Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

Gregory M. Kapfhammer\textsuperscript{1} & Phil McMinn\textsuperscript{2} & Chris J. Wright\textsuperscript{2}

\textsuperscript{1}Allegheny College, USA
\textsuperscript{2}University of Sheffield, UK

Sixth IEEE International Conference on Software Testing, Verification and Validation (ICST 2013)

Tuesday, March 19, 2013
Databases Are Everywhere!

Relational Database Management Systems
Databases Are Everywhere!

Relational Database Management Systems

PostgreSQL
Apache Derby
HyperSQL
IBM DB2
SQLite
MySQL
Databases Are Everywhere!

Deployment Locations for Databases
Databases Are Everywhere!

Deployment Locations for Databases

Database Application Server

PostgreSQL

Kapfhammer, McMinn, and Wright March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Databases Are Everywhere!

Deployment Locations for Databases

- Database Application Server
- Mobile Phone or Tablet

Database Management Systems:
- PostgreSQL
- SQLite
Databases Are Everywhere!

Deployment Locations for Databases

- Database Application Server
- Mobile Phone or Tablet
- Office and Productivity Software

PostgreSQL
SQLite
HyperSQL
Databases Are Everywhere!

Deployment Locations for Databases

- Database Application Server
- Mobile Phone or Tablet
- Office and Productivity Software
- PostgreSQL
- SQLite
- HyperSQL
- Government
Databases Are Everywhere!

Deployment Locations for Databases

- Database Application Server
- Mobile Phone or Tablet
- Office and Productivity Software

PostgreSQL

SQLite

HyperSQL

Government

Astrophysics

Kapfhammer, McMinn, and Wright
March 19, 2013
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

PostgreSQL

Relational Database Management System
Relational Database Schema

E-commerce

PostgreSQL

Relational Database Management System
Relational Database Schema

E-commerce

Relational Database Management System

PostgreSQL

Schema
Relational Database Schema
Relational Database Schema

Challenges

E-commerce

Relational Database Management System

PostgreSQL

Schema

State

Integrity Constraints

Kapfhammer, McMinn, and Wright

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

March 19, 2013
Relational Database Schema

PostgreSQL

E-commerce

Relational Database Management System

Schema

State

Integrity Constraints

PRIMARY KEY

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

Kapfhammer, McMinn, and Wright

March 19, 2013
Relational Database Schema

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

Kapfhammer, McMinn, and Wright

March 19, 2013
Relational Database Schema

E-commerce

Relational Components

Relational Database Management System

PostgreSQL

Schema

State

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

E-commerce

Relational Components

Tables

Schema

State

PostgreSQL

Relational Database Management System

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

E-commerce

Relational Components

Tables

Rows

Relational Database Management System

Schema

State

PostgreSQL
Relational Database Schema

E-commerce

Relational Components

- Tables
- Rows
- Columns

Relational Database Management System

- Schema
- State

PostgreSQL

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

PostgreSQL

Relational Database Management System

E-commerce

The Relational Schema is Working Correctly
Relational Database Schema

E-commerce

INSERT

Relational Database Management System

Schema

State

The Relational Schema is Working Correctly
Relational Database Schema

E-commerce

INSERT ✓

Relational Database Management System

Schema

State

The Relational Schema is Working Correctly

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

The Relational Schema is Working Correctly
Relational Database Schema

The Relational Schema is Working Correctly
Relational Database Schema
Relational Database Schema

INSERT ×

Relational Database Management System

Schema

State

The Relational Schema is Working Correctly
Relational Database Schema

E-commerce

Relational Database Management System

The Relational Schema is *Not* Working Correctly

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

E-commerce

The Relational Schema is *Not* Working Correctly
Relational Database Schema

E-commerce

The Relational Schema is *Not* Working Correctly

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

The Relational Schema is Not Working Correctly
Relational Database Schema

The Relational Schema is Not Working Correctly
Relational Database Schema

The Relational Schema is Not Working Correctly

E-commerce
Relational Database Schema

- E-commerce
- PostgreSQL
- The Relational Schema is *Not* Working Correctly

Kapfhammer, McMinn, and Wright
March 19, 2013
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Relational Database Schema

E-commerce

SELECT

The Relational Schema is *Not* Working Correctly

Relational Database Management System

PostgreSQL

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
The Relational Schema is Not Working Correctly
Relational Database Schema

Not working correctly!

