Verifying Data Interaction Coverage to Improve Testing of Data-intensive Systems

Fabio Biselli
fabio.biselli@studio.unibo.it

Introduction

- Data-intensive software systems are increasingly prominent in driving global processes such as e-governance, data curation for scientific and medical research, and social networking.
- Example: the Directorate of the Norwegian Custom and Excise uses TVINN system to process about 30,000 customs declarations a day.
- The live transaction stream of declarations is processed for conformance to well-formedness rules, laws and regulations by complex batch applications.
- The typical process to rapidly and effectively test batch applications on data-intensive systems involves usage of input **test databases** regularly copied from the live transaction processing stream.

Test Databases

- Using test databases is based on the assumption that data from live transactions represents realistic scenarios.
- The realistic scenarios correspond to patterns found in test databases that are expected to either demonstrate correctness of application functionality or uncover bugs in the way transactions were processed by batch applications.
- Despite a high practical reliance on such test databases there exists very few systematic approaches to verify their adequacy for testing.

Existing Approaches

- **Create and execute complex queries.** This approach is often:
  - Tedious and error-prone;
  - Databases technology specific;
  - Much needed for documentation and hard to maintain.
- **Semi-systematic approach.** Use spreadsheets as a structured checklist for properties to be found.
  - Documenting testing intentions in a spreadsheet does not associate clear computable meaning to the tests making them ambiguous.
- **Synthesis of databases.** Automatically generate databases.
  - Effective approach to test data-intensive systems under development or freshly deployed, not for long-running systems.

Contribution

- To improve testing authors propose to model and verify realistic scenarios as data interactions.
- Data interactions are modeled using the classification tree modeling formalism.
- The classification tree primarily models a domain of data interactions possible between fields of a test database.

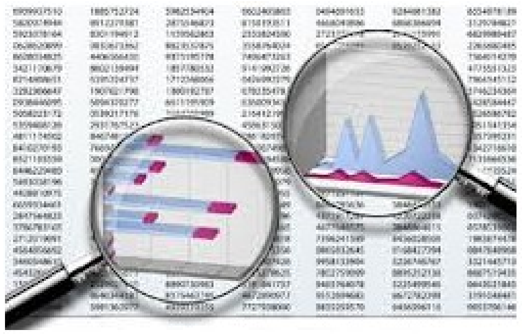
Representing test cases as data interactions is based on the idea that different fields cross-cutting a relational database schema have classes that interact or are inter-dependent (due to operations in a black-box application) and either:

(a) must exist together in the database for a fixed number of times

(b) not exist together at all.

Verification of Data Interactions

- How can we verify if these test cases representing data interactions are in fact covered in a test database?
- The principal contribution of the paper is a human-in-the-loop method implemented in a tool DEPICT to answer this question.



Certus

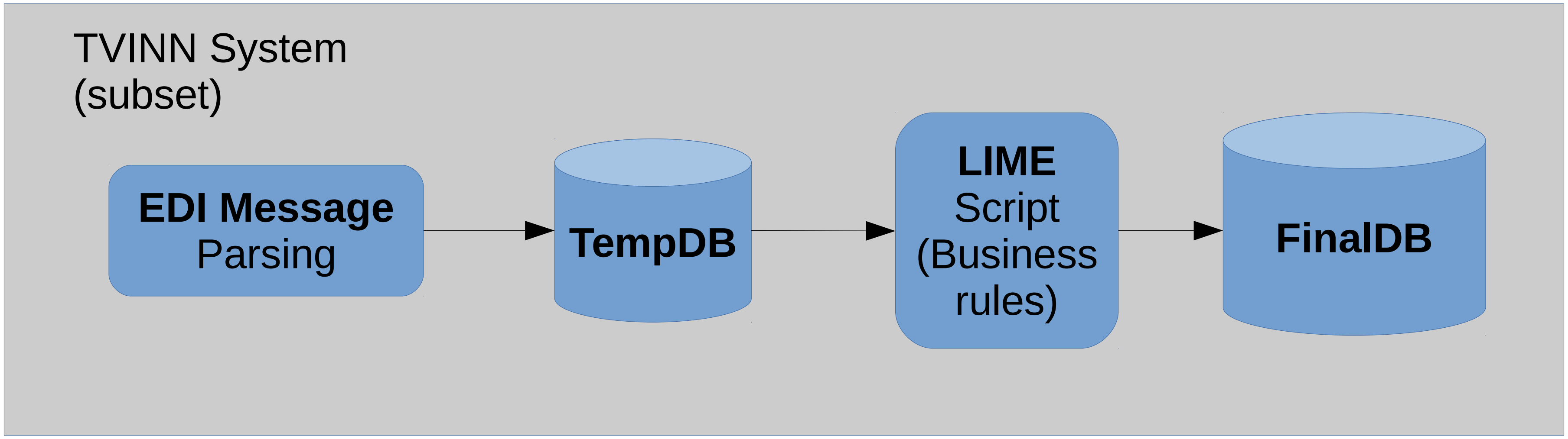
sfi Centre for Research-based Innovation

Established by the Research Council of Norway

DEPICT

DiscovEring Patterns and InteraCTions in databases

Case Study: Norwegian Customs



The principal challenge is to verify that the TEMPDB and FINALDB databases have correctly executed the checks.

