Rihan Hai, Vasileios Theodorou, Maik Thiele, Wolfgang Lehner

SCIT: A Schema Change Interpretation Tool for Dynamic-Schema Data Warehouses

Erstveröffentlichung in / First published in:

Databases Theory and Applications: 26th Australasian Database Conference. Melbourne, 04.-07.07.2015. Springer, S. 323–327. ISBN 978-3-319-19548-3.

DOI: http://dx.doi.org/10.1007/978-3-319-19548-3_28

Final edited form was published in "Databases Theory and Applications: 26th Australasian Database Conference. Melbourne 2015", S. 323–327, ISBN 978-3-319-19548-3
http://dx.doi.org/10.1007/978-3-319-19548-3_28

SCIT: A Schema Change Interpretation Tool for Dynamic-Schema Data Warehouses

Rihan Hai[1](✉), Vasileios Theodorou[2], Maik Thiele[1], and Wolfgang Lehner[1]

[1] Technische Universität Dresden, Dresden, Germany
{rihan.hai,maik.thiele,wolfgang.lehner}@tu-dresden.de
[2] Universitat Politècnica de Catalunya, Barcelona, Spain
vasileios@essi.upc.edu

Abstract. Data Warehouses (DW) have to continuously adapt to evolving business requirements, which implies structure modification (schema changes) and data migration requirements in the system design. However, it is challenging for designers to control the performance and cost overhead of different schema change implementations. In this paper, we demonstrate SCIT, a tool for DW designers to test and implement different logical design alternatives in a two-fold manner. As a main functionality, SCIT translates common DW schema modifications into directly executable SQL scripts for relational database systems, facilitating design and testing automation. At the same time, SCIT assesses changes and recommends alternative design decisions to help designers improve logical designs and avoid common dimensional modeling pitfalls and mistakes. This paper serves as a walk-through of the system features, showcasing the interaction with the tool's user interface in order to easily and effectively modify DW schemata and enable schema change analysis.

1 Introduction

Data Warehouses (DW) have traditionally been designed to work on carefully modeled, predefined schemata, developed in the first phase of the overall system design. However, in a modern and highly volatile business environment, with changing market needs, company policies and technological advances, neither the data structure nor the analytical needs remain stable. It has recently been stressed that the incessant pressure of schema change is impacting every database [1]. Moreover, schema changes pose a threat to the accuracy of current applications and queries regarding the consistency of query answering, deciding schema equivalence, schema mapping composition and inversion. In addition, DW designers often lack the intuition of the performance and cost impact of different schema change implementations, while at the same time some dimensional modeling pitfalls may even compromise the original DW system [2]. Nonetheless, to our best knowledge, there is no existing implemented tool facilitating effectiveness and automation in DW schema change operations and analysis.

We thereby present SCIT, a flexible schema change interpretation tool for DW designers, supporting the improvement of logical system modeling. In its

-

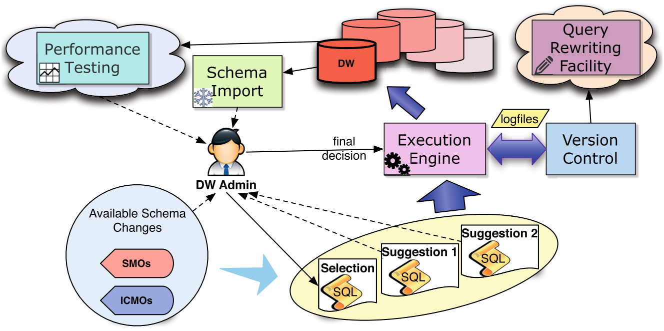


Fig. 1. SCIT framework

design, SCIT balances the need to interpret a wide range of schema changes with the imposed complexity of providing design alternatives for special dimensional structures. The main contributions of SCIT are summarized as follows:

- SCIT facilitates the interpretation of DW schema changes in a practical implementation where users are guided through a sequence of button clicks on an intuitive user interface. A representative set of thirty structural changes and eight basic integrity constraint changes, as well as their combinations, are supported by our tool, which automatically interprets them and generates executable SQL scripts containing equivalent Data Definition Language (DDL) statements. DW designers benefit from this time-saving function, especially during massive schema modification implementation tasks.
- SCIT employs an iterative mechanism to facilitate the design of alternative schemata. At every step, changes are assessed. And according to the satisfaction of specific dimensional structure conditions, SCIT recommends possible alternative designs based on best-practices. By comparing and simulating such alternatives, DW designers can obtain enhanced solutions regarding performance and cost.
- SCIT offers schema evolution traceability by maintaining files, which record the sequence of schema changes. Thus, it enables schema evolution analysis where designers can further explore potential issues, e.g., identifying and rewriting queries invalidated by schema changes and maintaining structural understandability.

2 System Overview

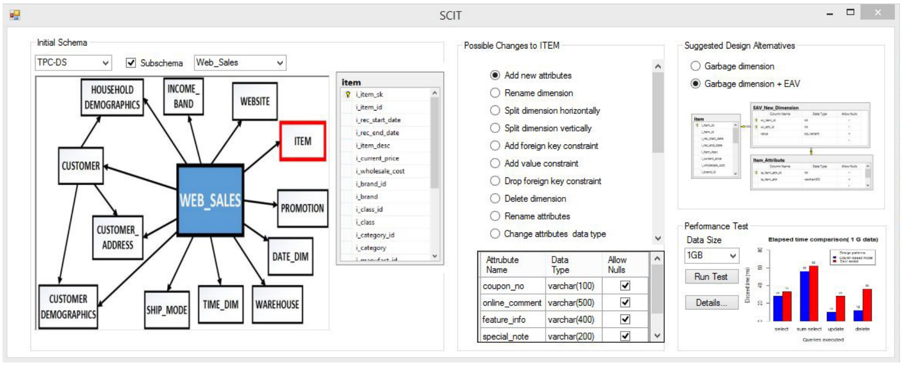
In this section, we introduce the functions and underlying mechanisms that make SCIT an efficient and flexible schema change interpretation tool. A high level view of our system architecture can be seen in Fig. 1 showing the process of using SCIT.