E-commerce

SELECT ✔
RESULT ✗

The Relational Schema is Not Working Correctly
Need for Relational Schema Testing

The Data Warehouse Institute reports that North American organizations experience a $611 billion annual loss due to poor data quality.
Need for Relational Schema Testing

The Data Warehouse Institute reports that North American organizations experience a $611 billion annual loss due to poor data quality.

Scott W. Ambler argues that the “virtual absence” of database testing — the validation of the contents, schema, and functionality of the database — is the primary cause of this loss.
Need for Relational Schema Testing

The Data Warehouse Institute reports that North American organizations experience a $611 billion annual loss due to poor data quality.

Scott W. Ambler argues that the “virtual absence” of database testing — the validation of the contents, schema, and functionality of the database — is the primary cause of this loss.

This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas.
Defects in Relational Schemas

```
CREATE TABLE Flights (  
  FLIGHT_ID CHAR(6) NOT NULL,  
  SEGMENT_NUMBER INT NOT NULL,  
  ORIGINAL_AIRPORT CHAR(3),  
  DEPART_TIME TIME,  
  DEST_AIRPORT CHAR(3),  
  ARRIVE_TIME TIME,  
  MEAL CHAR(1),  
  PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),  
  CHECK (MEAL IN ('B', 'L', 'D', 'S'))  
);
```
Defects in Relational Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

The highlighted integrity constraints determine what data is valid.
Defects in Relational Schemas

CREATE TABLE Flights(
  FLIGHT_ID CHAR(6) NOT NULL,
  SEGMENT_NUMBER INT NOT NULL,
  ORIGINAL_AIRPORT CHAR(3),
  DEPART_TIME TIME,
  DEST_AIRPORT CHAR(3),
  ARRIVE_TIME TIME,
  MEAL CHAR(1),
  PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
  CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

The highlighted integrity constraints determine what data is valid.
Defects in Relational Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER),
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);

The highlighted integrity constraints determine what data is valid.
Defects in Relational Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME, 
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME, 
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

The highlighted integrity constraints determine what data is valid.
Defects in Relational Schemas

CREATE TABLE FlightAvailable (  
    FLIGHT_ID CHAR(6) NOT NULL,  
    SEGMENT_NUMBER INT NOT NULL,  
    FLIGHT_DATE DATE NOT NULL,  
    ECONOMY_SEATS_TAKEN INT,  
    BUSINESS_SEATS_TAKEN INT,  
    FIRSTCLASS_SEATS_TAKEN INT,  
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),  
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)  
    REFERENCES Flights (FLIGHT_ID, SEGMENT_NUMBER)  
);
Defects in Relational Schemas

CREATE TABLE FlightAvailable (  
    FLIGHT_ID        CHAR(6) NOT NULL,
    SEGMENT_NUMBER   INT     NOT NULL,
    FLIGHT_DATE      DATE    NOT NULL,
    ECONOMY_SEATS_TAKEN INT,
    BUSINESS_SEATS_TAKEN INT,
    FIRSTCLASS_SEATS_TAKEN INT,
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)
        REFERENCES Flights(FLIGHT_ID, SEGMENT_NUMBER)
);
Defects in Relational Schemas

CREATE TABLE FlightAvailable (  
    FLIGHT_ID CHAR(6) NOT NULL,  
    SEGMENT_NUMBER INT NOT NULL,  
    FLIGHT_DATE DATE NOT NULL,  
    ECONOMY_SEATS_TAKEN INT,  
    BUSINESS_SEATS_TAKEN INT,  
    FIRSTCLASS_SEATS_TAKEN INT,  
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),  
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)  
    REFERENCES Flights (FLIGHT_ID, SEGMENT_NUMBER)  
);  

The highlighted integrity constraints determine what data is valid
Defects in Relational Schemas

```
CREATE TABLE FlightAvailable (    
  FLIGHT_ID          CHAR(6)  NOT NULL,  
  SEGMENT_NUMBER     INT       NOT NULL,  
  FLIGHT_DATE        DATE      NOT NULL,  
  ECONOMY_SEATS_TAKEN INT,  
  BUSINESS_SEATS_TAKEN INT,  
  FIRSTCLASS_SEATS_TAKEN INT,  
  PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER),  
  FOREIGN KEY(FLIGHT_ID, SEGMENT_NUMBER) REFERENCES Flights(FLIGHT_ID, SEGMENT_NUMBER) 
);  
```