The challenge is evergreen since checks in TVINN evolve on a regular basis (approximately, every six months), depending on new governmental policies, sanctions, and change in political parties.

Testing TVINN

Testing TVINN has been achieved by a small testing staff who manually execute the tests and verifies the results.

However, the current practice of using such a test database presents three crucial problems:

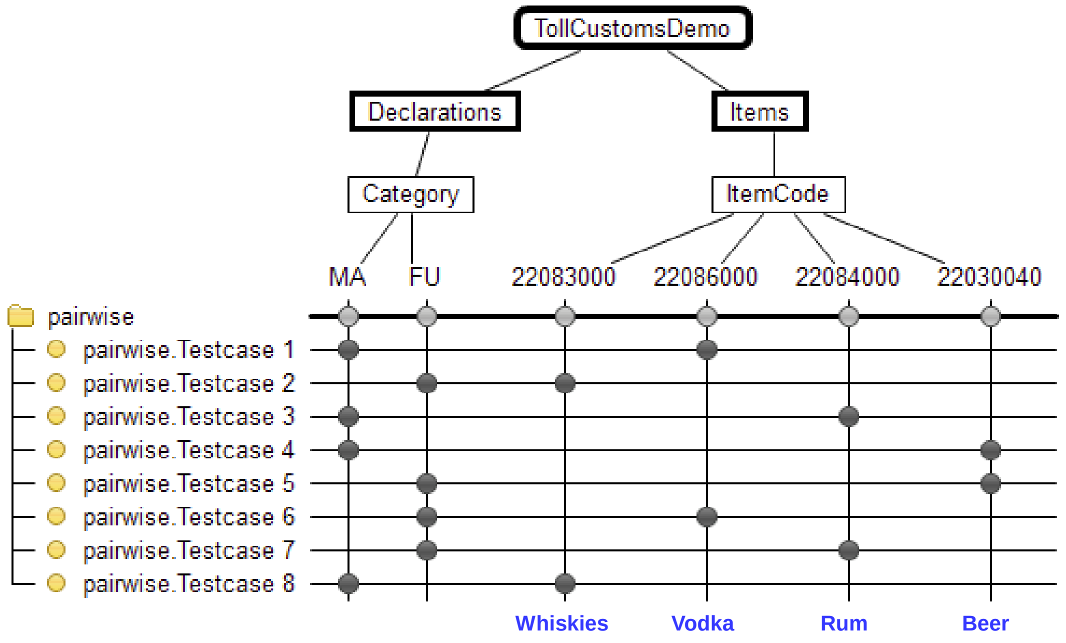
- No Coverage Guarantee.
- Very Large Set of Test Records.
- Constantly Changing Rules.

Modelling Data Interactions

- Modelling data interactions was introduced by the authors in "*Modelling data interaction requirements: A position paper. In Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on, pages 50–54, July 2013*".
- Data interactions are modeled on a relational database schema.
- The classification tree modeling formalism is used to model data interactions.

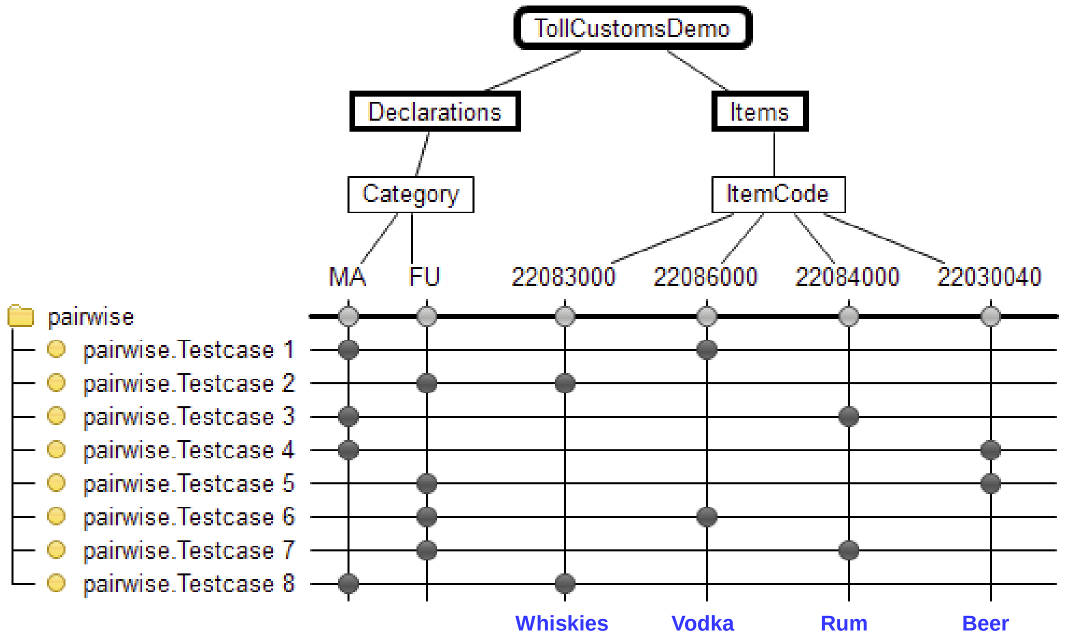
Classification Tree Model

- Classification tree models are typically used to model the input domain or a subset of it for a software system.
- They contain the concepts of compositions, classifications, and classes for each classification to structure the input domain.



