A Distributed OLAP Infrastructure for E-Commerce

Qiming Chen, Umesh Dayal, Meichun Hsu
Software Technology Laboratory
HP Laboratories
Palo Alto, California, USA
{qchen, dayal, mhsu}@hpl.hp.com

Abstract

Warehousing and mining sales transaction data to generate summary information, customer profiles, and business rules has become increasingly important in e-commerce. Such summary information and rules have to be extracted from very large collections of transaction data gathered at many distributed sites. This is challenging data mining, both in terms of the magnitude of data involved, and the need to incrementally adapt the mined patterns and rules as new data is collected. This paper describes a distributed and cooperative data warehousing, OLAP, and data mining infrastructure that addresses these challenges. Our contributions are as follows. First, we define various new classes of multi-dimensional and multi-level association rules (scoped multidimensional, with conjoint items, and functional) that can be extracted from customer profiles and are useful for e-commerce applications. Then, we show how customer profiles and different classes of association rules can be computed in a distributed, cooperative manner using OLAP tools. Finally, we show how the summaries, profiles, and rules can be incrementally updated as new transaction data is collected. This infrastructure has been prototyped at HP Labs.

1. Introduction

Mining transaction data to generate summary information, customer profiles and business rules benefits both merchants and customers [2,3,9,11], and has become increasingly important in e-commerce. For example, customer profiles can be used to derive shopping behavior patterns to guide personalized marketing, commercial promotion and fraud detection; association rules can be used to identify opportunities for cross-selling, explain causes of sudden sale increases or drops, analyze trends, etc. Such summary information and rules have to be extracted from very large collections of transaction data gathered at multiple distributed sites. To create and incrementally update the information, typically hundreds of millions of transaction records may have to be processed daily [1,4,6,10]. Such applications have challenged data mining in several aspects.

One challenge is to provide continuous rather than one-time value to e-commerce. Currently most data mining efforts are focused on analyzing historical data. In reality, however, data are continuously collected, and it is important to mine the data continuously to detect trends and changes dynamically in real-time. For instance, a cross-sale association rule describes the relationship of the sales of one item to the sales of another. While such relationships are helpful for making planning and promotion decisions, the *changes* of cross-sale associations may be even more significant, since such changes usually reflect real-time trends, the reaction to a promotion, or the cause of sales drops or rises. For example, suppose the sales of VCRs had been strongly associated with the sales of TVs, but this association has recently weakened as TV buyers turn to buying DVDs instead of VCRs. Such a change in the association helps to explain or predict the slow down of VCR sales. To catch such dynamic association relationships requires us to do association rule mining continuously and incrementally.

Another challenge is scaling to very large data sets. In a shopping network, a huge volume of transaction records must be processed every day, and it is unlikely that centralized processing will yield satisfactory performance. The scalability issue becomes even more critical to providing the real-time data mining service described above. In order to scale up, it is necessary to distribute data processing, reduce data volumes at each local site by summarization, and mine data incrementally at multiple levels of aggregation.

These challenges have motivated us to develop a distributed and cooperative data-warehouse, OLAP and data mining infrastructure. This infrastructure involves multiple Local Data-warehouse/OLAP Stations (LDOS), and a Global Data-warehouse/OLAP Station (GDOS) (in practice these stations may be organized in a multilevel hierarchy, but for simplicity in this paper

we consider only two levels). Although various data mining results may be generated, in this paper we focus on the generation and incremental update of *customer shopping behavior profiles* and *cross-sale association rules* derived from these profiles.

LDOSs are typically divided by geography. The transaction data from the area covered by an LDOS are summarized and reduced in volume, and the resulting summary information, partial customer profiles, including the local data for mining association rules, are sent to the GDOS to be used in combination with the data sent in by other LDOSs. Thus, LDOSs serve as distributed data collection, aggregation and reduction stations. They dramatically improve the parallelism of data mining and reduce the data load and computation load on the GDOS.

The GDOS integrates the summary information or partial knowledge fed from the LDOSs, and hence the GDOS can generate more complete knowledge than any single LDOS. For example, since customers shop around, the complete shopping activities of a single customer in two cities, which are covered by two separate LDOSs, may not be captured at any single LDOS. Thus a single LDOS may not have sufficient information to generate complete shopping behavior profiles or association rules.

The customer profiles, rules, etc, generated at the GDOS may be fed back to the LDOSs periodically or upon request. As a result, at each LDOS, global trends, rules or alerts can be used to provide local decision support guidelines. For example, suppose the cause of a dramatic drop in VCR sales was discovered at the GDOS by detecting the change in the association relationship between VCR sales and TV sales, and suppose that this conclusion was drawn from data covering a few geographic areas. Alerts can then be issued to other markets where the sales drop has not yet happened for guiding promotions or inventory management.

The distributed OLAP infrastructure allows us to generate association rules with enhanced expressive power. Let us focus on the typical cross-sale association rules, e.g. how the sale of item A is associated with the sale of item B. The cross-sale association rules reported in the literature so far are typically based on transactions, that is, they associate items sold together *in the same transaction*. This is too limiting for real applications. In practice, we might be interested in associations based on different elements, such as customers instead of transactions. These base elements form the *scope* of the association rules. For example, we can consider customer-based association of VCR sales with TV sales in one month, regardless of when, where, or whether in the same transaction, the purchases are made. As a customer may buy a TV and a VCR in different shops covered by different LDOSs, such an association rule may not be generated from the transactions seen at a single LDOS. Further, we can combine items, e.g. *product* and *time*, into conjoint items, e.g. *<product, time>*, to express more powerful association relationships such as the percentage of customers who bought a VCR in Jan99 after they bought a TV in Dec98. We can also introduce variables in rules to express such association relationships as the percentage of customers who bought a VCR within a month after they bought a TV.