The highlighted integrity constraints determine what data is valid
Defects in Relational Schemas

Defect: The schema does not contain the correct primary key!
Defects in Relational Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);
Defects in Relational Schemas

```sql
CREATE TABLE FlightAvailable ( 
    FLIGHT_ID CHAR(6) NOT NULL, 
    SEGMENT_NUMBER INT NOT NULL, 
    FLIGHT_DATE DATE NOT NULL, 
    ECONOMY_SEATS_TAKEN INT, 
    BUSINESS_SEATS_TAKEN INT, 
    FIRSTCLASS_SEATS_TAKEN INT, 
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER), 
    FOREIGN KEY(FLIGHT_ID, SEGMENT_NUMBER) REFERENCES Flights(FLIGHT_ID, SEGMENT_NUMBER) 
); 
```

Defect: The schema does not contain the correct primary key!
Defects in Relational Schemas

CREATE TABLE FlightAvailable (  
    FLIGHT_ID               CHAR(6) NOT NULL,  
    SEGMENT_NUMBER          INT     NOT NULL,  
    FLIGHT_DATE             DATE    NOT NULL,  
    ECONOMY_SEATS_TAKEN     INT,  
    BUSINESS_SEATS_TAKEN    INT,  
    FIRSTCLASS_SEATS_TAKEN  INT,  
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),  
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)  
        REFERENCES Flights(FLIGHT_ID, SEGMENT_NUMBER)  
);  

Question: What kind of INSERT(s) will reveal this defect?
Defects in Relational Schemas

```
INSERT INTO Flights
VALUES ('UA20', 1, ... )
```

Question: What kind of **INSERT**(s) will reveal this defect?
Defects in Relational Schemas

Question: What kind of \texttt{INSERT}(s) will reveal this defect?

\begin{align*}
\text{INSERT INTO Flights} & \quad \text{VALUES ('UA20', 1, ... )} & \checkmark \\
\text{INSERT INTO Flights} & \quad \text{VALUES ('UA20', 2, ... )} & \times
\end{align*}
Defects in Relational Schemas

\[
\text{INSERT INTO Flights VALUES('UA20', 1, ... ) } \checkmark
\]
\[
\text{INSERT INTO Flights VALUES('UA20', 2, ... ) } \times
\]

Explanation: A flight with two different segments is no longer allowed!

Question: What kind of \text{INSERT}(s) will reveal this defect?
Defects in Relational Schemas

SchemaAnalyst automatically generates these `INSERT`s and this data!

```
INSERT INTO Flights
VALUES ('UA20', 1, ...) ✔
```

```
INSERT INTO Flights
VALUES ('UA20', 2, ...) ✗
```

Explanation: A flight with two different segments is no longer allowed!

Question: What kind of `INSERT`(s) will reveal this defect?
Search-Based Testing with *SchemaAnalyst*

![Diagram](image)

**Kapfhammer, McMinn, and Wright**

March 19, 2013

*Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems*
Search-Based Testing with *SchemaAnalyst*

Schema → Representation Generator → Test Suite Generator → Test Suites

Mutation Analysis → Test Suites → Mutants and Scores

**PostgreSQL**
Search-Based Testing with *SchemaAnalyst*

![Diagram](image)

**Test Data Generation**

**Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems**
Search-Based Testing with SchemaAnalyst

Test Data Generation

Test Suite Generator

Mutation Analysis

Test Suites

Mutants and Scores
Goals and Stages of Test Data Generation

Goal of test data generation?
Goals and Stages of Test Data Generation

Goal of test data generation?

```
INSERT INTO T1 VALUES (1, Jan-08-99, ... ) ✓
```
Goals and Stages of Test Data Generation

INSERT INTO $T_1$ VALUES (1, Jan-08-99, ... ) ✓

INSERT INTO $T_1$ VALUES (1, Jan-08-99, ... ) ✗
Goals and Stages of Test Data Generation

Goal of test data generation?