Classification Tree Model

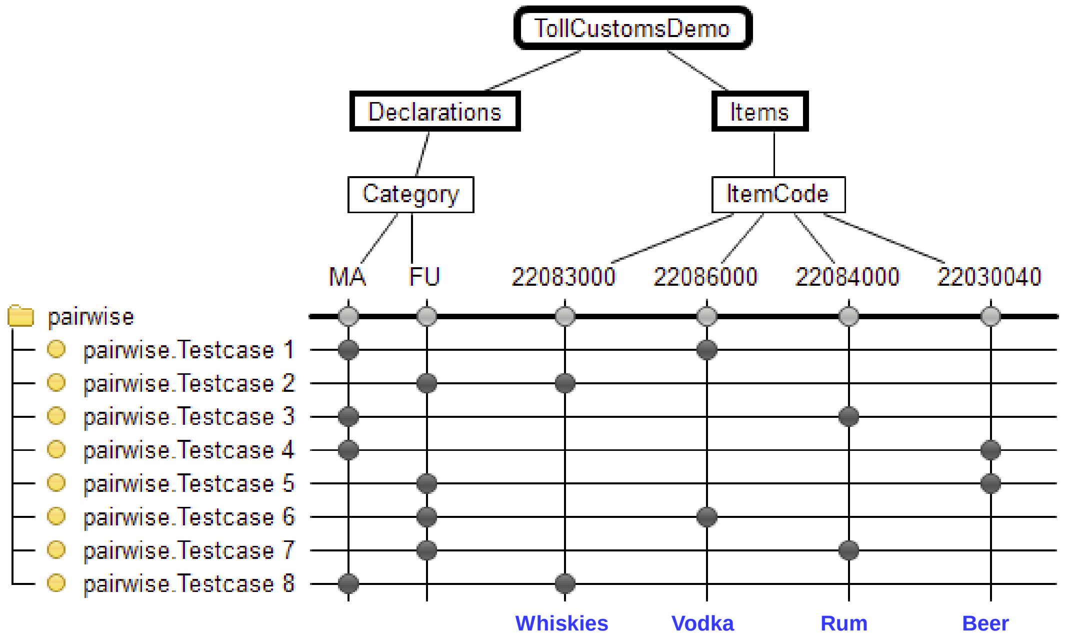
- **Root Composition for Database**: It is an identifier for a database on a server. Software that analyzes the classification tree model can identify a concrete database using this identifier.
- **Compositions for Tables**: The root composition can contain several compositions representing identifiers for tables.



Modelling Data Interactions

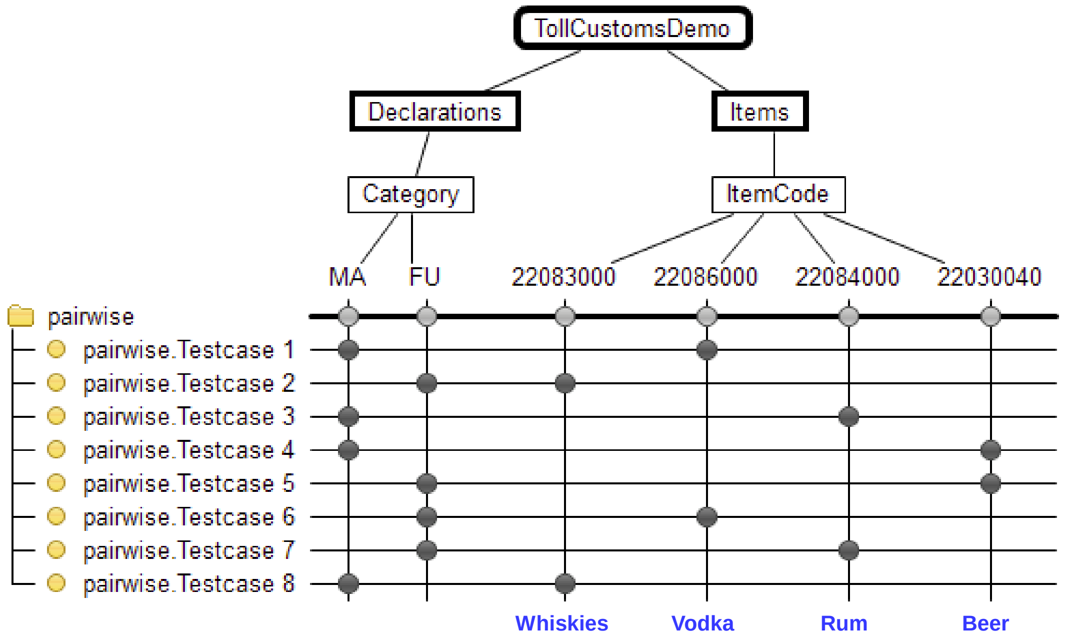
Classification Tree Model

- **Classifications for Fields**: Fields in tables are classifications in the third level of the classification tree. All or only a subset of fields for a table might be specified in the model.
- **Classes for Fields**: A field can have one or more concrete values or domains of values.