Customer profiles and association rules are represented as *multidimensional data cubes* and handled by OLAP servers. The profile cubes at the LDOSs and the GDOS give *measures* such as sales in units, discounts, payment methods, etc, from which many other related measures such as discount rate and sales forecast, can be derived. Profile cubes can be based on volume and probability. Volume-based cubes are stored in the data warehouse. Probability-based cubes are derived from volume-based cubes as "views". Association rule related cubes are generated from profile cubes.

In summary, our major contributions are twofold. First, we have developed a **distributed OLAP based infrastructure** for mining e-commerce transaction data. Second, since our infrastructure enables us to *flexibly combine information from different locations and across different time periods*, we have introduced new kinds of multilevel, multidimensional association rules with enhanced expressive power, including the following:

- **scoped association rules**, which can be based on different elements, such as transactions or customers, as described above;
- **association rules with conjoint items**; and
- **functional association rules**.

This infrastructure has been designed and prototyped at HP Labs to support business intelligence in electronic commerce. At each site the mining engine is built on top of an Oracle-8 based data-warehouse and Oracle Express, a multidimensional OLAP server. At each site, the engine is responsible for building and updating related cubes ***incrementally*** by mining the data that flow into the data-warehouse daily. We have demonstrated the practical value of using OLAP servers as computation engines to perform data mining, and using the distributed OLAP infrastructure to scale data mining. We have also shown the power of the enhanced multilevel and multidimensional association rules for e-commerce applications.

OLAP technology has gained increasing popularity in data warehousing [1,2,5,8]. However, the issues regarding the use of OLAP servers as distributed

computation engines have not been studied. There exist a number of previous efforts on association rule mining from databases or other data sets [9,10,11]. Several of these efforts are based on cube structures using OLAP [6]. However, so far there is no work reported on mining scoped or functional association rules using OLAP. Although time-variant association rules have been studied, such rules fall into a special case of association rules with general conjoint items. Most significantly, our approach is based on the support of a distributed OLAP infrastructure to enhance the expressive power of association rules. We claim the novelty of our solution from this point of view.

The rest of this paper is organized as follows. Section 2 gives an overview of our distributed OLAP infrastructure. Section 3 describes customer profiling and shopping pattern analysis based on this infrastructure. Section 4 discusses our extension to multilevel and multidimensional association rules. Section 5 describes distributed and incremental rule mining. Finally in section 6 some conclusions are given.

2. Architecture

As shown in Figure 1, our infrastructure includes at least two layers of data warehouse/OLAP stations: LDOSs and a GDOS. The LDOSs are responsible for local data mining and summarization, while the GDOS is responsible for merging and mining the input data from LDOSs, and for providing the mining results to LDOSs for business applications such as personalized promotions, inventory management, etc.

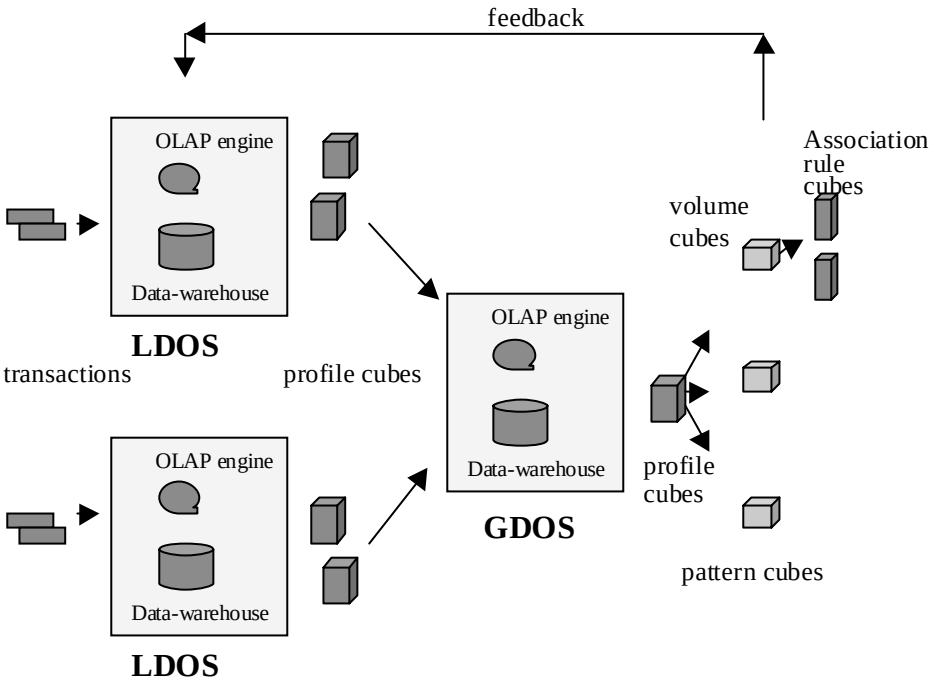

Figure 1: Distributed data-warehouse/OLAP based architecture

LDOS

A local data warehouse and one or more OLAP servers are maintained in each LDOS. The basic input data to a LDOS are transactions that are fed in daily and dumped to archive after use [4]. The basic output data sent to the GDOS periodically from an LDOS is, conceptually, a *Profile Snapshot Cube* (PSC), which contains partial information for customer profiling. Transaction data and related reference data are stored in the warehouse. Each OLAP server is used as a computation engine for building and *incrementally* updating PSC, by mining new transactions flowing into the local data-warehouse, and deriving patterns for local analysis. Loading transaction data into the data-warehouse, and then loading the warehoused data to the OLAP server for generating PSCs, is a periodic process (e.g. hourly or daily).

Besides feeding PSCs to the GDOS periodically, an LDOS also receives data mining results, such as global association rules or alerts, from the GDOS. It can access the GDOS to get rules and customer profiles when necessary.

GDOS

A global data warehouse and one or more OLAP servers are maintained in the GDOS. The GDOS has bi-directional communications with the LDOS. It combines the summary information from multiple LDOSs to build and *incrementally* update global customer profiles, association rules, etc, which cannot be completed at a single LDOS, and feeds back the resulting profiles, rules and other derived objects such as promotion plans, to the LDOSs. Multilevel and multidimensional customer shopping *patterns* may be extracted, analyzed, and compared. In our implementation, the volume cubes for generating association rules are extracted from cubes representing customer profiles. We shall describe this in more detail later.

