Generating multidimensional schemata from relational aggregation queries

Pang, Chaoyi; Taylor, Kerry; Zhang, Xiuzhen; Cameron, Mark

Pang, C., Taylor, K., Zhang, X., & Cameron, M. (2004). Generating multidimensional schemata from relational aggregation queries. Web Information Systems - WISE 2004 International Conference on Web Information Systems Engineering. https://doi.org/10.1007/b103344

Published Version: https://doi.org/10.1007/b103344

Generating Multidimensional Schemata from Relational Aggregation Queries

Chaoyi Pang[1], Kerry Taylor[1], Xiuzhen Zhang[2], and Mark Cameron[1]

[1] CSIRO ICT Centre and Preventative Health National Flagship Program, Australia
{chaoyi.pang, kerry.taylor, mark.cameron}@csiro.au
[2] School of Computer Science & IT, RMIT University, Australia
zhang@cs.rmit.edu.au

Abstract. Queries on operational databases can be viewed as a form of business rules on data warehouse schema design. We propose to use such queries to automatically generate measures, dimensions and dimension hierarchies and their representation in a star schema. The schema produced with our approach has good properties such as non-redundant dimensional attributes and orthogonality among dimensions and can answer many more queries than just those it was generated from.

1 Introduction

The multidimensional model has proven extremely successful in data warehouse and On-line Analytical Processing (OLAP) applications. OLAP applications are dominated by analytical queries: rather than retrieving detailed information from a data repository about individual transactions, the main concern is to retrieve summaries of data. Designing a multidimensional schema involves deciding (1) the dimensions and attributes describing each dimension, (2) the aggregation hierarchy for the dimensions, and (3) the measure attributes and their dependent dimensions. Most current methods on schema design, such as [5], start the design process with requirements analysis and specification, then conceptual design, logical design, and finally physical design. In contrast, we treat summary queries over a source schema as the statement of requirements, and automatically derive the logical design. Our rationale is that, when collecting requirements for designing data warehouses, stakeholders can more easily formulate the knowledge or analysis they are expecting as queries over existing data resources, rather than by describing a view of the enterprise business process or research needs.

Our motivation for researching this problem is stimulated by the following observations in practice:

– Designing a multidimensional schema is difficult. Complex business processes make it hard to achieve an accurate overall picture of an enterprise to understand data analysis needs. As an alternative to the phase of requirement collection and business rules analysis employed in data warehouse design, we use the schemas and analytical queries of conventional databases as a proxy

Citation:
Pang, C, Taylor, K, Zhang, X and Cameron, M 2004, 'Generating multidimensional schemata from relational aggregation queries', in X. Zhou et al. (ed.) Web Information Systems - WISE 2004 Fifth International Conference on Web Information Systems Engineering, Berlin, 1 November 2004.

for a specialist designer. A significant feature of this approach is that queries can be composed easily by business analysts and database developers with little expertise in data warehouse design.
- The schemas and analytical queries on the conventional database are essential elements in building any data warehouse. Generally, the schemas imply functional dependencies among attributes which suggest hierarchical structures within dimensions and can be used to eliminate unprofitable redundancy. In practice, functional dependencies and referential constraints are stored in system tables as metadata and may be readily available. Queries, on the other hand, associate raw data in the conventional database with the manipulated data in the data warehouse. They identify the desired application scope.
- OLAP applications are a data driven exercise where the requirements for the output often evolve as the data is extracted. Moreover, such applications are characterized by the use of many similar and repeated queries. When new demands occur, our design methodology can evolve the previously generated schemata into new schemata by integrating the new queries interpreted from the demands [12].

2 Our Method

We analyze the attributes from the given relational queries and the dependency relationships among them to design a schema with good properties such as non-redundant dimensional attributes and orthogonality among dimensions. Our designed schema can answer the queries that it was generated from and has the minimal number of dimensions for an aggregation. Furthermore, the schema can answer many more queries other than the given ones.