Classification Tree Model

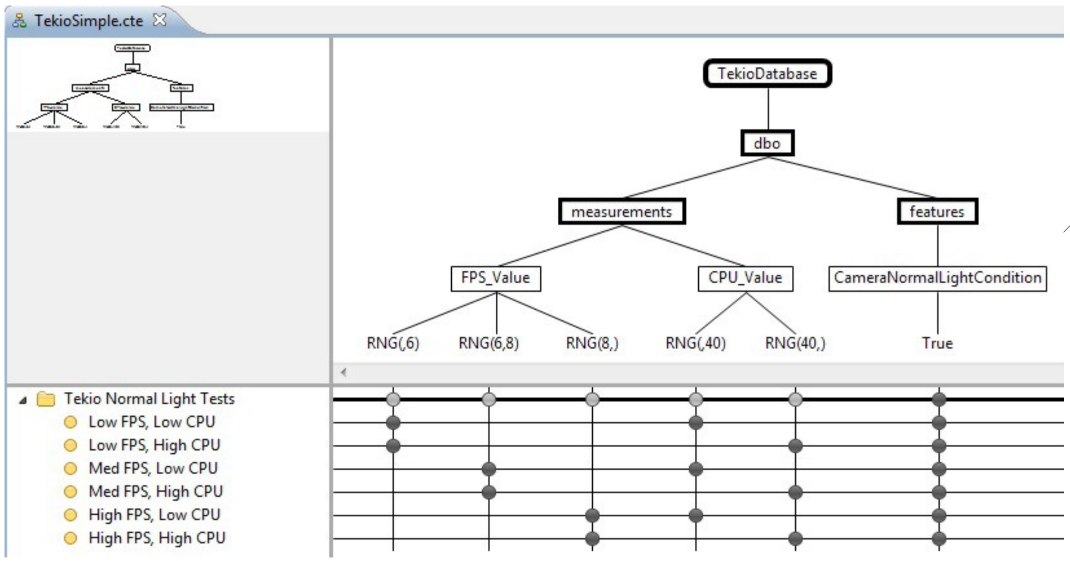
- **Interactions as Test Cases in Groups**: Interactions between field values across different tables are represented as test cases in a classification tree model.