In summary, the GDOS and the LDOSs must cooperate. The GDOS relies on the LDOSs to reduce data load and computation load for enhanced scalability; and the customer profiles and rules are generated at the GDOS by using the summary information fed by the LDOSs in combination. As we indicated before, more levels of stations may be introduced.

3. Customer Profiles and Shopping Patterns

We represent customer profiles and shopping patterns as cubes. A cube *C* has a set of underlying *dimensions* $D_1,\ldots, D_n$, and is used to represent a multidimensional *measure*. Each cell of the cube is identified by one value from each of the dimensions, and contains a value of the measure. We say that the measure is dimensioned by $D_1,\ldots, D_n$. The set of values of a dimension *D*, called the *domain* of *D*, may be limited (using the OLAP *limit* operation) to a subset. A sub-cube (slice or dice) can be derived from a cube *C* by dimensioning C by a subset of its dimensions, and/or by limiting the value sets of these dimensions.

3.1. Profile Cubes

A profile cube, say *PF*, contains profiling information of multiple customers, and has dimensions *kind, product, customer, merchant, time, area*. It is derived from the transaction data stored in the relational data warehouse. In Oracle Express Language it is defined as

define PF variable decimal <kind, sparse <product, customer, merchant, time, area>>

where dimension *kind* has values *'saleInUnits', 'saleInDollars', 'discounts', 'couponDiscounts', 'loyaltyDiscounts', 'paymentMethods'*, etc. Introducing this dimension allows us to represent multiple measures by the single cube. The decimal data type is used to cover all numerical data types. A specific integer measure may be derived from a profile cube, and the integer data type converted from the decimal data type. The use of keyword "sparse" in the above definitions instructs Oracle Express to create a composite dimension *<product, customer, merchant, time, area>*, in order to handle sparseness in an efficient way.

Profile cubes are maintained at both the GDOS and the LDOSs. At an LDOS, a local profile cube is populated by means of ***binning***. A transaction data record contains fields with values mapping to each dimension of the cube. Such a mapping is referred to as binning. For example, '01Jan98 8:44am' is mapped to time-bin '01Jan98'. A transaction occurring at '01Jan98 8:44am' and at the 'Safeway Market' in S.F. falls into the cell corresponding to *time* = '01Jan98' and *area* = 'S.F.'.

At the GDOS, a centralized profile cube with desired coverage in time, area etc, is retrieved from the database and updated by merging the appropriate local profile cubes, and then may be stored back to the database. In Oracle Express, expressing such a merge operation is straightforward. Let *PF* be the centralized cube and $PF_1,\ldots, PF_k$ the local cubes. Their merge is simply expressed as

$$PF = PF + PF_1 + \ldots + PF_k$$

In this way customer profiles are combined and updated incrementally as each new local cube flows into the GDOS.

3.2. Multilevel Pattern Representation

Shopping pattern cubes are derived from profile cubes and are used to represent the **shopping behavior** of *individual customers*. In order to represent shopping behavior at multiple levels, we define each of the dimensions to be a hierarchical dimension, along which the shopping pattern cubes can roll up.

A hierarchical dimension *D* contains values at different levels of abstraction. Associated with *D* there are a dimension *DL* describing the levels of *D*, a relation *DL_D* mapping each value of *D* to the appropriate level, and a relation *D_D* mapping each value of *D* to its parent value (the value at the immediate upper level). Let *D* be an underlying dimension of a numerical cube *C*. *D, DL, DL_D* and *D_D*, fully specify a dimension

hierarchy. They provide sufficient information to roll up cube *C* along dimension *D*; that is, to calculate the total of the cube's measure values at the upper levels using the corresponding lower-level measure values. A cube may be rolled up along multiple dimensions. In the applications discussed in this paper, the following hierarchies are introduced. The **product hierarchy** is made up of the following objects.

- *product*: dimension with values at the product level (e.g. 'HP Inkjet500'), product_category level (e.g. 'printer'), etc, and a special value 'top' at top-level.
- prodLevel: dimension with values 'prod_item', 'prod_category', 'prod_kind', 'top'.
- prod_prod: parent relation (product, product) mapping each value to its parent, e.g.
  prod_prod(product 'HP Inkjet500') = 'printer'
  prod_prod(product 'top') = NA
- prodLevel_prod: level relation (product, prodLevel) mapping each value to its level, e.g.
  prodLevel_prod(product 'HP Inkjet500')='prod_item'
  prodLevel_prod(product 'printer')='comp peripheral'

Analogously, the **merchant hierarchy** is made up of dimension *merchant*; dimension *mercLevel* with values 'store', 'store_category'and 'top'; parent relation *merc_merc* and level relation *mercLevel_merc.* The **customer hierarchy** is made up of dimension *customer*; dimension *custLevel* with values 'shopper', 'shopper_group', 'shopper_category'and 'top'; parent relation *cust_cust* and level relation *custLevel_cust.* The **time hierarchy** is made up of dimension *time*; dimension *timeLevel* with values 'day', 'month', 'year'and 'top'; parent relation *time_time* and level relation *timeLevel_time.* The **area hierarchy** is made up of dimension *area*; dimension *areaLevel* with values 'city', 'state', 'region' and 'top'; parent relation *area_area* and level relation *areaLevel_area.*

3.3. Deriving Shopping Pattern Cubes

Various shopping patterns can be derived from profile cubes and handled as shopping pattern cubes. They may be used to represent the shopping behavior of a collection of customers or a single customer; they may be based on volumes or probability distributions; and they may be materialized (defined as variables) or not (defined as formulas).