<table>
<thead>
<tr>
<th>Test Data Generation</th>
<th>Goals and Stages of Test Data Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT INTO $T_1$ VALUES (1, Jan-08-99, ... )</td>
<td>✓</td>
</tr>
<tr>
<td>INSERT INTO $T_1$ VALUES (1, Jan-08-99, ... )</td>
<td>×</td>
</tr>
<tr>
<td>INSERT INTO $T_n$ VALUES (true, 'L-20', ... )</td>
<td>✓</td>
</tr>
<tr>
<td>INSERT INTO $T_n$ VALUES (false, 'L-1', ... )</td>
<td>×</td>
</tr>
</tbody>
</table>
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER),
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID    CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME    TIME,
    DEST_AIRPORT  CHAR(3),
    ARRIVE_TIME   TIME,
    MEAL          CHAR(1),
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER),
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);

Stage 1: Generate rows of data to satisfy the integrity constraints
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID               CHAR(6) NOT NULL, 
    SEGMENT_NUMBER          INT NOT NULL, 
    ORIGINAL_AIRPORT       CHAR(3),
    DEPART_TIME            TIME, 
    DEST_AIRPORT           CHAR(3), 
    ARRIVE_TIME            TIME, 
    MEAL                   CHAR(1), 
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER), 
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);

Stage 1: Generate rows of data to satisfy the integrity constraints
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY_KEY(FLIGHT_ID, SEGMENT_NUMBER),
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);
Goals and Stages of Test Data Generation

Stage 2: Generate rows of data to negate a constraint
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID       CHAR(6) NOT NULL,
    SEGMENT_NUMBER  INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME     TIME,
    DEST_AIRPORT    CHAR(3),
    ARRIVE_TIME     TIME,
    MEAL            CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

Stage 2: Generate rows of data to negate a constraint
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

A fitness function computes a numeric value minimized by search.
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

Data’s fitness is closer to zero when nearer to a primary key value
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

Types, primary and foreign keys, UNIQUE, NOT NULL, and CHECK
Goals and Stages of Test Data Generation

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

See the paper for more details about the computation of fitness
## Test Data Generation

### Alternating Variable Method

\[ V_i \]
# Alternating Variable Method

$V_i$

$V_j$
Alternating Variable Method

$V_i$, $V_j$, $V_k$
Use the defaults to form the initial values of the \texttt{INSERT} variables
# Alternating Variable Method

Use exploratory moves to determine the correct direction for search

Kapfhammer, McMinn, and Wright  March 19, 2013
Alternating Variable Method

Use exploratory moves to determine the correct direction for search
Alternating Variable Method

Use exploratory moves to determine the correct direction for search
Alternating Variable Method

Use exploratory moves to determine the correct direction for search
Alternating Variable Method

Use exploratory moves to determine the correct direction for search
Alternating Variable Method

Use exploratory moves to determine the correct direction for search
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness

Kapfhammer, McMinn, and Wright
March 19, 2013
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Alternating Variable Method

AVM terminates when the fitness is zero or an exploration cycle fails
Alternating Variable Method

Use pattern moves to accelerate the improvements in fitness

AVM terminates when the fitness is zero or an exploration cycle fails

Restart AVM with random column values when an exploration cycle fails
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID       CHAR(6) NOT NULL,
    SEGMENT_NUMBER  INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME     TIME,
    DEST_AIRPORT    CHAR(3),
    ARRIVE_TIME     TIME,
    MEAL            CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

Use mutation analysis to assess the adequacy of INSERTs and values

Kapfhammer, McMinn, and Wright  March 19, 2013
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Mutation Operators for Schemas

CREATE TABLE Flights(
   FLIGHT_ID CHAR(6) NOT NULL,
   SEGMENT_NUMBER INT NOT NULL,
   ORIGINAL_AIRPORT CHAR(3),
   DEPART_TIME TIME,
   DEST_AIRPORT CHAR(3),
   ARRIVE_TIME TIME,
   MEAL CHAR(1),
   PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
   CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**Primary Keys**: Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

Primary Keys: Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**Primary Keys:** Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (ORIGINAL_AIRPORT, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**Primary Keys:** Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**Primary Keys:** Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER, DEST_AIRPORT),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**Primary Keys:** Remove, replace, and add column operators
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID       CHAR(6) NOT NULL,
    SEGMENT_NUMBER  INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME     TIME,
    DEST_AIRPORT    CHAR(3),
    ARRIVE_TIME     TIME,
    MEAL            CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
) ;