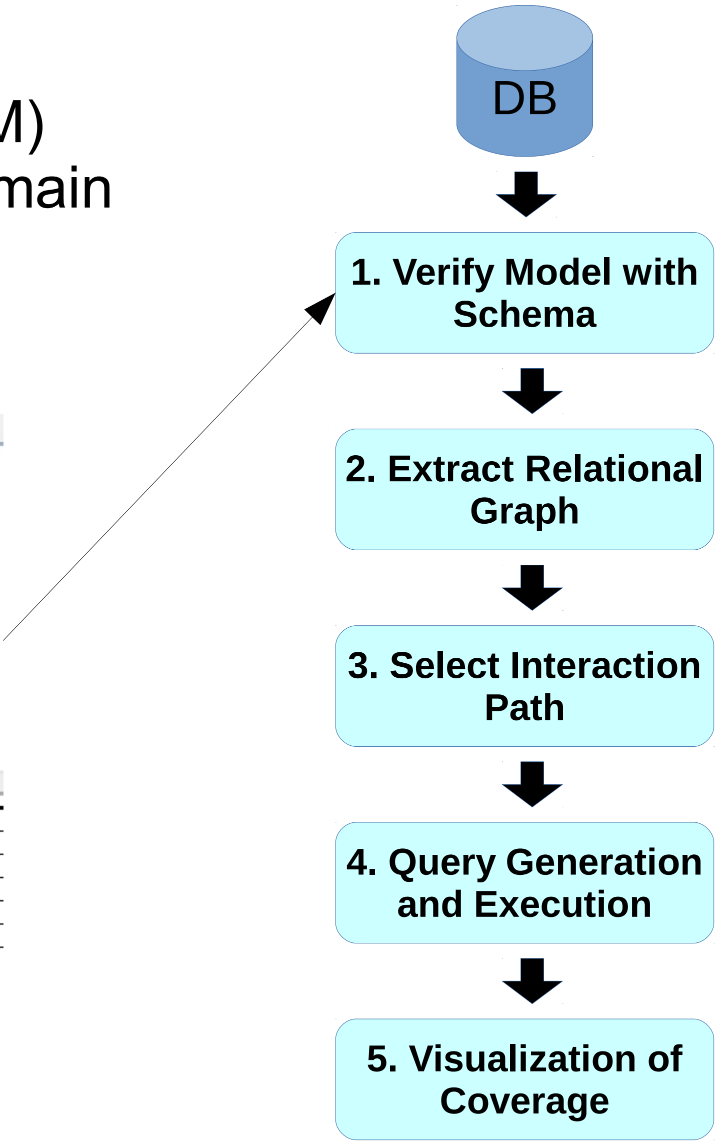


Modelling Data Interactions

Method

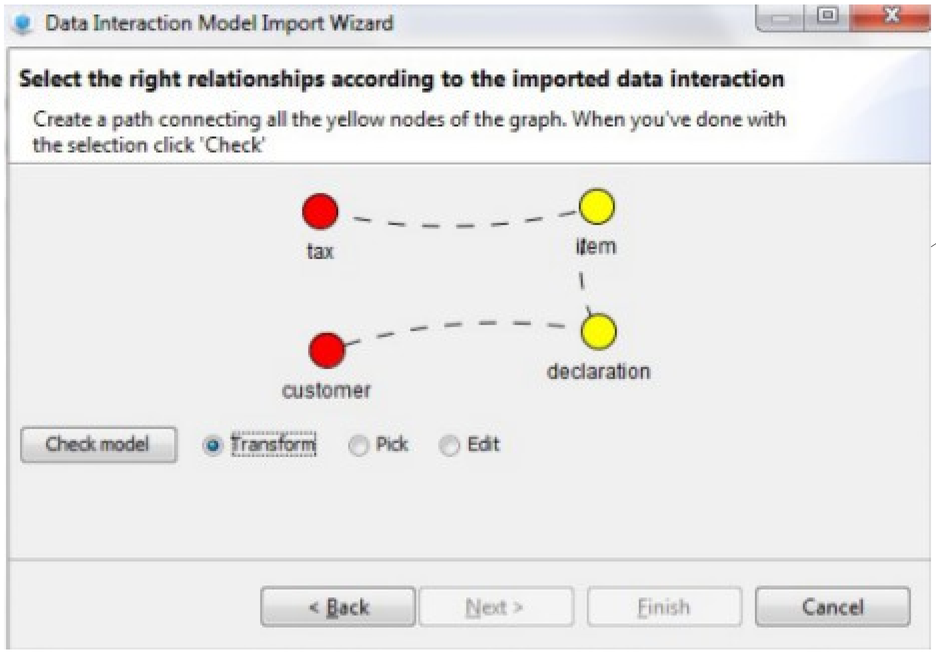
- The input classification tree model (CTM) with test cases is first specified by a domain expert.




DB
1. Verify Model with Schema
2. Extract Relational Graph
3. Select Interaction Path
4. Query Generation and Execution
5. Visualization of Coverage

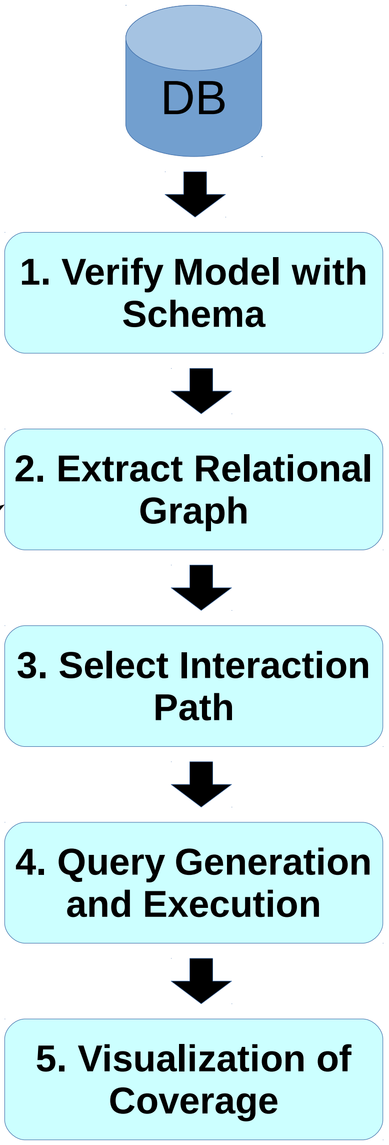
Method

- DEPICT extracts a relational graph by querying the meta-data of the database schema.
- One must manually select an edge to establish a relationship between nodes.



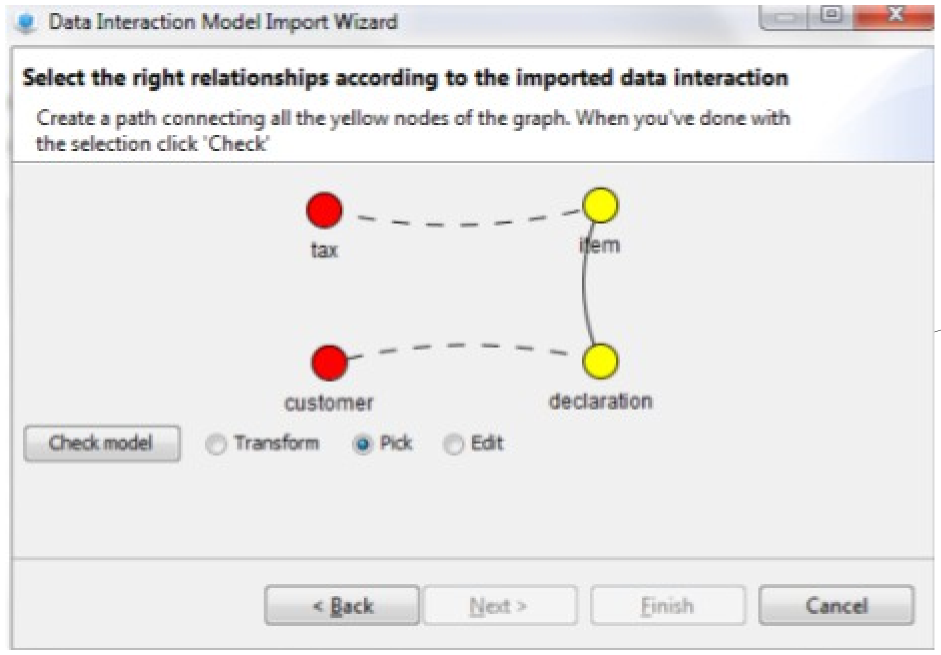
Select the right relationships according to the imported data interaction