Multiple or single customer based patterns

For example, a cube representing a single measure, *SaleUnits* may be defined as a formula (view) of the above profile cube *PF* by the following.

define SaleUnits formula (PF(kind 'saleInUnits'))
int <product, customer, merchant, time, area>

A cube representing the same measure for a single customer, say, 'Doe', is defined as

define SaleUnits.1 formula (PF(kind 'saleInUnits', customer 'Doe')) int <product, merchant,time, area>

Volume and probability based patterns

In a volume-based shopping pattern cube, each cell value is a quantitative measure. The cube may be rolled up along its hierarchical dimensions. For example, cube *SaleUnits* defined above can be rolled up along dimension *product* using relation prod_prod, and analogously, along dimensions *customer, merchant, time, area.* Then, for example, cell

SaleUnits(product 'pen', customer 'John Doe', merchant 'Sears', time '01Jan98', area 'San Francisco')

gives the number of pens purchased by John Doe at Sears in San Francisco on 01Jan98. Cell

SaleUnits.Doe(product 'pen', customer 'John Doe', merchant 'top', time 'top', area 'top')

gives the number of pens purchased by John Doe anywhere and anytime covered by the profiling period and area.

Cubes representing probability distribution based shopping patterns are derived from volume-based pattern or profile cubes. They provide more fine-grained representation of dynamic behavior than fixed value based ones. They also allow shopping patterns corresponding to different lengths of profiling interval to be compared. For example, the following cube, *PLD,* represents the probability distributions of loyalty discounts over all discounts along multiple dimensions.

define PLD variable decimal <customer, product, merchant, time, area>

It can be calculated through the following steps.

- Define cubes *LoyaltyDiscount* and *Discount* over dimensions *customer, product, merchant, time, area*
- Populate *LoyaltyDiscount* and *Discount* as sub-cubes of *PF* with dimension *kind* limited to "LoyaltyDiscount" and "Discount" respectively
- Rollup *LoyaltyDiscount* and *Discount* along dimension *product, merchant, time* and *area*
- Then *PLD = LoyaltyDiscount / Discount* (please note this is a cell-wise operation)

For efficiency as well as consistency, it is only necessary to store profile cubes persistently in the data-

warehouse. Shopping patterns, either based on volume or probability, can be derived on the fly (at analysis time) using the OLAP engine for computation. For storing, combining and updating profile cubes, only the bottom level of each dimension is necessary. Shopping pattern cubes are allowed to rollup along any hierarchical dimension. This shows the simplicity, and yet the power, of using OLAP to handle customer profiling.

4. Extended Association Rules

An important application of our distributed data-warehouse/OLAP infrastructure is to enhance association rule mining. In e-commerce, association rules benefit both merchants and customers. However, association rules are created and incrementally updated from hundreds of millions of transaction records generated daily. Our infrastructure allows us to distribute data processing, reduce data volumes at each local site by summarization, and mine data incrementally at multiple levels. In addition to scaling up rule mining, this infrastructure also allows us to combine information from different locations for generating association rules with enhanced expressive power. In this section we will extend multilevel and multidimensional association rules by introducing *scoped association rules*, *association rules with conjoint items* and *functional association rules*, and we will show how to compute these rules using data cubes and OLAP.

4.1. Scoped Multidimensional and Multilevel Association Rules

Association rules provide a quantitative measurement of the association relationships between facts. Association rule mining aims at inferring such relationships from transaction summary data. For example, a cross-sale association rule is used to answer "how many customers who bought product A, also bought product B in one month?" An association rule can be simply expressed by $X \Rightarrow Y$, where X is its ***antecedent***, and Y is its ***consequent*** and they are conjunctive predicates. Related to each association rule there is a **confidence** and a **support**. In the above example, if 80% of the customers who bought A also bought B, and only 10% of all the customers bought both, we say that the association rule has confidence 80% and support 10%. Given application-specific minimum support and confidence thresholds, a rule is considered *strong* if it satisfies these thresholds.

An association rule has an underlying **base** B that contains a population over which the rule is defined. For example, the above cross-sale association rule can be based on transactions, as

$$x \in \textit{Transactions: contain\_product}(x,A) \Rightarrow \textit{contain\_product}(x, B),$$

or based on customers, as

$$x \in \textit{Customers: buy\_product}(x, A) \Rightarrow \textit{buy\_product}(x, B),$$

regardless of whether the purchases were made in the same transaction or not. In this example, the association rule uses binary predicates with the first place denoting a base element and the second place denoting an item. In these examples, the items are elements of a set of products.

The notion of base is especially important for cooperative rule mining between GDOS and LDOSs. For example, if the customers of interest shop at several locations covered by different LDOSs, then customer based cross-sale association rules should be mined at the GDOS on the summary data fed from multiple LDOSs. If the customers of interest are partitioned geographically and covered by individual LDOSs, such rules can be first mined locally at each LDOS and then assembled in the GDOS.

For the kind of application we are considering in this paper, we provide association rules with different bases, and we refer to such rules as **scoped** association rules. For a rule

$$B: X \Rightarrow Y,$$

let us denote the set of *base elements* in B that match X by P_X, and its cardinality by $|P_X|$. The **confidence** of rule $X \Rightarrow Y$ in B, denoted by $\xi_B(X \Rightarrow Y)$, can be calculated by $\xi_B(X \Rightarrow Y) = |P_X \cap P_Y| / |P_X|$, and ranges from 0 to 1. The **support** of rule $X \Rightarrow Y$ in B, denoted by $\theta_B(X \Rightarrow Y)$, can be calculated by $\theta_B(X \Rightarrow Y) = |P_X \cap P_Y| / |B|$. For simplicity, when the base B is understood from the context, we drop the suffix B from ξ and θ.

We also provide *multidimensional* association rules, e.g.

$$[x \in \textit{Customers: buy\_product}(x,\text{'A'}) \Rightarrow \textit{buy\_product}(x,\text{'B'})] \mid \textit{merchant} = \text{'Sears'}, \textit{area} = \text{'Los Angeles'}, \textit{time} = \text{'Jan98'}$$

In this example, *customer* is the **base**, *products* are the **items**, and *merchant*, *area* and *time* are underlying **features** of the rule. Essentially, the base of multidimensional rules are dimensioned by the features.