**UNIQUE**: Handle in a fashion similar to the primary key operator
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID    CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME     TIME,
    DEST_AIRPORT    CHAR(3),
    ARRIVE_TIME     TIME,
    MEAL            CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

**NOT NULL**: Reverse the status for all non-primary key columns
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID             CHAR(6) NOT NULL,
    SEGMENT_NUMBER        INT NOT NULL,
    ORIGINAL_AIRPORT     CHAR(3),
    DEPART_TIME          TIME,
    DEST_AIRPORT         CHAR(3),
    ARRIVE_TIME          TIME,
    MEAL                 CHAR(1),
    PRIMARY KEY(FLIGHT_ID, SEGMENT_NUMBER),
    CHECK(MEAL IN ('B', 'L', 'D', 'S'))
);

**NOT NULL**: Reverse the status for all non-primary key columns
Mutation Operators for Schemas

CREATE TABLE Flights(
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    ORIGINAL_AIRPORT CHAR(3) NOT NULL,
    DEPART_TIME TIME,
    DEST_AIRPORT CHAR(3),
    ARRIVE_TIME TIME,
    MEAL CHAR(1),
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
)
Mutation Operators for Schemas

CREATE TABLE Flights(
  FLIGHT_ID CHAR(6) NOT NULL,
  SEGMENT_NUMBER INT NOT NULL,
  ORIGINAL_AIRPORT CHAR(3),
  DEPART_TIME TIME,
  DEST_AIRPORT CHAR(3),
  ARRIVE_TIME TIME,
  MEAL CHAR(1),
  PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
  CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);

CHECK: Remove the constraint for each of the checked columns
Mutation Operators for Schemas

```
CREATE TABLE Flights(
    FLIGHT_ID    CHAR(6)    NOT NULL,
    SEGMENT_NUMBER INT    NOT NULL,
    ORIGINAL_AIRPORT CHAR(3),
    DEPART_TIME    TIME,
    DEST_AIRPORT   CHAR(3),
    ARRIVE_TIME    TIME,
    MEAL           CHAR(1),

    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    CHECK (MEAL IN ('B', 'L', 'D', 'S'))
);
```

**CHECK:** Remove the constraint for each of the checked columns
Mutation Operators for Schemas

```
CREATE TABLE FlightAvailable (
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    FLIGHT_DATE DATE NOT NULL,
    ECONOMY_SEATS_TAKEN INT,
    BUSINESS_SEATS_TAKEN INT,
    FIRSTCLASS_SEATS_TAKEN INT,
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)
        REFERENCES Flights(FLIGHT_ID, SEGMENT_NUMBER)
);  
```

Foreign Keys: Remove each column from the key
Mutation Operators for Schemas

CREATE TABLE FlightAvailable (  
  FLIGHT_ID CHAR(6) NOT NULL,  
  SEGMENT_NUMBER INT NOT NULL,  
  FLIGHT_DATE DATE NOT NULL,  
  ECONOMY_SEATS_TAKEN INT,  
  BUSINESS_SEATS_TAKEN INT,  
  FIRSTCLASS_SEATS_TAKEN INT,  
  PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),  
  FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)  
    REFERENCES Flights (FLIGHT_ID, SEGMENT_NUMBER)  
);  

**Foreign Keys:** Remove each column from the key
Mutation Operators for Schemas

CREATE TABLE FlightAvailable (
    FLIGHT_ID CHAR(6) NOT NULL,
    SEGMENT_NUMBER INT NOT NULL,
    FLIGHT_DATE DATE NOT NULL,
    ECONOMY_SEATS_TAKEN INT,
    BUSINESS_SEATS_TAKEN INT,
    FIRSTCLASS_SEATS_TAKEN INT,
    PRIMARY KEY (FLIGHT_ID, SEGMENT_NUMBER),
    FOREIGN KEY (FLIGHT_ID, SEGMENT_NUMBER)
        REFERENCES Flights (FLIGHT_ID, SEGMENT_NUMBER)
);
Relational Schema Mutation

Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Relational Schema Mutation

Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Relational Schema Mutation

Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]

