

# OLAP-based Scalable Profiling of Customer Behavior

Qiming Chen, Umesh Dayal, Meichun Hsu

HP Labs, Hewlett-Packard, 1501 Page Mill Road, MS 1U4, Palo Alto, CA 94303, USA  
{qchen,dayal,mhsu}@hpl.hp.com

**Abstract.** Profiling customers' behavior has become increasingly important for many applications such as fraud detection, targeted marketing and promotion. Customer behavior profiles are created from very large collections of transaction data. This has motivated us to develop a data-warehouse and OLAP based, scalable and flexible profiling engine. We define profiles by probability distributions, and compute them using OLAP operations on multi-dimensional and multilevel data cubes. Our experience has revealed the simplicity and power of OLAP-based solutions to scalable profiling and pattern analysis.

## 1 Introduction

Profiling customers' behavior aims at extracting patterns of their activities from transactional data, and using these patterns to provide guidelines for service provisioning, trend analysis, abnormal behavior discovery, etc. It has become increasingly important in a variety of application domains, such as fraud detection, personalized marketing and commercial promotion. It has also given rise to the need for a scalable infrastructure to support filtering, mining and analyzing massive transaction data continuously [1],[2],[4]. We have developed such an infrastructure with data-warehousing and OLAP technology.

In this paper we will focus on the construction and application of customer behavior profiles from telephone call data for the purpose of fraud detection. Typically, a customer's calling behavior is represented by the composition and periodic appearance of his call destination, time-window and duration. One way of doing fraud detection is to discover abnormal calling behavior, which may be further classified into the following two categories.

**Threshold based fraud detection.** For example

- ? a call is suspicious if its duration  $> 24$  hours,
- ? a call is suspicious if its duration  $> 4$  hours and it is made in the evening.

**Pattern based fraud detection.** For example,

- ? a caller (identified by phone number) is suspicious if his calling pattern is similar to a previously known fraudulent one.

Profiling callers' behavior is significant for both kinds of fraud detection. In threshold based fraud detection, without information about personalized calling behavior,

only generalized thresholds may be set, such as to consider a call to be suspicious if it lasts over 24 hours. With the availability of a customer's calling behavior profile, *personalized*, rather than generalized thresholds can be set, so that

? calls by John for 4 hours are considered usual, but

? calls by Jane for 2 hours are considered unusual.

Thus, personalized or group-based thresholds can be used to provide more precise fraud detection than generalized thresholds. Similarly, pattern based fraud detection is based on profiles to do pattern matching. Each new customer's calling behavior is profiled and compared against known fraudulent profiles. Customer profiles are also useful for other summary information oriented applications.

To create and update customer behavior profiles, hundreds of millions of call records must be processed everyday. This has motivated us to develop a scalable and maintainable framework to support such profiling. The profiling engine is built on top of an Oracle-8 based telecommunication data-warehouse and Oracle Express, a multidimensional OLAP server. Profiles and calling patterns are represented as multidimensional cubes and based on the *probability distribution* of call volumes. The profiling engine is capable of building and updating customer calling behavior profiles *incrementally* by mining call records that flow into the data-warehouse daily, deriving *calling patterns* from profiles, analyzing and comparing the *similarity* of calling patterns. We have demonstrated the practical value of using an OLAP server as a scalable computation engine to support profile computation, maintenance and utilization.

We share the same view as described in [1] and [5], in taking advantage of OLAP technology for analyzing data maintained in data-warehouses. Particularly, we are in-line with the efforts described in [5] to use OLAP tools to support large-scale data mining. However, to our knowledge, there is no prior work reported on OLAP based customer behavior profiling and pattern analysis.

Section 2 introduces the concept of behavior profiling with probability distributions. Section 3 describes the architecture of our profiling engine. Section 4 illustrates how to compute profile cubes, analyze and compare calling pattern cubes. Finally in section 5 some conclusions are given.

## **2 Probability Distribution based Profiling with OLAP**

For customer behavior profiling, we first have to decide which features (dimensions) are relevant. For our calling behavior profiling application, the features of interest are the phone-numbers, volume (the number of calls), duration, time of day, and day of week for a customer's outgoing and incoming calls. Next, we have to select the granularity of each feature. Thus, the time of day feature may be represented by the time-bins 'morning', 'afternoon', 'evening' or 'night'; the duration feature may be represented by 'short' (shorter than 20 minutes), 'medium' (20 to 60 minutes), or 'long' (longer than 60 minutes). Finally, we have to decide the profiling interval (e.g. 3 months) over that the customer profiles will be constructed, and the periodic-

ity of the profiles (e.g. weekly). Thus, in our application, a customer's profile is a weekly summarization of his calling behavior during the profiling interval.