Further, a feature may be represented at *multiple levels*. For example, an *area* may be represented at the city level (e.g. San Francisco) or at the state level (e.g.

California); a time may be represented at the day, month or year levels. Accordingly, association rules at different *area* levels and *time* levels can be specified, e.g.

[x∈Customers: buy_product(x,'A') ⇒ buy_product(x,'B')]
| merchant = 'Sears',area='Los Angeles', time = 'Jan98'

[x∈Customers: buy_product(x,'A') ⇒ buy_product(x,'B')]
*| merchant = 'Sears', area ='**California**', time = 'Jan98'*

[x∈Customers: buy_product(x,'A')⇒ buy_product(x,'B')]
*| merchant = 'Sears', area ='**California**', time = '**Year98**'*

For multidimensional association rules, the cardinalities, confidence and support are dimensioned by, or as functions of, the features.

4.2. Cube-based Association Rule Mining

We represent association rules by cubes. We will use *volume cubes*, where cell values are counts, for deriving and measuring multidimensional $|P_X \cap P_Y|$, $|P_X|$, and $|B|$ in the intermediate steps of computing multidimensional association rules.

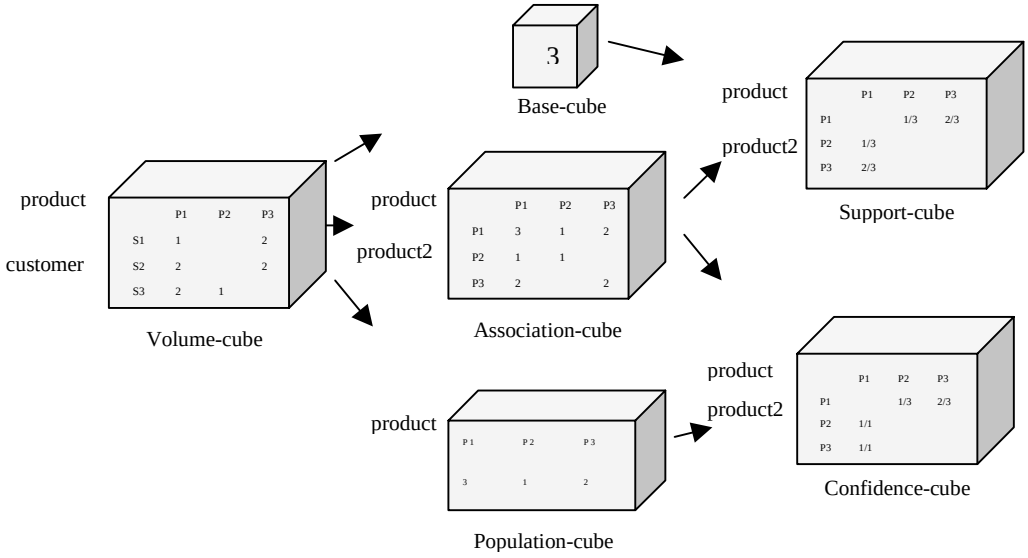

Figure 2: Slices of a list of cubes related to cross-sale association rules

Volume cube

A volume cube contains multidimensional counts for deriving association rules. For example, cross-sale association rules are derived from volume cubes such as

SaleUnits(customer, product, merchant, time, area)

Each cell contains the number of purchased units dimensioned by *customer*, *product*, *merchant*, *time* and *area*. This cube is derived from a profile cube and materialized for mining cross-sale association rules.

Volume cubes such as *SaleUnits* are maintained at both GDOS and LDOSs. At a LDOS, a local cube is populated from transaction data. At the GDOS, a centralized cube with desired coverage in time, area, etc, is retrieved from the database and updated by merging local cubes.

The dimensions of a volume cube, as well as other association rule related cubes, can be classified into the following categories for the purpose of association rule mining.

- **Item dimensions** on which volume data for quantifying the association relationship are counted, such as the *product* dimension in the above example.
- **Base dimensions** on which rules are quantified such as the *customer* dimension. A volume cube may roll up along its hierarchical dimensions, but not along a base dimension. For example, aggregating the number of purchased products along the *customer* dimension to a high-level *customer* value, say, 'engineer', may not be meaningful to deriving cross-sale associations, e.g. if one engineer buys milk and another buys eggs, this does not imply any meaningful cross-sale association.

- **Feature dimensions** on which the generated rules may be dimensioned such as *merchant*, *time* and *area*.

A volume cube C_v is sufficient for deriving the instances of rule $X \Rightarrow Y$ if it has a **base dimension** that represents the base of the rule, and the **association conditions** for qualifying $X \wedge Y$ are definable on C_v. For deriving cross-sale association rules from cube *SaleUnits*, an association condition can be

for each base and feature dimension,
C_v*(product A)* $> 0 \wedge C_v$*(product B)* > 0

If the association conditions used to compute multidimensional $|P_X \cap P_Y|$ are definable on C_v, then another kind of condition, called **antecedent conditions,** that are used to compute multidimensional $|P_X|$, are also definable on C_v, such as

for each base and feature dimension,
C_v*(product A)* > 0

Association cube

The association cube C_a for rule $X \Rightarrow Y$ gives a volume-based measure of multidimensional association relationships that are computed from the volume cube C_v, and is used to derive the confidence cube and the support cube of association rules. More exactly, it maintains dimensioned $|P_X \cap P_Y|$, *i.e.* the number of base elements that satisfy $X \wedge Y$.

Usually C_a is dimensioned differently from C_v. In the cross-sale association rule example, the association cube is defined as

CrossSales (*product, product2, customer_group, merchant, time, area****)***

A cell of this cube, CrossSales(*product* 'A', *product2* 'B', *customer_group* 'engineer', *merchant* 'Sears', *time* 'Jan98', *area* 'Los Angeles') = 4500 means that there are 4,500 customers who are engineers, who bought item A as well as item B, at a Sears store in Los Angeles in Jan98.

