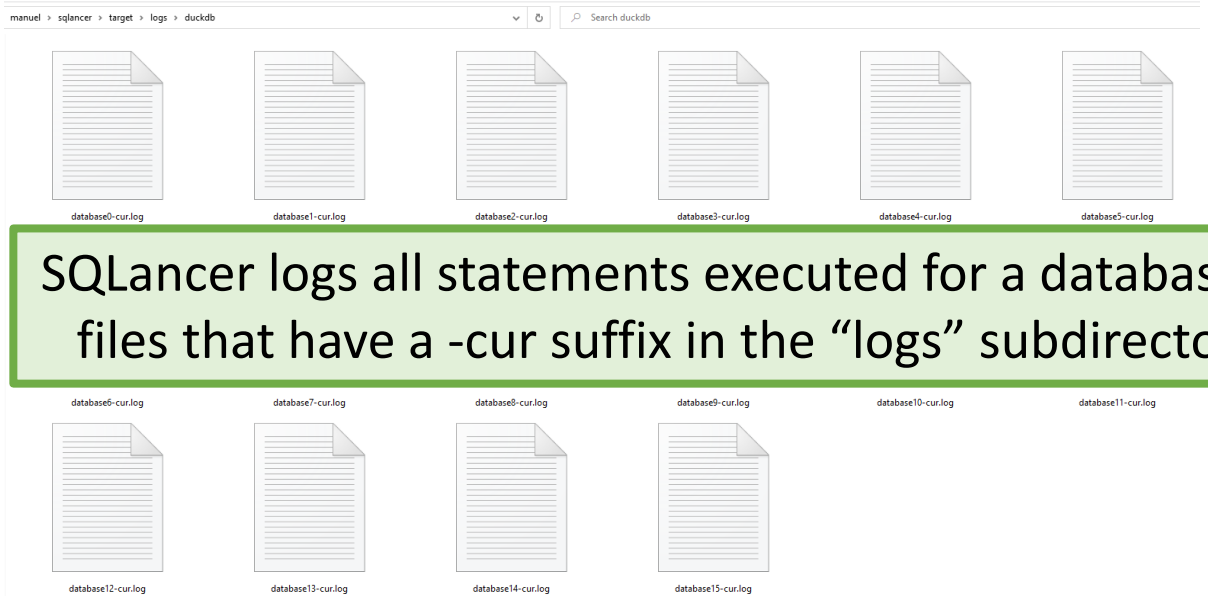
```
@Parameters(commandDescription = "DuckDB")
public class DuckDBOptions implements DBMSSpecificOptions<DuckDBOracleFactory> {

    @Parameter(names = "--oracle")
    public List<DuckDBOracleFactory> oracles = Arrays.asList(DuckDBOracleFactory.WHERE);

    public enum DuckDBOracleFactory implements OracleFactory<DuckDBGlobalState> {
        WHERE {
            @Override
            public TestOracle<DuckDBGlobalState> create(DuckDBGlobalState globalState) {
                return new DuckDBQueryPartitioningWhereTester(globalState);
            }
        }
    };

    @Override
    public List<DuckDBOracleFactory> getTestOracleFactory() {
        return oracles;
    }

}
```

The MainOptions class specifies options applicable to all DBMSs

Duplicate Bugs

```
public final class TiDBBugs {
    // https://github.com/pingcap/tidb/issues/35677
    public static boolean bug35677 = true;


    // https://github.com/pingcap/tidb/issues/35522
    public static boolean bug35522 = true;


    // https://github.com/pingcap/tidb/issues/35652
    public static boolean bug35652 = true;


    // https://github.com/pingcap/tidb/issues/38295
    public static boolean bug38295 = true;


}
```

Logs



SQLancer logs all statements executed for a database in files that have a -cur suffix in the “logs” subdirectory

Hint: The log files are overwritten for each newly-generated database. If SQLancer finds a bug, it creates an additional log file without the -cur suffix.

Output

[2023/06/07 08:24:00] Executed 38362 queries (7632 queries/s; 2.19/s dbs, successful statements: 88%). Threads shut down: 0.
[2023/06/07 08:24:05] Executed 143039 queries (20973 queries/s; 0.80/s dbs, successful statements: 93%). Threads shut down: 0.
[2023/06/07 08:24:10] Executed 266483 queries (24867 queries/s; 0.20/s dbs, successful statements: 95%). Threads shut down: 0.
[2023/06/07 08:24:15] Executed 394076 queries (25615 queries/s; 0.00/s dbs, successful statements: 95%). Threads shut down: 0.