Create a path connecting all the yellow nodes of the graph. When you've done with the selection click 'Check'


DB
1. Verify Model with Schema
2. Extract Relational Graph
3. Select Interaction Path
4. Query Generation and Execution
5. Visualization of Coverage

Method

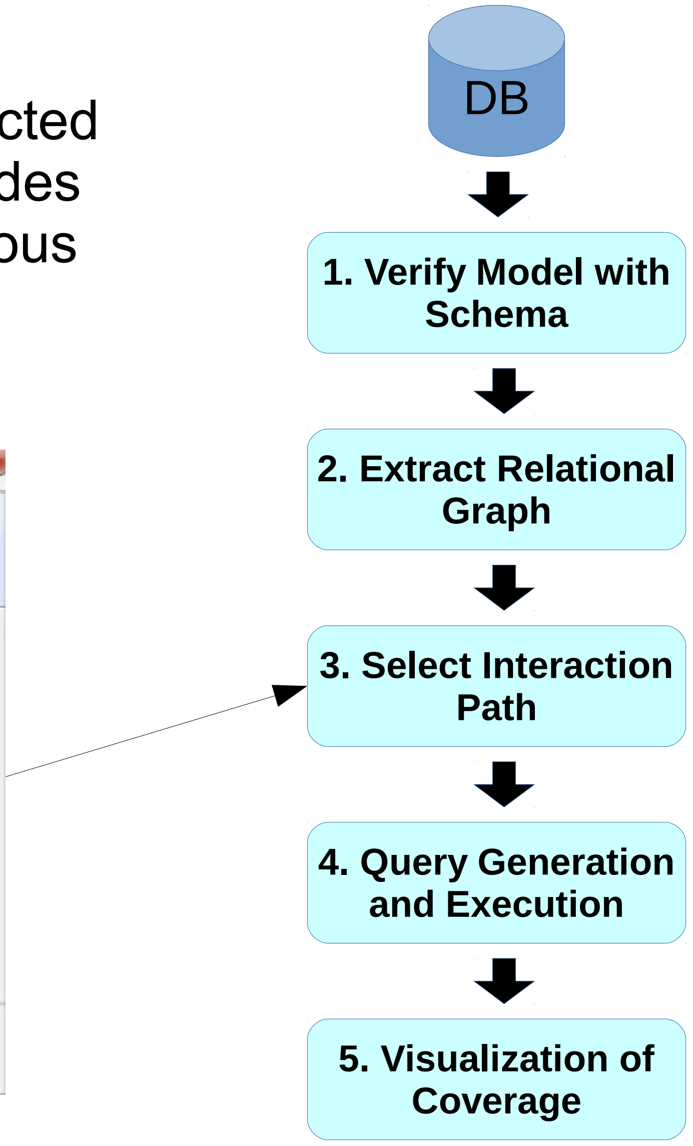
- The human domain expert defines a connected sub-graph between interacting tables or nodes in the relational graph obtained in the previous step.



Data Interaction Model Import Wizard

Select the right relationships according to the imported data interaction

Create a path connecting all the yellow nodes of the graph. When you've done with the selection click 'Check'

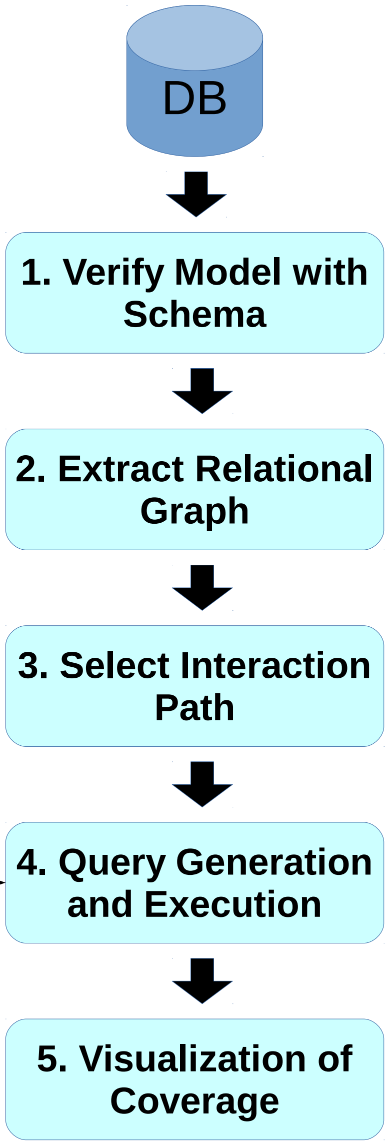


DB
1. Verify Model with Schema
2. Extract Relational Graph
3. Select Interaction Path
4. Query Generation and Execution
5. Visualization of Coverage