For an association cube, the **item dimensions** underlie the counts for deriving association rules, such as dimensions *product* and *product2* for the above *CrossSales* cube. The dimension *product2* has the same set of values as *product*, and we call it the **mirror dimension** of *product*. We introduce a mirror dimension simply because the cross-sale association rule involves more than one element of the item dimension.

The **base dimension**, such as the *customer* dimension, underlies the base of rules. Unlike the volume cube, the association cube does not necessarily have to be dimensioned by the base dimension. However, we can dimension rules by a derived dimension, each value of which identifies a group of base dimension values at bottom levels. In the cube *CrossSales* shown above, we introduce the hierarchical dimension *customer_group*, which has levels 'customer_profession', 'customer_category' and 'top'. A relation is also defined for relating customers and customer groups. For example, a value of the derived *customer_group* dimension, say, "engineer", is used to identify a group of individual customers who are engineers.

An association can cube also have underlying **feature dimensions** such as *merchant*, *time* and *area*.

Population cube and base cube

The population cube C_p and the base cube C_b for rule $X \Rightarrow Y$ are also derived from the volume cube C_v. C_p is used to measure dimensioned $|P_X|$, *i.e.* the numbers of base elements satisfying X. C_b is used to represent dimensioned $|B|$. For the above cross-sale rules, the population cube is defined as

NumOfBuyers (product, customer_group, merchant, time, area)

A cell of this cube, ***NumOfBuyers(****product* 'A', *customer_group* 'engineer', *merchant* 'Sears', *time* 'Jan98', *area* 'Los Angeles') = 10000 means that there are 10,000 customers who are engineers, and who bought item A in Los Angeles in Jan98. The base cube is defined as

NumOfShoppers (customer_group, merchant, time, area)

Note that ***NumOfShoppers*** is not aggregated from ***NumOfBuyers***, as a single customer may buy multiple products.

Confidence cube and support cube

The confidence of rule $X \Rightarrow Y$, defined as $|P_X \cap P_Y| / |P_X|$, and the support, defined as $|P_X \cap P_Y| / |B|$, are represented as cubes C_f and C_s. C_f is derived from C_a and C_p; and C_s is derived from C_a and C_b. They have the same dimensions as C_a. For the above cross-sale rules, the confidence cube and support cube are defined as

*Confidence (product, product2, customer_group, merchant, time, area****)***

Support (product, product2, customer_group, merchant, time, area)

Figure 2 shows the cubes related to cross-sale association rules, with one slice of each cube. The volume-cube is generated from transactions; the

association-cube, base-cube and population-cube are derived from the volume cube; the confidence-cube is derived from the association cube and population cube; and the support-cube is derived from the association-cube and base-cube. The slices of these cubes shown in Figure 2 correspond to the same list of values in dimension *merchant, time, area* and *customer_group*.

Multidimensional and multilevel rules

Representing association rules by cubes, and underlying cubes by hierarchical dimensions, naturally supports multidimensional and multilevel rules. Also, these rules are well organized and can be easily queried.

First, the cells of an association cube with different dimension values are related to association rule instances in different scopes. In the association cube *CrossSales*, cell

CrossSales(product 'A', product2 'B', customer_group 'engineer', merchant 'Sears', area 'Los Angeles', time 'Jan98')

represents the following multidimensional rule

[x∈Customers: buy_product(x, 'A') ⇒ buy_product(x,'B')] | customer_group = 'engineer', merchant = 'Sears', area = 'Los Angeles', time = 'Jan98'

If this cell has value 4500, and the corresponding cell in the population cube has value 10000, then this rule has confidence 0.45.

Next, as the cubes representing rules can have hierarchical dimensions, they represent not only multi-dimensional but also multi-level association rules. For example, the following cells

*CrossSales(product 'A', product2 'B', customer_group 'engineer', merchant 'Sears', area '**California**', time 'Jan98')*

*CrossSales(product 'A', product2 'B', customer_group 'engineer', merchant 'Sears', area '**California**', time '**Year98**')*

represent association rules at different *area* levels (i.e., the city level and the state level) and different *time* levels (i.e., the month level, the year level):

*[x∈Customers: buy_product(x, 'A') ⇒ buy_product(x, 'B')] | customer_group = 'engineer', merchant = 'Sears', area= '**California**', time = 'Jan98'*

*[x∈Customers: buy_product(x, 'A') ⇒ buy_product(x, 'B')] | customer_group = 'engineer', merchant = 'Sears', area= '**California**', time = '**Year98**'*

The cell

CrossSales(product 'A', product2 'B', customer_group 'top', merchant 'top', area 'top', time 'top')

represents the customer-based cross-sale association rule for all customers, merchants, areas, and times in the given range of these dimensions, expressed as

[x∈Customers: buy_product(x, 'A') ⇒ buy_product(x, 'B')]

4.3. Generating Association Rule Related Cubes

The basic task of our OLAP based association rule mining framework, either at the GDOS or at an LDOS, is to convert a ***volume cube***, i.e. the cube representing the purchase volumes of customers dimensioned by *product, area*, etc, into an ***association cube***, a ***base cube*** and a ***population cube***. These cubes are then used to derive the ***confidence cube*** and the ***support cube*** of multidimensional association rule instances. The following general steps are involved in cross-sale association rule mining.