We generate small databases (few tables and rows), for which we create many queries for efficiency

Hint: Use `java -jar sqlancer-2.0.0.jar --num-queries 1 --max-num-inserts 50 duckdb` to create one query per database and increase the number of rows inserted to 50

Expected Errors

[2023/06/07 08:24:00] Executed 38362 queries (7632 queries/s; 2.19/s dbs, successful statements: 88%). Threads shut down: 0.
[2023/06/07 08:24:05] Executed 143039 queries (20973 queries/s; 0.80/s dbs, successful statements: 93%). Threads shut down: 0.
[2023/06/07 08:24:10] Executed 266483 queries (24867 queries/s; 0.20/s dbs, successful statements: 95%). Threads shut down: 0.
[2023/06/07 08:24:15] Executed 394076 queries (25615 queries/s; 0.00/s dbs, successful statements: 95%). Threads shut down: 0.

SQLancer uses various empirically-determined heuristics and mechanisms to make it more likely to generate semantically valid statements

Hint: All statements are expected to be syntactically valid, since the database and query generators are specific to the database system under test.

Output

```
[2023/06/07 08:24:00] Executed 38362 queries   (7632 queries/s; 2.19/s dbs, successful statements: 88%). Threads shut down: 0.
[2023/06/07 08:24:05] Executed 143039 queries (20973 queries/s; 0.80/s dbs, successful statements: 93%). Threads shut down: 0.
[2023/06/07 08:24:10] Executed 266483 queries (24867 queries/s; 0.20/s dbs, successful statements: 95%). Threads shut down: 0.
[2023/06/07 08:24:15] Executed 394076 queries (25615 queries/s; 0.00/s dbs, successful statements: 95%). Threads shut down: 0.
```

Each thread tests the DBMS using a separate database

Hint: Use `java -jar sqlancer-2.0.0.jar --num-threads 1 duckdb` for single-threaded execution (e.g., useful for debugging)

Testing an Old Version of SQLite

```
diff --git a/pom.xml b/pom.xml
index 46211aac..f35d9ff1 100644
--- a/pom.xml
+++ b/pom.xml
@@ -299,7 +299,7 @@
<dependency>
<groupId>org.xerial</groupId>
<artifactId>sqlite-jdbc</artifactId>
-      <version>3.40.0.0</version>
+      <version>3.27.2</version>
</dependency>
<dependency>
<groupId>mysql</groupId>
```

Three Tasks

- Task 1: add a new statement generator
  - Learn how to add support for new DBMSs
- Task 2: add a new test oracle
- Task 3: add a new test-case generation technique

Task 1: Pragmas

- Generate pragmas (i.e., options) for DuckDB
- For simplicity, just choose a single pragma

```
PRAGMA disable_optimizer;
```

https://duckdb.org/docs/sql/pragmas

Task 1: Pragmas

- Take a look at `DuckDBProvider` to see how other generators are implemented

```
PRAGMA disable_optimizer;
```

Task 2: New Test Oracle

- Create a predicate that always is false (i.e., a contradiction)
- Validate that the result bag returned by the database system is indeed empty

WHERE clause

```
SELECT * FROM t0
WHERE <p> AND NOT <p>;
```

HAVING clause

```
SELECT * FROM t0
WHERE ...
GROUP BY ...
HAVING <p> AND NOT <p>;
```

Task 2: New Test Oracle

WHERE clause

```
SELECT * FROM t0
WHERE <p> AND NOT <p>;
```

- Take a look at `DuckDBQueryPartitioningWhereTester`
- We can implement our test oracle by inheriting from `DuckDBQueryPartitioningBase` which already creates a query skeleton (`select`)
- We can generate an **`AND`** in the same way as `DuckDBExpressionGenerator`
- Add the new test oracle as an option in `DuckDBOptions`

Task 2: WHERE Contradiction Oracle

```
public class DuckDBContradictionOracle extends DuckDBQueryPartitioningBase {


    public DuckDBContradictionOracle(DuckDBGlobalState state) {
        super(state);
    }


    @Override
    public void check() throws SQLException {
        super.check();
        select.setWhereClause(new NewBinaryOperatorNode<DuckDBExpression>(predicate, negatedPredicate,
DuckDBBinaryLogicalOperator.AND));
        String selectStr = DuckDBToStringVisitor.asString(select);
        List<String> resultSet = ComparatorHelper.getResultSetFirstColumnAsString(selectStr, errors, st
        if (!resultSet.isEmpty() ) {
            throw new AssertionError(selectStr + " " + resultSet);
        }
    }
}
```

