SchemaAnalyst
Search-based Testing for Relational Database Schemas

Cody Kinneer
Institute for Software Research
Carnegie Mellon University

Oct 6, 2016
Relational Databases

Databases are everywhere!
Databases are everywhere!
Relational Databases

Databases are everywhere!

Database Application Server

Mobile Phone or Tablet

PostgreSQL

SQLite
Relational Databases

Databases are everywhere!

Database Application Server

Mobile Phone or Tablet

Office and Productivity Software
Relational Databases

Databases are everywhere!

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

Government
Relational Databases

Databases are everywhere!

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

Government

Astrophysics

PostgreSQL

SQLite

HyperSQL
Relational Databases

Databases are everywhere!

Database Application Server

Mobile Phone or Tablet

Office and Productivity Software

Government

Astrophysics

Over 1,000,000 posts!
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>CREATE TABLE DEPT_INFO (</code></td>
</tr>
<tr>
<td>2</td>
<td><code>DEPT_ID INTEGER NOT NULL,</code></td>
</tr>
<tr>
<td>3</td>
<td><code>DEPT_NAME VARCHAR(50),</code></td>
</tr>
<tr>
<td>4</td>
<td><code>PRIMARY KEY (DEPT_ID)</code></td>
</tr>
<tr>
<td>5</td>
<td><code>);</code></td>
</tr>
<tr>
<td>6</td>
<td><code>CREATE TABLE OFFICE_INFO (</code></td>
</tr>
<tr>
<td>7</td>
<td><code>OFFICE_ID INTEGER NOT NULL,</code></td>
</tr>
<tr>
<td>8</td>
<td><code>OFFICE_NAME VARCHAR(50),</code></td>
</tr>
<tr>
<td>9</td>
<td><code>HAS_PRINTER SMALLINT,</code></td>
</tr>
<tr>
<td>10</td>
<td><code>PRIMARY KEY (OFFICE_ID)</code></td>
</tr>
<tr>
<td>11</td>
<td><code>);</code></td>
</tr>
</tbody>
</table>

Figure: A sample of the UnixUsage schema.
Database Schemas

Figure: A sample of the UnixUsage schema.
```
CREATE TABLE DEPT_INFO (  
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);

CREATE TABLE OFFICE_INFO (  
    OFFICE_ID INTEGER NOT NULL,
    OFFICE_NAME VARCHAR(50),
    HAS_PRINTER SMALLINT,
    PRIMARY KEY (OFFICE_ID)
);
```

Figure: A sample of the UnixUsage schema.
Testing Database Schemas

Manual testing is onerous and error prone
Testing Database Schemas

Manual testing is onerous and error prone

DBMonster only supports one DMBS
Testing Database Schemas

- Manual testing is onerous and error prone
- DBMonster only supports one DMBS
- Crashes and poor constraint coverage
Testing Database Schemas

Manual testing is onerous and error prone

DBMonster only supports one DMBS

Crashes and poor constraint coverage

Schemas often not tested at all!
SchemaAnalyst
SchemaAnalyst

Coverage Criterion
SchemaAnalyst

Coverage Criterion

Data Generator
SchemaAnalyst

Coverage Criterion

Data Generator

Database Schema
Schema Analyst

Coverage Criterion

Data Generator

Database Schema

JUnit Test Suite
SchemaAnalyst

- Coverage Criterion
- Data Generator
- Database Schema

JUnit Test Suite

Extensible tool for test data generation
Search-Based Testing

X: Parameter 1 Value

Y: Parameter 2 Value

Z: Fitness
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.
Automatically Testing Schemas

CREATE TABLE DEPT_INFO (DEPT_ID INTEGER NOT NULL, DEPT_NAME VARCHAR(50), PRIMARY KEY (DEPT_ID));

Figure: A sample of the UnixUsage schema.

INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

CREATE TABLE DEPT_INFO (
  DEPT_ID INTEGER NOT NULL,
  DEPT_NAME VARCHAR(50),
  PRIMARY KEY (DEPT_ID)
);

INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');

Figure: A sample of the UnixUsage schema.

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```sql
CREATE TABLE DEPT_INFO (  
    DEPT_ID INTEGER NOT NULL,  
    DEPT_NAME VARCHAR(50),  
    PRIMARY KEY (DEPT_ID)  
);
```

Figure: Data generated by SchemaAnalyst.

```sql
INSERT INTO DEPT_INFO  
VALUES (0, '');
```

```sql
INSERT INTO DEPT_INFO  
VALUES (NULL, '');
```
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```sql
CREATE TABLE DEPT_INFO (
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: Data generated by SchemaAnalyst.