Based on the profiled information, *calling patterns* of individual customers may be derived. Conceptually we can consider the following three kinds of calling patterns.

A *fixed-value based calling pattern* represents a customer's calling behavior with fixed values showing his "average" behavior. For example, a calling pattern from number A to number B says that on the average calls are short in afternoons and long in evenings.

A *volume based calling pattern* summarizes a customer's calling behavior by counting the number of calls of different duration in different time-bins. For example, a calling pattern from number A to number B says that there were 350 short calls in the mornings of the profiling period, etc.

A *probability distribution based calling pattern* represents a customer's calling behavior with probability distributions. For example, a calling pattern from number A to number B says that 10% of the calls in the morning were long, 20% were medium, 70% were short.

Probability distribution based calling patterns provide more fine-grained representation of dynamic behavior than fixed value based ones. They also allow calling patterns corresponding to different lengths of profiling interval to be compared.

We represent profiles and calling patterns as *cubes*. A cube has a set of underlying *dimensions*, and each cell of the cube is identified by one value from each of these dimensions. The set of values of a dimension *D*, called the *domain* of *D*, may be limited (by 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.

As mentioned above, the profile of a customer is a *weekly* summarization of his activities in the *profiling period*. For efficiency in our prototype system we group the information for profiling multiple customers' calling behavior into a single *profile cube* with dimensions  $\langle \textit{duration}, \textit{time}, \textit{dow}, \textit{callee}, \textit{caller} \rangle$ , where *dow* stands for *day\_of\_week* (e.g. Monday,..., Sunday), *callee* and *caller* are calling and called phone numbers. The value of a cell in a profiling cube measures the volume, i.e. number of calls, made in the corresponding duration-bin, time-bin in a day, and day of week, during the profiling period. In this way a profile cube records multiple customers' outgoing and incoming calls week by week. From such a multi-customer profile cube, *calling pattern cubes* of individual customers may be derived. They have similar dimensions as the profile cubes except that a calling pattern cube for outgoing calls is not dimensioned by *caller*, and a calling pattern cube for incoming calls is not dimensioned by *callee*, because they pertain to a single customer.

Multiple calling pattern cubes may be generated to represent a customer's calling behavior from different aspects. In our design, several calling pattern cubes representing *probability-based information* are actually derived from intermediate calling pattern cubes representing *volume-based information*.

Let us consider a volume-based cube *V* for a single customer derived from the above profile cube by totaling outgoing calls over days of week. *V* holds the counts of

calls during the profiling period dimensioned by  $\langle time, duration, callee \rangle$ , where dimension *time* has values 'morning', 'evening', etc; *duration* has values 'short', 'long', etc; dimension *callee* contains the called phone numbers. A cell in the cube is identified by one value from each of these dimensions. From cube *V* the following different probability cubes (and others) may be generated:

?  $C_{pri}$  for the prior probability of time-bin of calls wrt each callee, that is dimensioned by  $\langle time, callee \rangle$ , and indicates the percentage of calls made in 'morning', 'afternoon', 'evening' and 'night' respectively.

?  $C_p$  for the conditional probability of call duration-bin given time-bin of calls wrt each callee, that is dimensioned by  $\langle time, duration, callee \rangle$ , indicates the percentage of calls that are 'long', 'medium' and 'short' respectively, given the time-bin.

?  $C_{con}$  for the probabilistic consequence of the above, i.e. the probability of calls in every cell crossing dimensioned by  $\langle time, duration, callee \rangle$  over the total calls.

All the above probability cubes,  $C_{pri}$ ,  $C_p$ , and  $C_{con}$ , can be derived from cube *V* using OLAP operations. In the Oracle Express OLAP language, these are expressed as

?  $C_{pri} = total(V, time, callee) / total(V, callee)$

?  $C_p = (V / C_{pri}) / total(V, callee)$

?  $C_{con} = V / total(V, callee)$

In the above expressions, total is a typical OLAP operation on cubes with numerical cell values. While  $total(V)$  returns the total of the cell values of *V*,  $total(V, callee)$  returns such a total dimensioned by callee,  $total(V, time, callee)$  returns such a total dimensioned by time and callee. In fact a dimensioned total represents a cube. The arithmetic operations on cubes, such as '/' used above, are computed cell-wise.

With the above mechanism it is only necessary to make volume cubes persistent data-warehouse objects. In the other words, only the volume based information need to be profiled; calling patterns, either based on volume or probability, can be derived.

### 3 Architecture of the Profiling Engine

The profiling engine provides the following major functions.

? Building and *incrementally* updating customer calling behavior profiles by mining call records flowing into the data-warehouse daily, using an OLAP server.

? Maintaining profiles by *staging data* between the data-warehouse and the OLAP multidimensional database.

