

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Formal Design of Data Warehouse and OLAP Systems

A dissertation presented in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy
in
Information Systems

at Massey University, Palmerston North, New Zealand

Jane Qiong Zhao

2007

Abstract

A data warehouse is a single data store, where data from multiple data sources is integrated for online business analytical processing (OLAP) of an entire organisation. The rationale being single and integrated is to ensure a consistent view of the organisational business performance independent from different angles of business perspectives. Due to its wide coverage of subjects, data warehouse design is a highly complex, lengthy and error-prone process. Furthermore, the business analytical tasks change over time, which results in changes in the requirements for the OLAP systems. Thus, data warehouse and OLAP systems are rather dynamic and the design process is continuous. In this thesis, we propose a method that is integrated, formal and application-tailored to overcome the complexity problem, deal with the system dynamics, improve the quality of the system and the chance of success.

Our method comprises three important parts: the general ASMs method with types, the application tailored design framework for data warehouse and OLAP, and the schema integration method with a set of provably correct refinement rules.

By using the ASM method, we are able to model both data and operations in a uniform conceptual framework, which enables us to design an integrated approach for data warehouse and OLAP design. The freedom given by the ASM method allows us to model the system at an abstract level that is easy to understand for both users and designers. More specifically, the language allows us to use the terms from the user domain not biased by the terms used in computer systems. The pseudo-code like transition rules, which gives the simplest form of operational semantics in ASMs, give the closeness to programming languages for designers to understand. Furthermore, these rules are rooted in mathematics to assist in improving the quality of the system design.

By extending the ASMs with types, the modelling language is tailored for data warehouse with the terms that are well developed for data-intensive applications, which makes it easy to model the schema evolution as refinements in the dynamic data warehouse design.

By providing the application-tailored design framework, we break down the design complexity by business processes (also called subjects in data warehousing) and design concerns. By designing the data warehouse by subjects, our method resembles Kimball's "bottom-up" approach. However, with the schema integration method, our method resolves the stovepipe issue of the approach. By building up a data warehouse iteratively in an integrated framework, our method not only results in an integrated data warehouse, but also resolves the issues of complexity and delayed ROI (Return On Investment) in Inmon's "top-down" approach. By dealing with the user change requests in the same way as new subjects, and modelling data and operations explicitly in a three-tier architecture, namely the data sources, the data warehouse and the OLAP (online Analytical Processing), our method facilitates dynamic design with system integrity.

By introducing a notion of refinement specific to schema evolution, namely schema refinement, for capturing the notion of schema dominance in schema integration, we are able to build a set of correctness-proven refinement rules. By providing the set of refinement rules, we simplify the designers's work in correctness design verification. Nevertheless, we do not aim for a complete set due to the fact that there are many different ways for schema integration, and neither a prescribed way of integration to allow designer favored design.

Furthermore, given its flexibility in the process, our method can be extended for new emerging design issues easily.

Acknowledgement

My sincere and huge thanks go to my supervisor Prof. Klaus-Dieter Schewe for all the guidance and support that he has given me generously during my study.

My thanks also goes to my co-supervisor Associate Prof. Roland Kaschek for his critical discussions on this work.

I am grateful towards my parents for their constant love and care, and the great help in looking after my children.

Finally my thanks to my partner, Henning Köhler, and my two beautiful and lovely daughters, Angela and Nicole, for their love and support which made my life meaningful and enjoyable.

It would be impossible for the completion of this work without any of the help from the people mentioned and the people around me.

Contents

1	Introduction	5
1.1	Contributions	8
1.2	Outline	9
2	Literature Review	11
2.1	Data Warehouse and OLAP Systems Design	11
2.1.1	Design Methods	11
2.1.2	Data Warehouse Evolution	17
2.2	Schema Integration	19
2.3	Software System Development Methods	20
2.3.1	Process Oriented Methods - a Brief Introduction	21
2.3.2	Formal Methods	21
2.3.3	Application of Formal methods	23
2.4	Summary	23
3	Abstract State Machines	25
3.1	The Notion of ASMs	25
3.2	Mathematical Definition of ASMs	26
3.2.1	Abstract States	26
3.2.2	Transition Rules and Runs	28
3.2.3	The Reserve of ASMs	32
3.3	ASM Modules	32
3.4	Distributed ASMs	33
3.5	The Ground Model Method	34
3.5.1	The Properties of the Ground Model	34
3.5.2	Three Problems in Formulation	35
3.5.3	ASMs for the Formalisation Problems	35
3.5.4	An Example of Ground Model	36
3.6	The ASM Refinement Method	37
3.6.1	The Notion of Refinement	37
3.6.2	The Refinement Patterns	38
3.6.3	Correctness Proofs	39
3.6.4	Notions of Refinement: A Comparison with ASM refinement	41
4	Typed Abstract State Machines	43
4.1	A Type System	43
4.2	Signatures and States	45
4.3	Transition Rules	46
4.4	Terms	47