- Roll up the volume cube **SaleUnits** by aggregating it along *merchant, time, area* dimensions.
- Derive cube **NumOfBuyers** from **SaleUnits** based on the antecedent condition **SaleUnits** > 0.
- Populate cube **NumOfShoppers** by the counts of customers dimensioned by *merchant, area, time* (not by product) that satisfy the antecedent conditions.
- Derive cube **CrossSales** from **SaleUnits** based on the association conditions **SaleUnits**(*product* p_1) > 0 and **SaleUnits**(*product2* p_2) > 0.
- Derive cube **Confidence** and cube **Support** using cell-wise operations:
  - **Confidence = CrossSales / NumOfBuyers**
  - **Support = CrossSales / NumOfShoppers**

Cubes **Confidence**, **Support**, **CrossSales** are dimensioned by *product, product2, customer_group,merchant, time, area;* **NumOfBuyers** is dimensioned by *product, customer_group, merchant, time, area*; **NumOfShoppers** is dimensioned by *customer_group, merchant, time, area).*

Rules with confidence and support that exceed specified thresholds, may be considered interesting.

4.4. Rules with Conjoint Items

Cubes with conjoint dimensions can be used to represent refined multidimensional association rules. For example, using OLAP, we can derive *association rules across time*. **Time-variant**, or **temporal** association rules such as

*[x∈Customers: buy_product(x,' A', '**Jan98'**) ⇒ buy_product(x, 'B', '**Feb98'**)] | area = 'Los Angeles'*

can be used to answer such questions as "*How are the sales of B in Feb98 associated with the sales of A in Jan98?*" The items in this rule are value pairs of dimensions *product* and *time.*

In order to specify this kind of association rule, we introduce a conjoint dimension <*product, time*>, and mirror it with dimension <*product2, time2*>. This allows a cell in the association cube to cross two time values. Accordingly, the cubes related to association rule mining are defined as follows.

Association cube
*CrossSales.2 (<product, time>, <product2, time2>, customer_group, merchant, area**)***
Population cube
NumOfBuyers.2 (<product, time>, customer_group, merchant, area)
Base cube
NumOfShoppers.2 (customer_group, merchant, area)
Confidence cube
Confidence.2 (<product, time>, <product2, time2>, customer_group, merchant, area)
Support cube
*Support.2 (<product, time>, <product2, time2>, customer_group, merchant, area**)***

The steps for generating these cubes are similar to the ones described before. The major differences are that a cell is dimensioned by, besides others, <*product, time*> and <*product2, time2*>, and the template of the association condition is

SaleUnits(<product p_1, time t_1>) > 0 and
SaleUnits(<product2 p_2 ,time2 t_2>) > 0

where, in any instance of this condition, the time expressed by the value of *time2*, is not contained in the time expressed by the value of *time.* The template of the antecedent condition is

SaleUnits(<product p_1, time t_1>) > 0

In general, other dimensions such as *area* may be added to the conjoint dimensions to specify more refined rules.

4.5. Functional Association Rules

A multidimensional association rule is functional if its predicates include variables, and the variables in the consequent are functions of those in the antecedent. For example, functional association rules can be used to answer the following questions, where a_month and a_year are variables.

- What is the percentage of people in California who buy a printer in the next month after they bought a PC?

  [x∈Customer: buy_product(x, 'PC', a_ month) ⇒ buy_product(x, 'printer', a_month+1)] | area = 'California'

- What is the percentage of people who buy a printer within the year when they bought a PC?

  [x∈Customer: buy_product(x, 'PC', a_ year) ⇒ buy_product(x, 'printer', a_year)] | area = 'California'

To be distinct, we call the association rules that are not functional as **instance** association rules; e.g.,

[x∈Customer: buy_product(x,' PC', 'Jan98')⇒ buy_product(x,' printer', 'Feb98')] | area = 'California'

Time variant, functional association rules can be derived from time variant, instance association rules through cube restructuring. Let us introduce a new dimension *time_delta* that has values one_day, two_day, …, at the day level, and values one_month, two_month, …, at the month level, etc. Then, let us consider the following functional association rule related cubes.

Association cube
CrossSales.3 (product, product2, customer_group, merchant, area, time_delta)
Population cube
NumOfBuyers.3 (product, customer_group, merchant, area)
Base cube
NumOfShoppers.3 (customer_group, merchant, area)
Confidence cube
Confidence.3 (product, product2, customer_group, merchant, area, time_delta)
Support cube
Support.3 (product, product2, customer_group, merchant, area, time_delta)

The association cube ***CrossSales.3*** can be constructed from ***CrossSales.2***. The cell values of ***CrossSales.2*** in the selected *time* and *time2* ranges are added to the corresponding cells of ***CrossSales.3***. For example, the count value in cell

CrossSales.2(<PC, Jan98>, <printer, Feb98>…)

is added to cell (bin)

CrossSales.3(PC, printer, one_month,…)

It can also be added to cell
CrossSales.3(PC, printer, one_year,…)

5. Distributed and Incremental Rule Mining

There exist two ways to deal with association rules:

- *Static*, that is, to extract a group of rules from a snapshot, or a history, of data and use "as is".
- *Dynamic*, that is, to evolve rules from time to time using newly available data.

We mine association rules from an e-commerce data warehouse holding transaction data. The data flows in continuously and is processed daily.

Mining association rules dynamically has the following benefits.

- "Real-time" data mining, that is, the rules are drawn from the latest transactions for reflecting the current commercial trends.
- Multilevel knowledge abstraction, which requires summarizing multiple partial results. For example, association rules on the month or year basis cannot be concluded from daily mining results. In fact, multilevel mining is incremental in nature.
- For scalability, incremental and distributed mining has become a practical choice.