We build our tool to operate on a relational star schema, since DW systems are usually implemented following the star schema pattern, and according to Kimbal's recommendation [2], detailed and atomic information should be loaded into a star schema database. Below we describe the main system functionalities, showing how SCIT takes a user selection from a wide range of schema changes as input and generates equivalent SQL scripts as output, as well as intelligently providing design alternatives for improving logical system design:

Supported Schema Changes. SCIT supports a comprehensive set of Schema Modification Operators (SMOs) and Integrity Constraint Modification Operators (ICMOs) to represent atomic structural schema changes and keys or other dependencies, respectively. Interested readers can refer to [1,3] for formal representation and characteristics of those operators. The enhancement of SCIT is that we extend the original set from 18 [1] to 38 operators (30 atomic structural changes and 8 integrity constraint changes) and their combinations. This set includes operators for special design alternatives, e.g., pivot a row-based table into an Entity-Attribute-Value (EAV) model (column-based model).

Special Design Alternatives. Based on [2,4], SCIT supports a representative, practical set of twelve common design alternatives corresponding to specific dimensional structures, e.g., mini-dimensions for fast-changing attributes, EAV model for highly sparse dimensions and bridge tables for ragged variable depth hierarchies and multi-valued attributes. When schema changes satisfy certain predefined conditions, SCIT recommends the implementation of such alternatives to users, while automatically generating the corresponding executable SQL script for each suggestion, which supports users to choose desirable design regarding further performance test results or specific system requirements. As an example, let us consider a schema change scenario where a user inputs *add new attributes* as desired schema change and the following conditions also stand: (1) new attributes are text attributes (e.g., comment fields) and (2) the target table is a fact table. Then, SCIT will suggest the usage of *Garbage dimension* by putting the text attributes in a new dimension table, adding a new column in the fact table and connecting the dimension table to the target fact table by a foreign key constraint. In case these attributes are highly sparse, SCIT additionally suggests the application of an EAV model to the new dimension table. Designers are then able to conduct performance testing to compare the original design and alternative recommendations.

Schema Change Interpretation. Considering the required processes after schema modifications, such as rewriting invalid queries, which requires information preserving, necessary actions are automatically taken by the system. All thirty atomic structural implementation changes are embedded with foreign keys, which hides dimensions or facts supposed to be dropped, by cutting all existing foreign key constraints and renaming the target table. Taking *split a dimension table vertically* as an example, each new sub-table will get a newly generated surrogate key as primary key, as well as equivalent constraints mapped from the original dimension table. In the original table, all the foreign key constraints are dropped and the table is renamed to be hidden from the schema.



Initial Schema: TPC-DS ✓ Subschema Web_Sales

( ) Add new attributes [✓]
( ) Rename dimension
( ) Split dimension horizontally
( ) Split dimension vertically
( ) Add foreign key constraint
( ) Add value constraint
( ) Drop foreign key constraint
( ) Delete dimension
( ) Rename attributes
( ) Change attributes data type

Attrubute Name	Data Type	Allow Nulls
coupon_no	varchar(100)	✓
online_comment	varchar(500)	✓
feature_info	varchar(400)	✓
special_note	varchar(200)	✓

( ) Garbage dimension
[✓ Garbage dimension + EAV]

Data Size: 1GB

Fig. 2. SCIT User Interface

3 Demonstration Scenario

The demonstration will illustrate the basic use of SCIT as an effective schema change interpretation tool, highlighting its role as an alternative design adviser. The following scenarios will be running on Microsoft SQL server 2014:

Initial Schema Set Up. We use the TPC-DS benchmark[1] for our demonstration, which has a relatively complex schema of a retail model, as our initial schema. Visualization of the whole or partial schema is presented on the main screen (shown in the left section of Fig.2). By clicking on one dimension/fact table, users can obtain basic structural information of the chosen element(s).

Schema Change Selection. In this scenario we demonstrate the implementation of certain schema changes. Possible structural or integrity constraint changes regarding the chosen dimension/fact will be automatically listed in the middle of the main screen. After choosing a desired schema modification from the list, users need to further input corresponding parameters, as shown in the middle section of Fig.2 (attribute name, data type, etc.).

Special Scenarios for Design Alternatives. In this scenario we select changes that trigger the suggestions of alternative designs. Users will observe the visualized sub-schema of each alternative and performance test results with configurable size of data loaded into the RDBMS. We measure the query elapsed time for the original design and a chosen alternative. Thus, we illustrate how users can obtain an estimation of performance comparison between alternative designs. In the right section of Fig.2, we show the performance comparison between a basic schema change implementation and SCIT suggested design alternative (Garbage dimension + EAV) based on 1GB data.

Acknowledgments. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC).

[1] http://www.tpc.org/tpcds/

References

1. Curino, Carlo, et al.: Automating the database schema evolution process. The VLDB Journal The International Journal on Very Large Data Bases **22**(1), 73–98 (2013)
2. Kimball, R., Ross, M.: The data warehouse toolkit: The definitive guide to dimensional modeling. John Wiley & Sons (2013)
3. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10**(4), 334–350 (2001)
4. Ballard, C., et al.: Dimensional Modeling: In a Business Intelligence Environment. IBM Redbooks (2012)