In the following, we compare a hand-designed schema with an automatically generated one for a running example. Our theoretical results can be explained through the following observations: (i) all the dimensional attributes are relevant; (ii) the dimensional hierarchies are properly described; (iii) aggregation-dimension relationships are defined precisely; and, as a result, (iv) aggregation-dimension relationships are non-redundant. In fact, these observations are inherited from the following properties: (1) the schema can answer the generating queries; (2) the schema can answer many more additional queries; (3) the total number of attributes in the dimensions is minimal (counting the attributes in a hierarchy relation as one attribute); and (4) the number of dimensions for aggregation is minimal. Properties (1) and (2) imply a query reuse capability of the obtained schema. Property (3) indicates that if an attribute in a dimension is removed then there is a query amongst the given queries which could not be answered from the computed schema. Property (4) indicates that any two dimensions for the same aggregation are *orthogonal* [9], which is a key design criteria for an efficient data warehouse.

We need the following steps to build a multidimensional schema from a set of queries: we first use the proposed *measure dependency* to represent each ag-

gregate query over a source schema; then apply the *transformations* to associate the "relevant" measure dependencies together so that they can be treated in the same cube of the multidimensional schema; and lastly apply functional dependencies to define dimensions and dimensional hierarchies in a cube. Two transformations are introduced: a *referential* transformation and a *join* transformation that relies on functional dependency and recognizing query equivalence. There exist polynomial time algorithms for computing the closure of functional dependencies and recognizing query equivalence that apply [1, 7]. By using the dependency relations and referential constraints between attributes, we have developed a method within a cube to determine the dimensional attributes and to group them into dimensional hierarchies.

3 Related Work

To the best of our knowledge, this is the first paper that designs data warehouse schemata using queries on a conventional database. We are not aware of any papers that use the dependency relationship and referential constraints to associate queries in designing a multidimensional schema.

Our method generalizes the procedure used in OLAP products such as Cognos, Business Objects and MicroStrategy, where a single query over a relational database is used to generate a specific, single-purpose multidimensional schema. Our idea on transformation rules for queries is different to the one used in materialized view advisors on the existing commercial products such as those from Oracle and IBM: Their methods are based on the existing data warehouse schema and extract views either to flood the schema or to materialize for efficient computation. [11] is based on the data warehouse cube.

Recently, materialized views have been explored extensively to provide massive improvements in query processing time, especially for aggregation queries over large databases [2–4]. These methods rely on the source database remaining intact for complete query answering, but they supplement the database with derived data to speed up query processing. This is very different from our method in which we aim to define a new data warehouse schema, with good design properties, against which data warehouse queries may be addressed without ongoing reference to the source data for query answering. Having designed the data warehouse, methods for view materialization might be applied to choose a population and maintenance strategy for the data warehouse, taking account of space efficiency and view maintenance costs.

To ensure a good design, papers such as [8–10] propose a *multidimensional normal form* for schema design. The normal form is used for reasoning about the quality of a schema to reduce redundancy, diminish sparsity and to retain summarizability. Sparsity relates to the dimension orthogonality, which, in turn, can be reflected by the existence of a functional dependency. We will use some of these concepts in our schema design.

4 Running Example

In the following, we use a running example to briefly describe our result on schema design. Refer to [13] for the detailed illustration and algorithm.

```
Branch(branchNo, bstreetAddress, bCity)
LoanManager(empNo, empName, phone, branchNo)
Customer(custNo, custName, profession, streetAddress, city, state)
Account(accNo, accType, balance,aDate, custNo)
LoanContract(conNo, loantype, amount, cDate, empNo, custNo)
```