Method

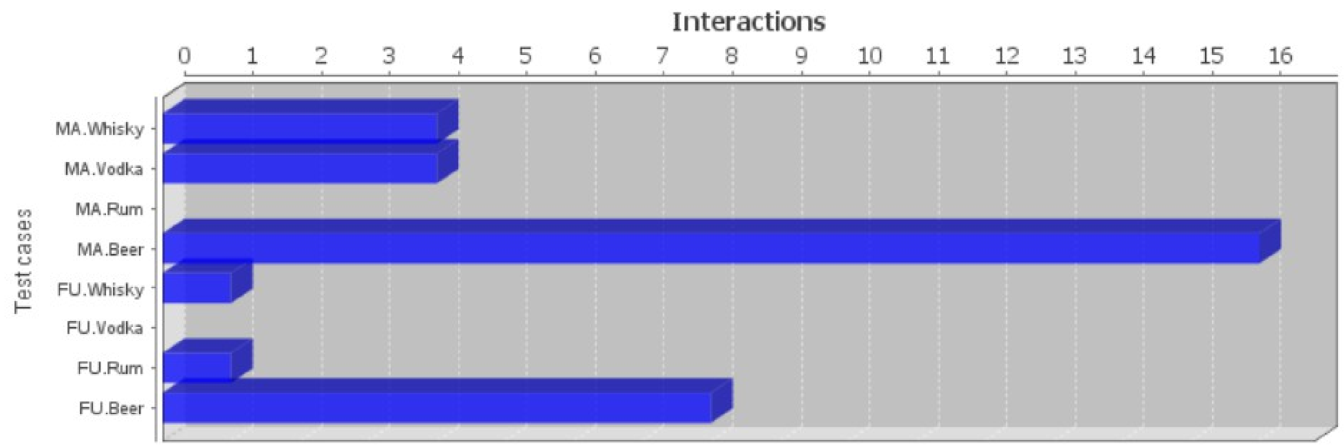
- The connected sub-graph between all interacting nodes is used by DEPICT to generate SQL queries to create an interaction table for each test case.
- These queries are executed by DEPICT on the test database to populate the interaction table, and compute the frequency of occurrence.

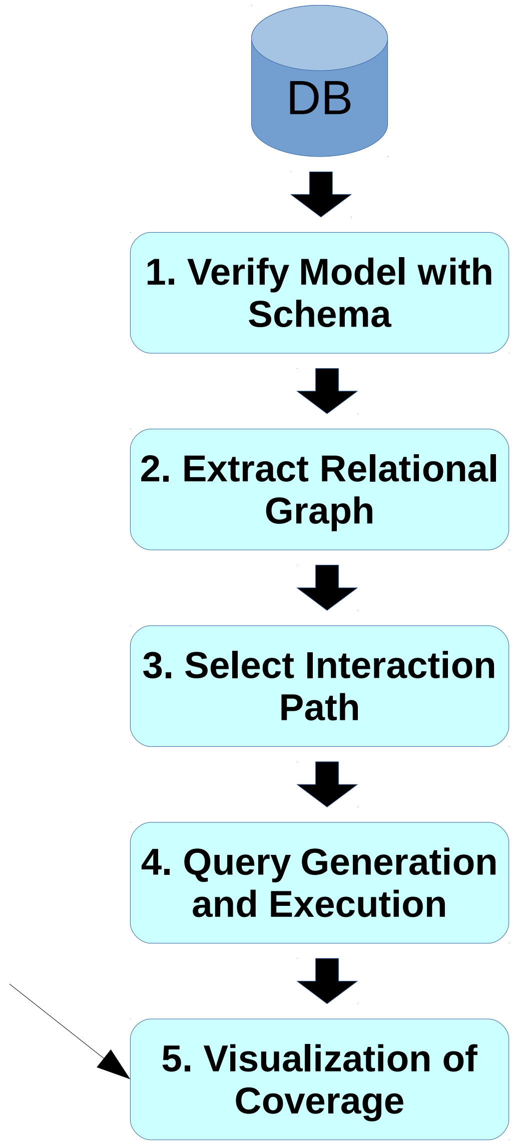
```
SELECT declarations.customerid AS f_1, declarations.date AS f_2,
 declarations.sequence AS f_3,declarations.version AS f_4,
 items.customerid AS f_5, items.date AS f_6, items.sequence AS f_7,
 items.version AS f_8, items.linenumber AS f_9
FROM declarations JOIN items ON declarations.customerid=items.customerid
AND declarations.date=items.date AND declarations.sequence=items.sequence
AND declarations.version=items.version WHERE declarations.category='MA'
AND items.itemcode=22086000
```


DB
1. Verify Model with Schema
2. Extract Relational Graph
3. Select Interaction Path
4. Query Generation and Execution
5. Visualization of Coverage

Method

- To visualize the results of the executed queries an HTML report is provided.
- The report gives instant feedback to a tester about interactions that are missing in a test database and need to be created.