Task 3: Duplicate queries

sort logs/duckdb/database0-cur.log | uniq -cd | sort

```
220 SELECT * FROM t1 WHERE ((true)AND((NOT true)));
220 SELECT t0.c0 FROM t0 WHERE ((true)AND((NOT true)));
497 SELECT * FROM t0, t1 WHERE ((t1.c1)AND((NOT t1.c1)));
504 SELECT * FROM t0, t1 WHERE ((t1.c0)AND((NOT t1.c0)));
```

Observation: many queries that are generated are redundant

```
593 SELECT * FROM t0, t1 WHERE ((t0.c0)AND((NOT t0.c0)));
777 SELECT * FROM t1 WHERE ((t1.rowid)AND((NOT t1.rowid)));
793 SELECT * FROM t1 WHERE ((t1.c0)AND((NOT t1.c0)));
797 SELECT * FROM t1 WHERE ((t1.c1)AND((NOT t1.c1)));
2365 SELECT * FROM t0 WHERE ((t0.c0)AND((NOT t0.c0)));
2400 SELECT t0.c0 FROM t0 WHERE ((t0.c0)AND((NOT t0.c0)));
```

Task 3: Enumerate predicates

- Idea: rather than generating a random query every call to the test oracle, enumerate queries in a systematic way
- For simplicity, enumerate only the **`WHERE`** predicate (for the partitioning queries in the TLP WHERE oracle)
  - The original query needs to be executed only once (speed up of close to 2x)

```
SELECT t1.rowid FROM t1 WHERE c0 = 0;
SELECT t1.rowid FROM t1 WHERE c0 > 0;
SELECT t1.rowid FROM t1 WHERE c0 >= 0;
…
```

Task 3: Enumerate predicates

- Take `DuckDBExpressionGenerator` as an example and create your own implementation that returns a list of predicates
- For simplicity, consider only leaf nodes (see `generateLeafNode`) and comparisons operators
- Rather than enumerating all constants, pick some corner case values (e.g., `DuckDBConstant.createIntConstant(0)`)
- In `DuckDBQueryPartitioningWhereTester`, enumerate multiple predicates
- You can update the statistics via `Main.nrQueries.addAndGet(1);` and `Main.nrSuccessfulActions.addAndGet(1);`

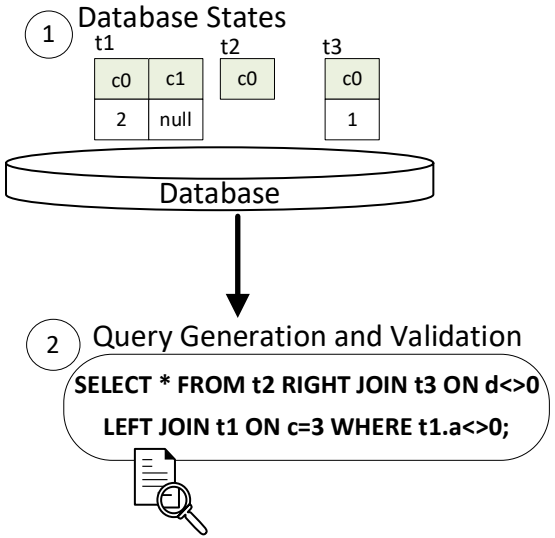
Exercise Conclusion

- You now know how to implement new test oracles and test-case generation approaches in SQLancer
- Chance for developing new approaches
  - Test oracle: Still many features have not been tested (e.g., window functions, procedural extensions, nested queries, …)
  - Test case generation: already outperforms SQLancer's current one by around 2x in terms of efficiency