Table 1. A banking relational database

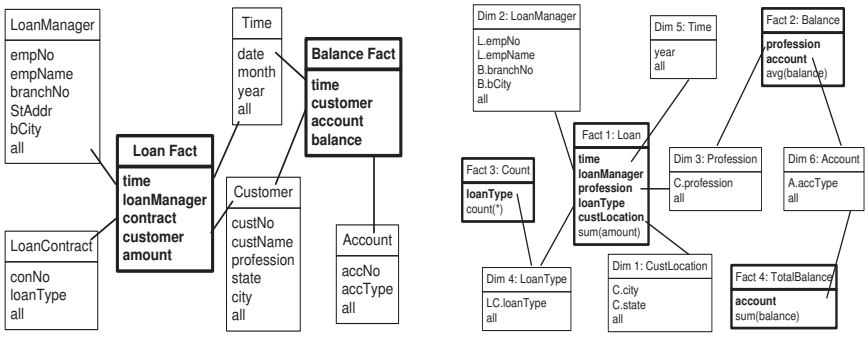

Fig. 1. The star schema: (1) by hand; (2) by queries.

We study the operational banking database as shown in Table 1, where the primary keys for each relation are underlined. Some sample queries and their SQL equivalents are given in Table 2. On one hand, we generate the warehouse schema of Figure 1(1) manually by the data warehouse design process recommended in a standard text [6]. We use the sample queries to indicate the application scope for that design. On the other hand we can design the warehouse schema directly from the queries. We therefore automatically derive the star schema of Figure 1(2) by using our method with the queries in Table 2. By assessing the two generated schemata, we have the following comments . First, the irrelevant dimensional attributes in Figure 1(1) result in poor dimension hierarchies and data redundancy in facts and dimensions; the automatically generated schemata does not include the redundant attributes such as branchName, Month, accNo and cusNo in Figure 1(1). These attributes are not referred to in any given queries and therefore they are considered inappropriate to the application scope. Including

Q_1 : The total loan in 2002.

```
select sum(LC.amount)
from LoanContract LC
where LC.cDate = 2002
```

Q_2 : For the types of loans with more than 10 loan contracts, the type of loan and the number of contracts.

```
select LC.loantype, count(*)
from LoanContract LC
group by LC.loantype
having count(*) > 10
```

Q_3 : For each loan type, and each city and state where customers resides, the total loan amount in 2002.

```
select C.city,C.state,C.loanType,
       sum(LC.amount)
from LoanContract LC, Customer C
where LC.custNo = C.custNo and
       LC.cDate = 2002
group by C.city,C.state,C.loanType
```

Q_4 : For each type of loan and each customer profession, list the total loan amount.

```
select LC.loanType,C.profession, sum(LC.amount)
from LoanContract LC, Customer C
where LC.custNo = C.custNo
group by LC.loanType,C.profession
```

Q_5 : The performance (total loan amount) of each loan manager in 2002.

```
select L.empNo,L.empName,sum(LC.amount)
from LoanContract LC, LoanManager L
where LC.empNo = L.empNo and LC.cDate ≥
       2002 and LC.cDate < 2003
group by L.empNo, L.empName
```

Q_6 : The total loan amount of each branch.

```
select B.branchNo, sum(LC.amount)
from Branch B, LoanManager L, LoanContract LC
where B.branchNo = L.branchNo and
       L.empNo = LC.empNo
group by B.branchNo
```

Q_7 : The overall performance of each customer profession and account type.

```
select A.accType,C.profession,avg(A.balance)
from Account A, Customer C
where A.custNo = C.custNo
group by A.accType, C.profession
```

Q_8 : The total loan amount for each branch and city

```
select B.bCity, B.branchNo, sum(LC.amount)
from Branch B, LoanManager L,
       LoanContract C
where B.branchNo = L.branchNo and
       L.empNo = LC.empNo
group by B.bCity, B.branchNo
```

Q_9 : The average balance for each customer profession.

```
select C.profession, avg(A.balance)
from Account A, Customer C
where A.custNo = C.custNo
group by C.profession
```

Q_{10} : The total balance for each account type.

```
select A.accType, sum(A.balance)
from Account A
group by A.accType
```