Kapfhammer, McMinn, and Wright March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
Calculating the Mutation Score

\[ M_D = \frac{|K \cup Q|}{|K \cup N|} \]
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
# Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
### Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankAccount</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>BookTown</td>
<td>23</td>
<td>69</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>11</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Cloc</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CoffeeOrders</td>
<td>5</td>
<td>20</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>CustomerOrder</td>
<td>7</td>
<td>32</td>
<td>1</td>
<td>7</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>42</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>DellStore</td>
<td>8</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>Employee</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Examination</td>
<td>2</td>
<td>21</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Flights</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>FrenchTowns</td>
<td>3</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>8</td>
<td>23</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright  
March 19, 2013  
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
# Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>DellStore</td>
<td>8</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>Employee</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Examination</td>
<td>2</td>
<td>21</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Flights</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>FrenchTowns</td>
<td>3</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>8</td>
<td>23</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Iso3166</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>JWhoisServer</td>
<td>6</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>44</td>
<td>6</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>NistDML181</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>NistDML182</td>
<td>2</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Iso3166</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>JWhoisServer</td>
<td>6</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>44</td>
<td>6</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>NistDML181</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>NistDML182</td>
<td>2</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Iso3166</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>JWhoisServer</td>
<td>6</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>44</td>
<td>6</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>NistDML181</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>NistDML182</td>
<td>2</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>NistDML183</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NistWeather</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NistXTS748</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NistXTS749</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Person</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Residence</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>RiskIt</td>
<td>13</td>
<td>56</td>
<td>0</td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>8</td>
<td>32</td>
<td>0</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>Usda</td>
<td>10</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>
## Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Residence</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>RiskIt</td>
<td>13</td>
<td>56</td>
<td>0</td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>8</td>
<td>32</td>
<td>0</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>Usda</td>
<td>10</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright  
March 19, 2013  
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
# Case Study Schemas

<table>
<thead>
<tr>
<th>Schema</th>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Residence</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>RiskIt</td>
<td>13</td>
<td>56</td>
<td>0</td>
<td>10</td>
<td>15</td>
<td>11</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>8</td>
<td>32</td>
<td>0</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>Usda</td>
<td>10</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>
# Case Study Schemas

<table>
<thead>
<tr>
<th>Tables</th>
<th>Columns</th>
<th>Checks</th>
<th>Foreign keys</th>
<th>Not Nulls</th>
<th>Primary keys</th>
<th>Uniques</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td>111</td>
<td>542</td>
<td>27</td>
<td>40</td>
<td>231</td>
<td>68</td>
<td>11</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright
March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Data Generation Techniques

DBMonster
Data Generation Techniques

DBMonster

SchemaAnalyst

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Data Generation Techniques

- **DBMonster**
- **SchemaAnalyst**
  - HSQLDB ✓
  - SQLite ✓
  - Postgres ✓
Data Generation Techniques

- **DBMonster**
- HSQLDB ✗

- **Schema Analyst**
- HSQLDB ✓
- SQLite ✓
- Postgres ✓
Data Generation Techniques

- **DBMonster**
  - HSQLDB ✗
  - SQLite ✗

- **SchemaAnalyst**
  - HSQLDB ✓
  - SQLite ✓
  - Postgres ✓
# Data Generation Techniques