Test Case Generation
Query Plan Guidance

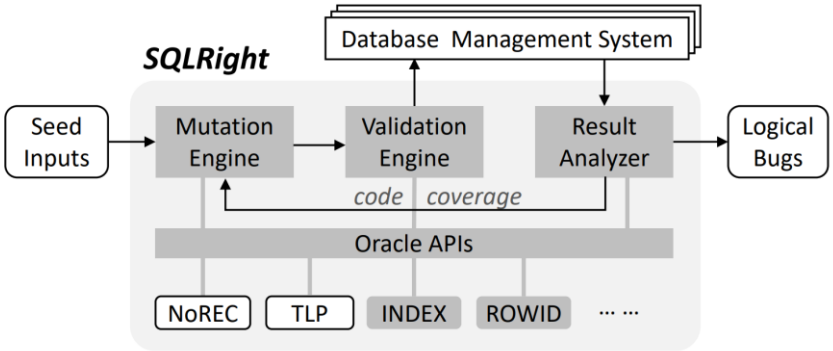
Random Generation Approaches

- Rule-based generation approaches
- Examples: SQLancer, SQLsmith
- No guidance

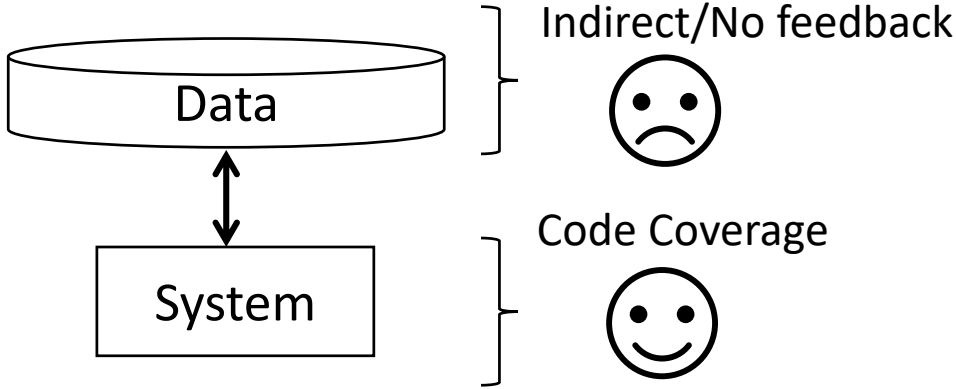


Coverage Guided Fuzzing

- Uses code coverage for guidance
- Example: AFL, SQLRight



Random Generation Approaches



“We found that this metric [code coverage] is not particularly useful for fuzzing DBMSs, since the core components of DBMS (e.g., query optimizer) already have high coverage (e.g., > 95%) after running tens of queries.”

APOLLO: Automatic Detection and Diagnosis of Performance Regressions in Database Systems, Jung et al.

Query Plan Guidance

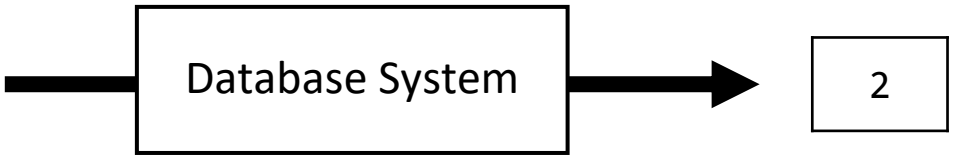
Query Plan Guidance (QPG) steers the test case generation process towards exploring diverse query plans

Query Plans

Query Plan

Bitmap Heap Scan on t0 (cost=10.74..31.37 rows=850 width=4)
Recheck Cond: (c0 > 1)
-> Bitmap Index Scan on i0 (cost=0.00..10.53 rows=850 width=0)
Index Cond: (c0 > 1)

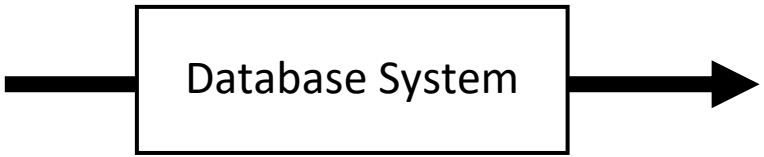
```
SELECT * FROM t0
WHERE c0 > 1;
```



Database systems translate a SQL query to a logical and then physical **execution plan**, which is then executed

Query Plans

```
EXPLAIN
SELECT * FROM t0
WHERE c0 > 1;
```



Query Plan

Bitmap Heap Scan on t0 (cost=10.74..31.37 rows=850 width=4)
Recheck Cond: (c0 > 1)
-> Bitmap Index Scan on i0 (cost=0.00..10.53 rows=850 width=0)
Index Cond: (c0 > 1)

Database systems readily expose query plans using an **EXPLAIN statement**

Query Plans

Rank			DBMS
Oct 2022	Sep 2022	Oct 2021	
1.	1.	1.	Oracle
2.	2.	2.	MySQL
3.	3.	3.	Microsoft SQL Server
4.	4.	4.	PostgreSQL
5.	5.	5.	IBM Db2
6.	6.	↑ 7.	Microsoft Access
7.	7.	↓ 6.	SQLite
8.	8.	8.	MariaDB
9.	9.	↑ 12.	Snowflake
10.	10.	10.	Microsoft Azure SQL Database

Most database systems expose query plans

https://db-engines.com/en/ranking/relational+dbms

Query Plans

```
CREATE TABLE t0(c0 INTEGER);

INSERT INTO t0(c0)
   VALUES (0), (1), (2);
SELECT * FROM t0 WHERE c0 > 1;
```