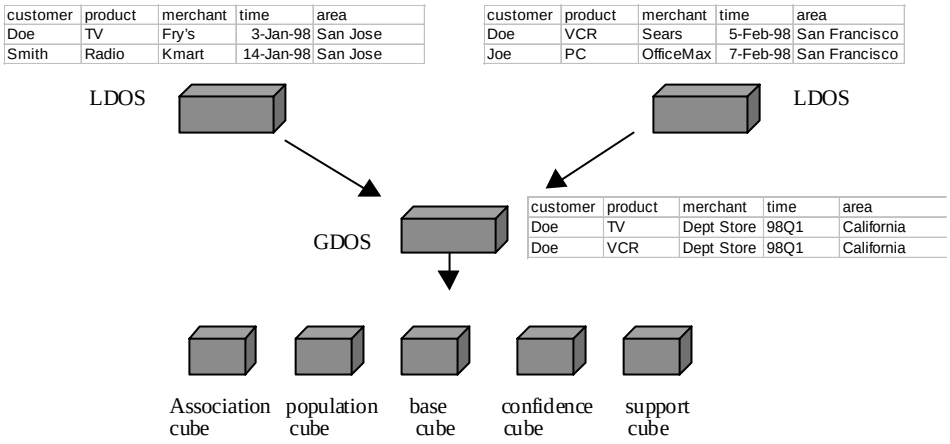

Figure 3: Distributed rule mining

Incremental association rule mining requires combining partial results. It is easy to see that the confidence and support of multiple rules may not be combined directly. This is why we treat them as "views" and only maintain the association cube, the population cube and the base cube that can be updated from each new copy of volume cube. Below, we discuss several cases to show how a GDOS can mine association rules by incorporating the partial results computed at LDOSs.

- The first case is to sum up volume-cubes generated at multiple LDOSs. Let $C_{v,i}$ be the volume-cube generated at $LDOS_i$. The volume-cube generated at the GDOS by combining the volume-cubes fed from these LDOSs is $C_v = \sum_{i=1}^{n} C_{v,i}$. The association rules are then generated at the GDOS from the centralized C_v.
- The second case is to mine local rules with distinct *bases* at participating LDOSs, resulting in a local association cube $C_{a,I,}$ a local population cube $C_{p,I}$, and a local base cube $C_{b,i}$ at each LDOS. At the GDOS, multiple association cubes, population cubes and base cubes sent from the LDOSs are simply combined, resulting in a summarized association cube and a summarized population cube, as $C_a = \sum_{i=1}^{n} C_{a,i}$, $C_p = \sum_{i=1}^{n} C_{p,i}$ and $C_b = \sum_{i=1}^{n} C_{b,i}$. The corresponding confidence cube and support cube can then be derived as described earlier. Cross-sale association rules generated from distinct customers belong to this case.

In general, it is inappropriate to directly combine association cubes that cover areas $a_1,\ldots, a_k$, to cover a larger area a. In the given example, this is because association cubes record counts of customers that satisfy

the association condition, and the sets of customers contained in $a_1,\ldots, a_k$ are not mutually disjoint. This can be seen in the following examples.

- A customer who bought A and B in both San Jose and San Francisco which are covered by different LDOSs , contributes a count to the rule covering each city, but has only one count, not two, for the rule $A \Rightarrow B$ covering California.
- A customer (e.g. Doe in Figure 3) who bought a TV in San Jose, but a VCR in San Francisco, is not countable for the cross-sale association rule $TV \Rightarrow VCR$ covering any of these cities, but countable for the rule covering California. This is illustrated in Figure 3.

6. Conclusions

In order to scale-up association rule mining in e-commerce, we have developed a distributed and cooperative data-warehouse/OLAP infrastructure. This infrastructure allows us to generate association rules with enhanced expressive power, by *combining information of discrete commercial activities* from different geographic areas, different merchants and over different time periods. In this paper we have introduced *scoped association rules*, *association rules with conjoint items*; and *functional association rules* as useful extensions to association rules.

The proposed infrastructure has been designed and prototyped at HP Labs to support business intelligence applications in e-commerce. Our preliminary results validate the scalability and maintainability of this infrastructure, and the power of the enhanced multilevel and multidimensional association rules. In this paper, we did not discuss privacy control in customer profiling. However, we did address this issue in our design by incorporating support for the P3P protocol [12] in our data warehouse. We plan to integrate this framework with a commercial e-commerce system.

References

[1] Sameet Agarwal, Rakesh Agrawal, Prasad Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, Sunita Sarawagi, "On the Computation of Multidimensional Aggregates", 506-521, Proc. VLDB'96, 1996.

[2] Surajit Chaudhuri and Umesh Dayal, "An Overview of Data Warehousing and OLAP Technology", SIGMOD Record Vol (26) No (1) , 1996

[3] Qiming Chen, Umesh Dayal, Meichun Hsu, "OLAP-based Scalable Profiling of Customer Behavior", Proc. Of 1st International Conference on Data Warehousing and Knowledge Discovery (DAWAK99), 1999, Italy.

[4] Hector Garcia-Molina, Wilburt Labio, Jun Yang, "Expiring Data in a Warehouse", Proc. VLDB'98, 1998.

[5] J. Han, S. Chee, and J. Y. Chiang, "Issues for On-Line Analytical Mining of Data Warehouses", SIGMOD'98 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98) , USA, 1998.

[6] J. Han, "OLAP Mining: An Integration of OLAP with Data Mining", Proc. IFIP Conference on Data Semantics (DS-7), Switzerland, 1997.

[7] Raymond T. Ng, Laks V.S. Lakshmanan, Jiawei Han, Alex Pang, "Exploratory Mining and Pruning Optimizations of Constrained Associations Rules", Proc. ACM-SIGMOD'98, 1998.

[8] Torben Bach Pedersen, Christian S. Jensen, "Multidimensional Data Modeling for Complex Data", Proc. ICDE'99, 1999.

[9] Sunita Sarawagi, Shiby Thomas, Rakesh Agrawal, "Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications", Proc. ACM-SIGMOD'98, 1998.

[10] Hannu Toivonen, "Sampling Large Databases for Association Rules", 134-145, Proc. VLDB'96, 1996.

[11] Dick Tsur, Jeffrey D. Ullman, Serge Abiteboul, Chris Clifton, Rajeev Motwani, Svetlozar Nestorov, Arnon Rosenthal, "Query Flocks: A Generalization of Association-Rule Mining" Proc. ACM-SIGMOD'98, 1998.

[12] P3P Architecture Working Group, "General Overview of the P3P Architecture", P3P-arch-971022, http://www.w3.org/TR/WD-P3P.arch.html, 1997.