<table>
<thead>
<tr>
<th>Configuration</th>
<th>DBMonster</th>
<th>SchemaAnalyst</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSQLDB</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>SQLite</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Postgres</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>100.0</td>
<td>70.0</td>
</tr>
<tr>
<td>FrenchTowns</td>
<td>100.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Inventory</td>
<td>100.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Iso3166</td>
<td>100.0</td>
<td>50.0</td>
</tr>
<tr>
<td>JWhoisServer</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flights</td>
<td>100.0</td>
<td>70.0</td>
</tr>
<tr>
<td>FrenchTowns</td>
<td>100.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Inventory</td>
<td>100.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Iso3166</td>
<td>100.0</td>
<td>50.0</td>
</tr>
<tr>
<td>JWhoisServer</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
### Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NistDML181</td>
<td>100.0</td>
<td>75.0</td>
</tr>
<tr>
<td>NistDML182</td>
<td>100.0</td>
<td>50.0</td>
</tr>
<tr>
<td>NistDML183</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>NistXTS748</td>
<td>100.0</td>
<td>72.2</td>
</tr>
<tr>
<td>NistXTS749</td>
<td>100.0</td>
<td>21.4</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NistDML181</td>
<td>100.0</td>
<td>75.0</td>
</tr>
<tr>
<td>NistDML182</td>
<td>100.0</td>
<td>50.0</td>
</tr>
<tr>
<td>NistDML183</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>NistXTS748</td>
<td>100.0</td>
<td>72.2</td>
</tr>
<tr>
<td>NistXTS749</td>
<td>100.0</td>
<td>21.4</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NistDML181</td>
<td>100.0</td>
<td>75.0</td>
</tr>
<tr>
<td>NistDML182</td>
<td>100.0</td>
<td>50.0</td>
</tr>
<tr>
<td>NistDML183</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>NistXTS748</td>
<td>100.0</td>
<td>72.2</td>
</tr>
<tr>
<td>NistXTS749</td>
<td>100.0</td>
<td>21.4</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residence</td>
<td>100.0</td>
<td>62.5</td>
</tr>
<tr>
<td>RiskIt</td>
<td>100.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Products</td>
<td>96.4</td>
<td>59.3</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>97.8</td>
<td>59.3</td>
</tr>
<tr>
<td>Usda</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residence</td>
<td>100.0</td>
<td>62.5</td>
</tr>
<tr>
<td>RiskIt</td>
<td>100.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Products</td>
<td>96.4</td>
<td>59.3</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>97.8</td>
<td>59.3</td>
</tr>
<tr>
<td>Usda</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
## Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residence</td>
<td>100.0</td>
<td>62.5</td>
</tr>
<tr>
<td>RiskIt</td>
<td>100.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Products</td>
<td>96.4</td>
<td>59.3</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>97.8</td>
<td>59.3</td>
</tr>
<tr>
<td>Usda</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
### Constraint Coverage Results

<table>
<thead>
<tr>
<th>Schema</th>
<th>AVM (%)</th>
<th>DBMonster (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residence</td>
<td>100.0</td>
<td>62.5</td>
</tr>
<tr>
<td>RiskIt</td>
<td>100.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Products</td>
<td>96.4</td>
<td>59.3</td>
</tr>
<tr>
<td>UnixUsage</td>
<td>97.8</td>
<td>59.3</td>
</tr>
<tr>
<td>Usda</td>
<td>100.0</td>
<td>50.0</td>
</tr>
</tbody>
</table>
Quasi-Mutant Results

![Graph showing the number of mutants for different database systems]

Kapfhammer, McMinn, and Wright  March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Quasi-Mutant Results
Quasi-Mutant Results

![Bar chart showing the number of mutants for different database systems and testing techniques.]

- DellStore
  - Non-Quasi
  - Quasi

- SQLite
- Postgres
- Hsqldb

Kapfhammer, McMinn, and Wright March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Testing Technique</th>
<th>Empirical Study</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ ☐ ☐ ☐ ☐ ☐</td>
<td>☐ ☐ ☐ ☐ ☐ ☐</td>
<td>☐ ☐ ☐ ☐ ☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

**Results Analysis**

**Summary: Quasi-Mutant Results**

- HyperSQL
- SQLite
- PostgreSQL

Kapfhammer, McMinn, and Wright

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
### Summary: Quasi-Mutant Results

<table>
<thead>
<tr>
<th>Database Management System</th>
<th>HyperSQL</th>
<th>SQLite</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Some</td>
<td>None</td>
<td>Some</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

March 19, 2013
Summary: Quasi-Mutant Results

HyperSQL: Some

SQLite: None

PostgreSQL: Some

Few quasi-mutants means that the mutation scores are good effectiveness indicators.
Mutation Score Results

DBMonster

SchemaAnalyst
Mutation Score Results

**DBMonster**

JWhoisServer

DBI=300, \( M_D = 0.2 \)

**SchemaAnalyst**

DBI=62, \( M_D = 0.7 \)
Mutation Score Results

**DBMonster**

JWhoisServer  \( DBI=300, M_D = 0.2 \)

NistDML181  \( DBI=13,650, M_D = 0.5 \)

**SchemaAnalyst**

DBI=62,  \( M_D = 0.7 \)

DBI=7,  \( M_D = 0.6 \)
Mutation Score Results

DBMonster

(0.0, 0.11, 0.41, 0.52, 0.68)

SchemaAnalyst

(0.29, 0.59, 0.65, 0.70, 0.89)
Mutation Score Results

*DBMonster* crashes for six schemas!