QUERY PLAN
Seq Scan on t0 (cost=0.00..41.88 rows=850 width=4)
Filter: (c0 > 1)

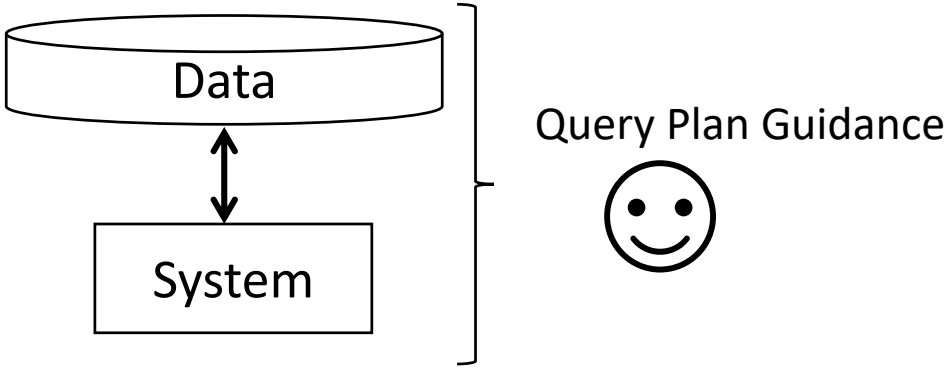
Query Plans

```
CREATE TABLE t0(c0 INTEGER);
CREATE INDEX i0 ON t0(c0);
INSERT INTO t0(c0)
   VALUES (0), (1), (2);
SELECT * FROM t0 WHERE c0 > 1;
```

Bitmap Heap Scan on t0 (cost=10.74..31.37 rows=850 width=4)
Recheck Cond: (c0 > 1)
-> Bitmap Index Scan on i0 (cost=0.00..10.53 rows=850 width=0)
Index Cond: (c0 > 1)

Query plans **provide rich information** on how tables are scanned, records filtered, tables joined, and which optimizations are performed

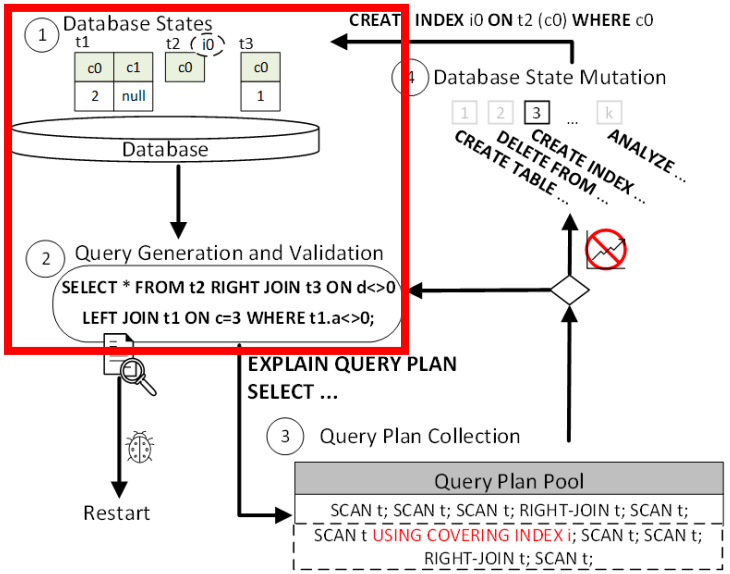
Query Plan Guidance



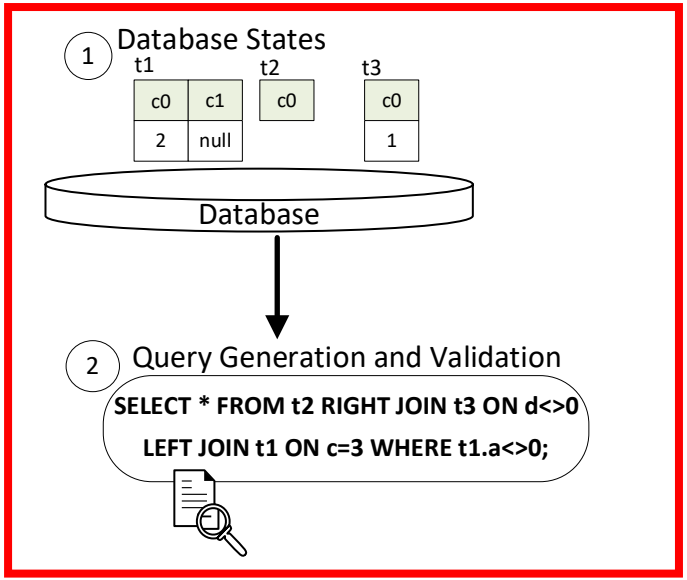
APOLLO: Automatic Detection and Diagnosis of Performance Regressions in Database Systems, Jung et al.