```sql
1) INSERT INTO DEPT_INFO VALUES (0, '');
2) INSERT INTO DEPT_INFO VALUES (NULL, ' ');
```
Automatically Testing Schemas

CREATE TABLE DEPT_INFO (
DEPT_ID INTEGER NOT NULL,
DEPT_NAME VARCHAR(50),
PRIMARY KEY (DEPT_ID)
);

Figure: A sample of the UnixUsage schema.

INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```sql
CREATE TABLE DEPT_INFO ( 
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: Data generated by SchemaAnalyst.

```sql
INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');
```
Automatically Testing Schemas

```sql
CREATE TABLE DEPT_INFO (
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: A sample of the UnixUsage schema.

```sql
INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');
```

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

```sql
CREATE TABLE DEPT_INFO (  
  DEPT_ID INTEGER NOT NULL,  
  DEPT_NAME VARCHAR(50),  
  PRIMARY KEY (DEPT_ID)  
);
```

Figure: A sample of the UnixUsage schema.

```sql
INSERT INTO DEPT_INFO  
VALUES (0, '');
```

```sql
INSERT INTO DEPT_INFO  
VALUES (NULL, '');
```

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

```sql
CREATE TABLE DEPT_INFO (
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: A sample of the UnixUsage schema.

```sql
INSERT INTO DEPT_INFO VALUES (0, '');
INSERT INTO DEPT_INFO VALUES (NULL, '');
```

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

```sql
CREATE TABLE DEPT_INFO ( 
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
); 
```

Figure: A sample of the UnixUsage schema.

```sql
INSERT INTO DEPT_INFO VALUES (0, '');
```

```sql
INSERT INTO DEPT_INFO VALUES (NULL, '');
```

Figure: Data generated by SchemaAnalyst.
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```
CREATE TABLE DEPT_INFO (
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: Data generated by SchemaAnalyst.

```
1 INSERT INTO DEPT_INFO VALUES (0, '');
2 INSERT INTO DEPT_INFO VALUES (NULL, '');
```
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```sql
CREATE TABLE DEPT_INFO (
  DEPT_ID INTEGER NOT NULL,
  DEPT_NAME VARCHAR(50),
  PRIMARY KEY (DEPT_ID)
);
```

Figure: Data generated by SchemaAnalyst.

```sql
INSERT INTO DEPT_INFO
VALUES (0, '');
```

```sql
INSERT INTO DEPT_INFO
VALUES (NULL, '');
```
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```sql
CREATE TABLE DEPT_INFO (  
  DEPT_ID INTEGER NOT NULL,  
  DEPT_NAME VARCHAR(50),  
  PRIMARY KEY (DEPT_ID)  
);  
```

Figure: Data generated by SchemaAnalyst.

```sql
INSERT INTO DEPT_INFO  
VALUES (0, '');  
INSERT INTO DEPT_INFO  
VALUES (NULL, '');  
```
Automatically Testing Schemas

Figure: A sample of the UnixUsage schema.

```
CREATE TABLE DEPT_INFO (
    DEPT_ID INTEGER NOT NULL,
    DEPT_NAME VARCHAR(50),
    PRIMARY KEY (DEPT_ID)
);
```

Figure: Data generated by SchemaAnalyst.

```
1  INSERT INTO DEPT_INFO VALUES (0, '');
2  INSERT INTO DEPT_INFO VALUES (NULL, '');
```
Real-world Ready

Schemas from Firefox and StackOverflow
Real-world Ready

Schemas from Firefox and StackOverflow

Scales to 1,000s of tables and constraints
Real-world Ready

Schemas from Firefox and StackOverflow

Scales to 1,000s of tables and constraints

Extensive documentation available on GitHub
Real-world Ready

Schemas from Firefox and StackOverflow

Scales to 1,000s of tables and constraints

Extensive documentation available on GitHub

*SchemaAnalyst* provides an efficient means of generating test data for real-world database applications
Usage

Tool Demo
Key Contributions

*SchemaAnalyst*: an open-source test data generator for relational database schemas
Key Contributions

**SchemaAnalyst**: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems
Key Contributions

*SchemaAnalyst*: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation supporting the use and modification of the tool
Key Contributions

*SchemaAnalyst*: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation supporting the use and modification of the tool

Enhance the testing of database systems in industry and enable future research!
Key Contributions

*SchemaAnalyst*: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation supporting the use and modification of the tool

Enhance the testing of database systems in industry and enable future research!

https://github.com/schemaanalyst-team/schemaanalyst