4.5	Bulk Updates	48
4.6	Schema Refinement in TASM	49
4.7	An Equivalence Result	51
5	Data Warehouse Design Using the ASM Method	53
5.1	The Data Warehouse Ground Model	54
5.1.1	The Operational Database ASM	54
5.1.2	The Data Warehouse ASM	55
5.1.3	The OLAP ASM	57
5.1.4	The Grocery Store Data Warehouse - a Simple Example	60
5.1.5	The Ground Model in TASM	65
5.1.6	Reasoning about the ASM Ground Model	68
5.2	The Refinement-based Design Framework	69
5.2.1	Requirements Capture	70
5.2.2	Optimisation	71
5.2.3	Implementation	72
5.3	Some Refinements	73
5.3.1	Incorporating Roll-up and Drill-down	73
5.3.2	Materialising OLAP Views	75
5.3.3	Incremental Updates	77
6	View Integration	83
6.1	HERM	83
6.2	Query Languages for HERM	86
6.3	Schema Dominance and Equivalence	88
6.4	Schema and View Integration Process	89
6.5	Transformation Rules	90
6.5.1	Schema Restructuring	91
6.5.2	Shifting Attributes	99
6.5.3	Schema Extension	104
6.5.4	Type Integration.	108
6.5.5	Handling Integrity Constraints.	113
6.6	Dialogue Types	120
6.7	Transformation Rules for Dialogue Types	121
7	Case Studies	123
7.1	Adding a New Data Mart	123
7.1.1	Example: CRM for Grocery Store	123
7.1.2	Incorporate CRM	124
7.2	Dynamic Data Warehouse Design	133
7.2.1	Cost and benefit Model	133
7.2.2	View Selection Process	134
7.2.3	Application Cases	135
7.3	Distribution Design	138
7.3.1	Architecture of Distributed Data Warehouses	138
7.3.2	Fragmentation	139
7.3.3	Query and Maintenance Cost	140
7.3.4	Recombination of Fragments	141
7.3.5	Distribution design for a Grocery Store Data Warehouse	142

7.4	Application of Business Statistics	148
7.4.1	Single Linear Regression and Correlations	148
7.4.2	Time Series Analysis	150
8	Conclusion	152

List of Figures

- 3.1 The ASM refinement scheme 38
- 3.2 data refinement vs. ASM refinement 42

- 5.1 The general architecture of a data warehouse and OLAP 54
- 5.2 The main process in OLAP-ASM 58
- 5.3 Schema underlying an OLAP view 61
- 5.4 The operational database schema 62
- 5.5 The data warehouse schema for sales 63

- 6.1 HERM diagram for loan application 85
- 6.2 The relationship types before and after application of Rule 9 100
- 6.3 The relationship types before and after application of Rule 10 101
- 6.4 The possible path directions 102
- 6.5 The relationship types before and after application of Rule 11 103
- 6.6 The relationship types before and after application of Rule 12 104
- 6.7 The relationship types before and after application of Rule 14 105
- 6.8 The relationship types before and after application of Rule 16 107
- 6.9 The relationship types before and after application of Rule 17 108
- 6.10 The relationship types before and after application of Rule 18 109
- 6.11 The relationship types before and after application of Rule 19 110
- 6.12 The relationship types before and after application of Rule 20 112
- 6.13 The relationship types before and after application of Rule 21 113
- 6.14 The relationship types before and after application of Rule 22 114
- 6.15 The relationship types before and after application of Rule 23 115
- 6.16 The relationship types before and after application of Rule 24 116
- 6.17 The relationship types before and after application of Rule 25 117
- 6.18 The relationship types before and after application of Rule 27 119
- 6.19 The relationship types before and after application of Rule 28 120

- 7.1 The data warehouse schema for CRM 124
- 7.2 The data warehouse schema after incorporating CRM 127
- 7.3 The operational DB schema after incorporating CRM 127
- 7.4 Distributed Data Warehouse Architecture 139