High-level Design

- Mutate databases rather than queries, allowing reuse of SQLancer's query generators
- When the rate of newly-seen query plans stagnates, mutate the database



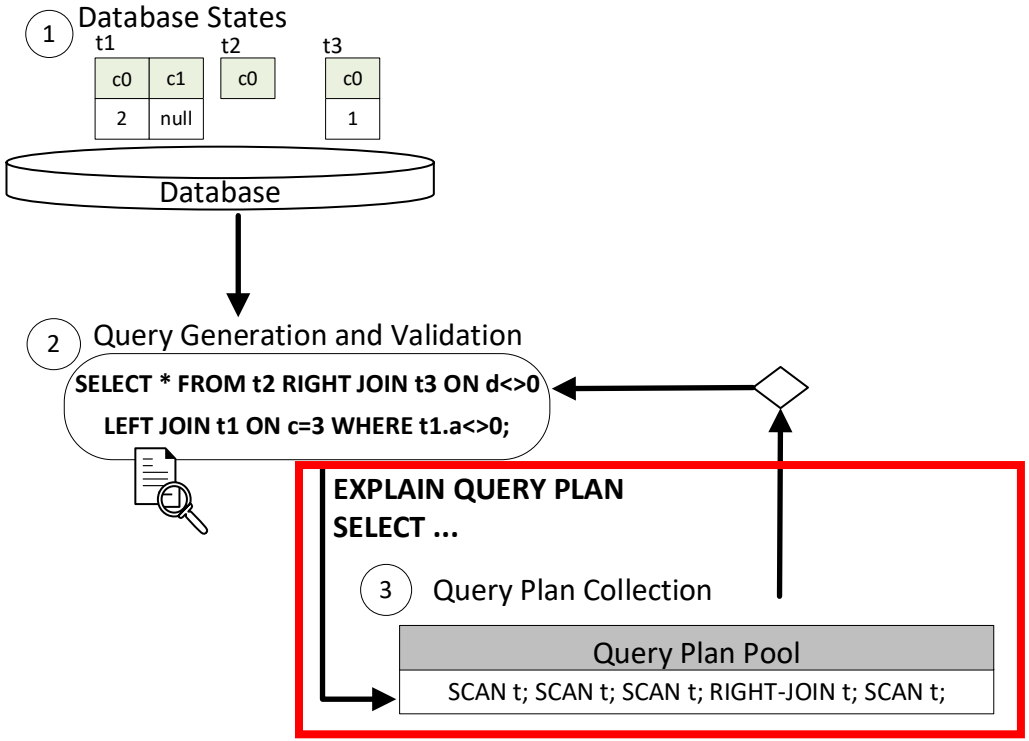
Initial Database and Query Validation



Re-use SQLancer's rule-based database and query generation approach

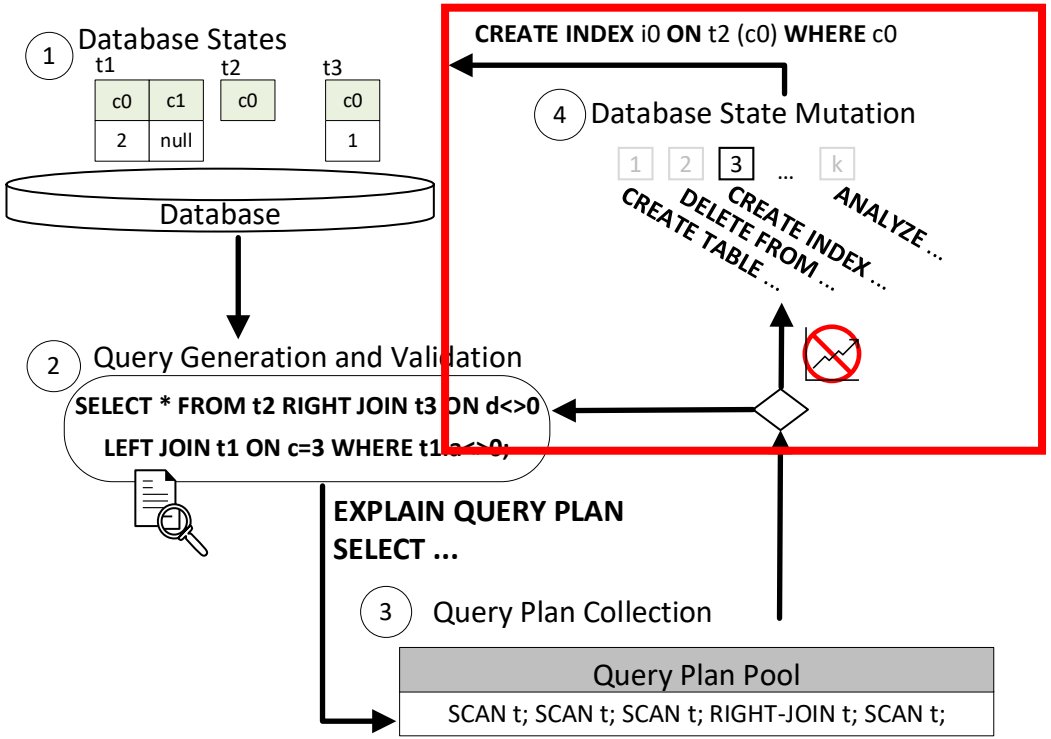
Query Plan Collection

Record newly seen query plans