Table 2. Aggregation queries on the banking database.

many irrelevant attributes in the dimensions could cause inefficient query execution and data storage. Moreover, due to the existence of irrelevant attributes in Figure 1(1), the dimension hierarchies of Figure 1(1) are not described properly. For example, the "Customer" dimension of Figure 1(1) is represented as two dimensions in Figure 1(2), CustomerLocation and CustomerProfession. But from the aggregation hierarchy, we can see that they are orthogonal, being two disjoint aggregation paths for the Customer dimension. cusNo and custName are not dimensional attributes in Figure 1(2). These differences mean that the design of Figure 1(2) is more compact than that of Figure 1(1). Secondly, the vague Aggregation-Dimension relationships of Figure 1(1) causes redundancy. For example, Figure 1(1) describes the dependency of the Balance measure on the Time, Customer and Account dimensions in general, leading to redundancy in the fact table due to unrelated attributes in one dimension table. Figure 1(2) describes this dependency more precisely: the Average of balance depends on the Customer Profession and Account, whereas the Sum of balance depends on the Account dimension only. Lastly, even though the queries used to generate the schema are quite simple, the obtained schema supports more complicated

queries. For example, it can handle a query for "the best loan manager each year", where he is the one with the largest contract value sum.

5 Conclusion

Our future work is to improve and extend this work in several ways: to address a wider class of queries including disjunctive queries; to merge some similar dimension tables; to subdivide some attributes in order to create new measures; to capture multi-database dependencies; to automatically populate the automatically-built schema from the source database; and, more importantly, to evaluate our approach in the real world.

Acknowledgements: We would like to thank the anonymous referees for their very helpful remarks.

References

1. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *6th Int. Conference on Database Theory; LNCS 1186*, pages 56–70, Delphi, Greece, 1997.
2. D. W. Cheung, B. Zhou, B. Kao, H. Lu, T. W. Lam, and H. F. Ting. Requirement-based data cube schema design. In *Proc. of the 8th Intl. Conf. on Information and knowledge management*, pages 162–169. ACM Press, 1999.
3. R. Chirkova, A. Y. Halevy, and D. Suciu. A formal perspective on the view selection problem. *The VLDB Journal*, 11(3):216–237, 2002.
4. J. Goldstein and P.-Å. Larson. Optimizing queries using materialized views: a practical, scalable solution. *SIGMOD Record*, 30(2):331–342, June 2001.
5. B. Husemann, J. Lechtenborger, and G. Vossen. Conceptual data warehouse modeling. In *Design and Management of Data Warehouses*, page 6, 2000.
6. R. Kimball. *The Data Warehouse Toolkit.* John Wiley and Sons, 1996.
7. P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 205–213. ACM Press, 1998.
8. J. Lechtenbrger and G. Vossen. Multidimensional normal forms for data warehouse design. *Information Systems*, 28(5):415–434, 2003.
9. W. Lehner, J. Albrecht, and H. Wedekind. Normal forms for multidimensional databases. In *10th Intl Conf. on Scientific and Statistical Database Management, Proc., Capri, Italy, July 1-3, 1998*, pages 63–72. IEEE Computer Society, 1998.
10. H.-J. Lenz and A. Shoshani. Summarizability in OLAP and statistical data bases. In *Statistical and Scientific Database Management*, pages 132–143, 1997.
11. T. Niemi, J. Nummenmaa, and P. Thanisch. Constructing OLAP cubes based on queries. In *Proc. of the 4th ACM international workshop on Data warehousing and OLAP*, pages 9–15. ACM Press, 2001.
12. C. Pang, K. Taylor, and X. Zhang. Multidimensional schema evolution from aggregation queries. *CSIRO ICT Centre, Technical Report*, 03(186), 2003.
13. C. Pang, K. Taylor, X. Zhang, and M. Cameron. Generating multidimensional schema from aggregation queries. *CSIRO Mathematical and Information Sciences Technical Report*, 2003.