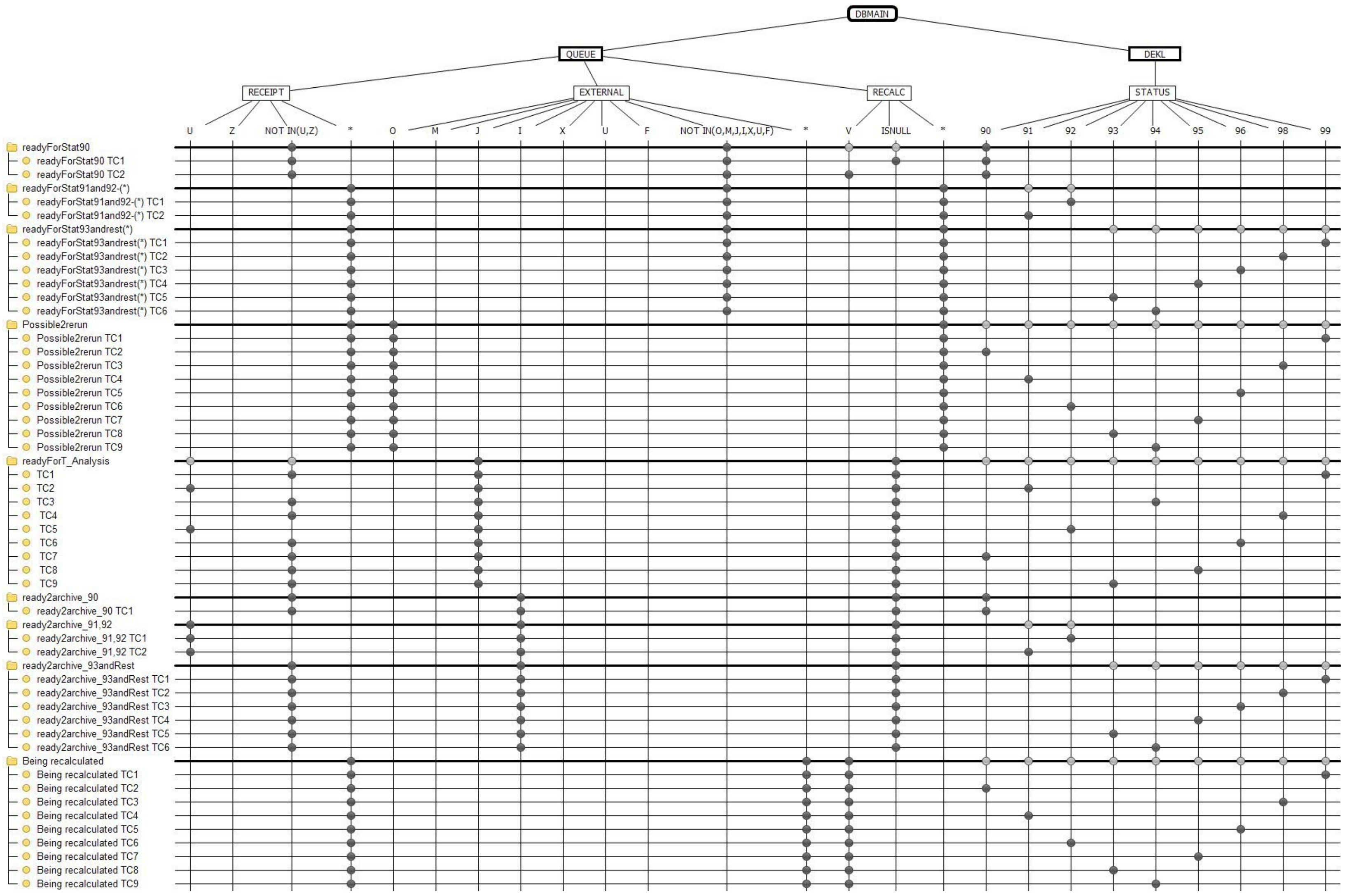

DB
1. Verify Model with Schema
2. Extract Relational Graph
3. Select Interaction Path
4. Query Generation and Execution
5. Visualization of Coverage

Industrial Evaluation

Authors evaluated this approach and DEPICT on the industrial test database TempDB and FinalDB of DNCE performing a case study to address the following questions:

- Can modeling data interactions effectively represent realistic scenarios in test databases?
- What are the features of human-in-the-loop interactive interface to facilitate accurate specification of data interactions?
- Is DEPICT time efficient and scalable in terms of query execution and report generation?

BENEFITS OF AUTOMATION IS OBVIOUS FOR LARGER MODELS



Industrial Evaluation

Conclusion

- In this paper, authors presented a method and tool DEPICT to verify coverage of data interactions (test cases) in a test database.
- The domain of data interactions and test cases are represented in a classification tree model. DEPICT, connects to a test database and verifies if these test cases are covered by the database.
- Data interactive coverage is proposed as a novel coverage criteria for test databases.
- The approach was qualitatively validated on industrial test databases extracted from live transaction streams at the DNCE.
- In future, authors would like to leverage DEPICT to promote the general research area of model-based data quality. They would like to use models to represent high-level properties of data and use DEPICT to verify these properties in very large databases (relational, graph XML, JSON).

References

- Sen, Sagar, Carlo Ieva, Arnab Sarkar, Astrid Grime, and Atle Sander. ***Experience Report: Verifying Data Interaction Coverage to Improve Testing of Data-Intensive Systems*** In International Symposium on Software Reliability Engineering., 2014.