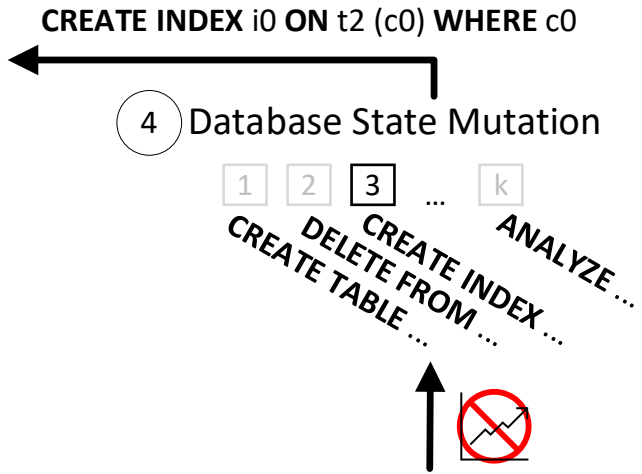
Database State Mutation

Mutate the database if no new query plans are observed



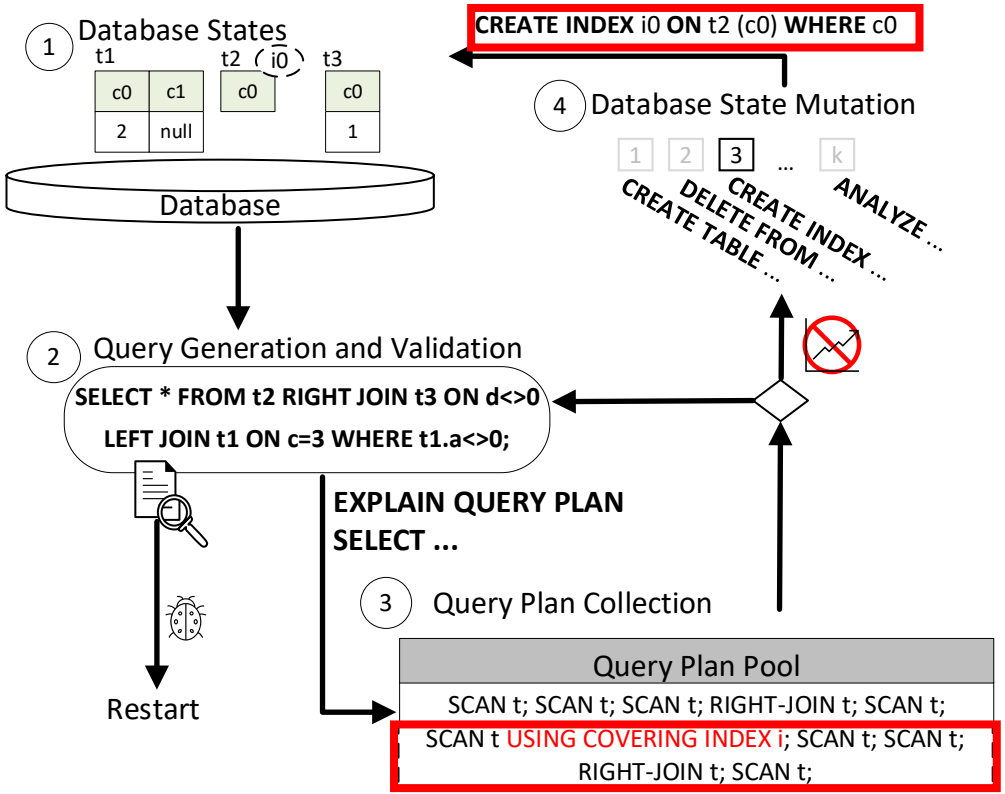
Database State Mutation

- Challenge: apply promising mutations that likely result in queries triggering new query plans
- Solution: model as a multi-armed bandit problem
  - With a fixed probability, we choose the mutator with the highest reward
  - Otherwise, randomly choose a mutator