CustomerOrder
Flights
NistDML182
NistXTS748
Person
RiskIt

*DBMonster*

(0.0, 0.11, 0.41, 0.52, 0.68)

*SchemaAnalyst*

(0.29, 0.59, 0.65, 0.70, 0.89)
Mutation Score Results

*DBMonster* crashes for six schemas!

CustomerOrder
Flights
NistDML182
NistXTS748
Person
RiskIt

*DBMonster*

(0.0, 0.11, 0.41, 0.52, 0.68)

*SchemaAnalyst*

(0.29, 0.59, 0.65, 0.70, 0.89)
Mutation Score Results

DBMonster crashes for six schemas!

CustomerOrder
Flights
NistDML182
NistXTS748
Person
RiskIt

DBMonster

(0.0, 0.11, 0.41, 0.52, 0.68)

SchemaAnalyst

(0.29, 0.59, 0.65, 0.70, 0.89)
## Mutation Score Results

<table>
<thead>
<tr>
<th>DBMonster</th>
<th>SchemaAnalyst</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.0, 0.11, 0.41, 0.52, 0.68)</td>
<td>(0.29, 0.59, 0.65, 0.70, 0.89)</td>
</tr>
</tbody>
</table>
Mutation Score Results

DBMonster
(0.0, 0.11, 0.41, 0.52, 0.68)

SchemaAnalyst
(0.29, 0.59, 0.65, 0.70, 0.89)
Mutation Score Results

`SchemaAnalyst`'s mutation score is higher than `DBMonster`'s for 96% of the schemas:

- `SchemaAnalyst` (0.29, 0.59, 0.65, 0.70, 0.89)
- `DBMonster` (0.0, 0.11, 0.41, 0.52, 0.68)

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
### Efficiency Results

<table>
<thead>
<tr>
<th>DBMonster</th>
<th>SchemaAnalyst</th>
</tr>
</thead>
</table>

Kapfhammer, McMinn, and Wright

March 19, 2013

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
# Efficiency Results

<table>
<thead>
<tr>
<th>Testing Technique</th>
<th>Empirical Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMonster</td>
<td>(0.41, 1.09, 1.90, 5.07, 36.52)</td>
</tr>
<tr>
<td>SchemaAnalyst</td>
<td>(1.50, 3.01, 5.21, 16.79, 639.93)</td>
</tr>
</tbody>
</table>

Kapfhammer, McMinn, and Wright

Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

March 19, 2013
Efficiency Results

DBMonster

(1.50, 3.01, 5.21, 16.79, 639.93)

SchemaAnalyst

(0.41, 1.09, 1.90, 5.07, 36.52)
Efficiency Results

*SchemaAnalyst* exhibits competitive data generation times that are less variable than *DBMonster*. 

\[(1.50, 3.01, 5.21, 16.79, 639.93)\] for *DBMonster* and \[(0.41, 1.09, 1.90, 5.07, 36.52)\] for *SchemaAnalyst*. 

Kapfhammer, McMinn, and Wright  
March 19, 2013  
Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems
Important Contributions

This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas.
Important Contributions

This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas.

The empirical study demonstrates that *SchemaAnalyst’s* efficiency is competitive with *DBMonster’s*.
Important Contributions

This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas. The empirical study demonstrates that *SchemaAnalyst*’s efficiency is competitive with *DBMonster*’s. *SchemaAnalyst* almost always covers 100% of the constraints in the 25 chosen relational schemas.
Important Contributions

This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas.

The empirical study demonstrates that *SchemaAnalyst*’s efficiency is competitive with *DBMonster*’s.

*SchemaAnalyst* almost always covers 100% of the constraints in the 25 chosen relational schemas.

*SchemaAnalyst*’s mutation score is higher than *DBMonster*’s for 96% of the schemas.
This paper presents *SchemaAnalyst*, a search-based system for testing the complex integrity constraints in relational schemas.

The empirical study demonstrates that *SchemaAnalyst*’s efficiency is competitive with *DBMonster*’s.

*SchemaAnalyst* almost always covers 100% of the constraints in the 25 chosen relational schemas.

*SchemaAnalyst*’s mutation score is higher than *DBMonster*’s for 96% of the schemas.

http://www.schemaanalyst.org