Approach Overview

Even when executing the same query, we might observe new query plans



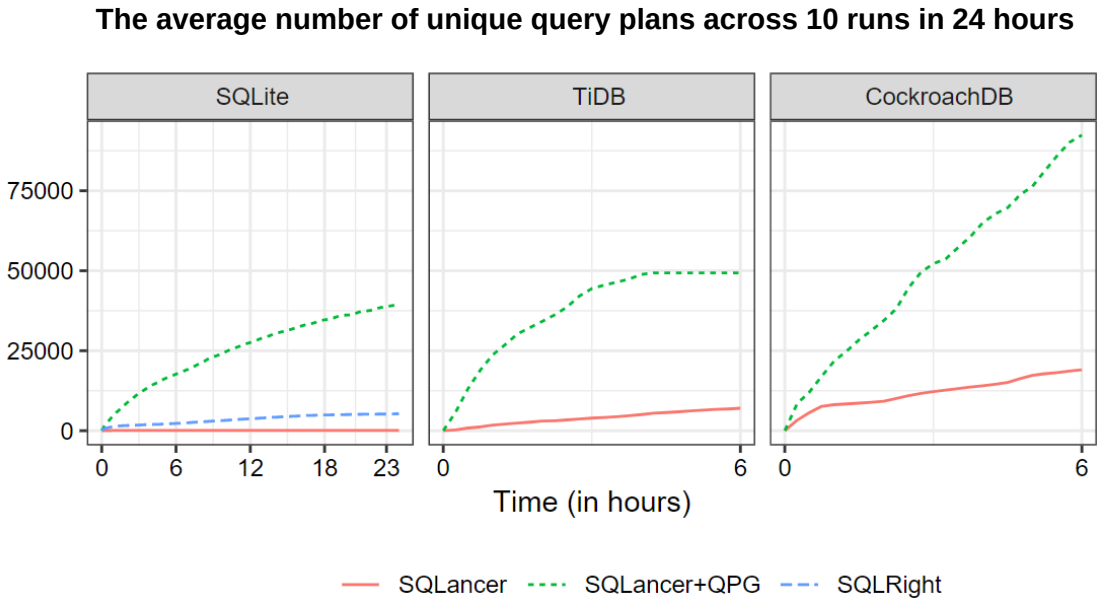
Evaluation: New Bugs

> (2) By Richard Hipp (drh) on 2022-07-15 12:59:59 in reply to 1 [link] [source]
>
> This bug goes back almost 8 years to check-in ddb5f0558c445699 on 2016-09-07, ve

DBMS	Crash	Error	Logic	All
SQLite	0	5	23	28
TiDB	2	4	3	9
CockroachDB	3	11	2	16
Sum:	**5**	**20**	**28**	**53**

Using QPG and existing test oracles like TLP, we found 53 unique, previously unknown bugs

Unique Query Plans Over Time



QPG exercises 4.85–408.48× more unique query plans than SQLancer, 7.46× more than SQLRight

SQLancer

- Automated testing tool for finding logic bugs in database systems
- Prototype platform
- Widely adopted by the industry

Today's talk will be about two approaches implemented in SQLancer

sqlancer / sqlancer Public

Detecting Logic Bugs in DBMS

www.sqlancer.com/

MIT license

1.1k stars 170 forks

Example: MySQL

```
SELECT * FROM t0, t1;
```

```
SELECT * FROM t0, t1 WHERE t0.c0=t1.c0
UNION ALL
SELECT * FROM t0, t1 WHERE NOT (t0.c0=t1.c0)
UNION ALL
SELECT * FROM t0, t1 WHERE (t0.c0=t1.c0) IS NULL;
```

ϕ

MySQL

t0.c0	t1.c0
0.0	-0.0

MySQL

t0.c0	t1.c0

≠

https://bugs.mysql.com/bug.php?id=99122

Approach Overview

Even when executing the same query, we might observe new query plans