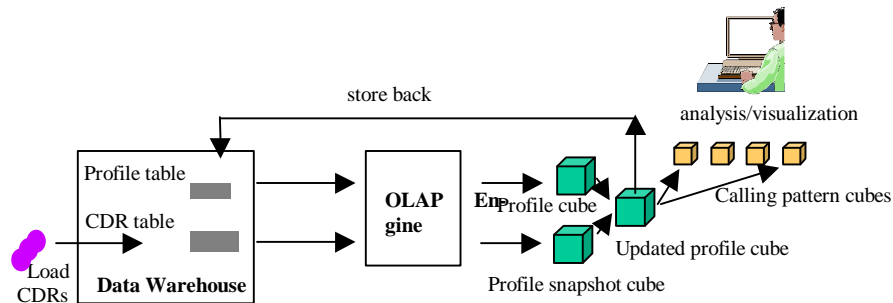
? Deriving multilevel and multidimensional customer *calling patterns* from profiles for analysis.

? Comparing the similarity of customer calling patterns from volume and probability distribution points of view, and generating multilevel and multidimensional similarity measures, to be used in such applications as fraud detection.

The profiling engine is built on top of an Oracle-8 based data-warehouse and Oracle Express, an OLAP server (Figure 1). Call data records, customer behavior profiles and other reference data are stored in the warehouse. Call data records are fed in daily and dumped to archive after use [3]. The OLAP server is used as a computation

engine for creating and updating profiles, deriving calling patterns from profiles, as well as analyzing and comparing calling patterns. The following process is repeated periodically (e.g. daily).

- ? Call data records are loaded into call data tables in the data-warehouse, and then loaded to the OLAP server to generate a *profile-snapshot cube* that is multi-customer oriented.
- ? In parallel with the above step, a *profile cube* covering the same set of customers is retrieved from the data-warehouse.
- ? The profile cube is updated by merging it with the profile-snapshot cube.
- ? The updated profile cube is stored back to profile tables in the data-warehouse. The frequency of data exchange between the data-warehouse and the OLAP server is controlled by certain data staging policies.



**Fig. 1.** Data warehouse and OLAP server based profiling engine architecture.

In order to reduce data redundancy and query cost, we chose to maintain minimal data in the profile tables in the data-warehouse. We include multiple customers' calling information in a single profile table or profile cube, without separating information on outgoing calls and incoming calls. We make the relational schema of the profile table directly correspond to the base level of the profile cube. Derivable values at higher levels are not maintained in the data-warehouse.

The OLAP engine actually serves as a *scalable* computation engine for generating *profile cubes*, deriving *calling pattern cubes*, analyzing individual calling patterns in multiple dimensions and at multiple levels, and comparing pattern similarity. From a performance point of view, it supports indexed caching, reduces database access dramatically and extends main memory based reasoning. From a functionality point of view, it allows us to deliver powerful solutions for profiling, pattern generation, analysis and comparison, in a simple and flexible way.

#### 4 Profile Cubes and Calling Pattern Cubes

We deal with two general kinds of cubes: multi-customer based profile cubes and single customer based calling pattern cubes.

## 4.1 Profile Cubes

A profile cube, say PC, and a profile-snapshot cube, say PCS, have the same underlying dimensions, and contain profiling information of multiple customers in direct correspondence with the relational tables in the data-warehouse. In the Oracle Express language they are defined as

```
define PC variable int <sparse <duration time dow callee caller>> inplace
define PCS variable int <sparse <duration time dow callee caller>> inplace
```

where callee and caller are called and calling numbers; dimension *time* has values ‘morning’, ‘evening’, etc, for time-bins; dimension *duration* has values representing duration-bins (e.g. ‘short’); and dimension *dow* has values representing days of week (e.g. ‘MON’). The use of keyword “sparse” in the above definitions instructs Oracle Express to create a composite dimension *<duration time dow callee caller>*, in order to handle sparseness, particularly between calling and called numbers, in an efficient way.

Profile-snapshot cube PCS is populated by means of *binning*. A call data record contains fields with values mapping to each dimension of the PCS cube. Such mapping is referred to as binning. For example, ‘8am’ is mapped to time-bin ‘morning’, 5 minutes is mapped to duration-bin ‘short’. A call made at 8am and lasting 5 minutes falls into the cell corresponding to *time* = ‘morning’ and *duration* = ‘short’.

Profile cube PC is retrieved from the database and updated by merging PCS, and then stored back to database. In Oracle Express, the merge of PC and PCS is simply expressed as

$$PC = PC + PCS$$

In this way customer profiles are updated incrementally as each new batch of call data records flow into the data-warehouse.

## 4.2 Hierarchical Dimensions for Multilevel Pattern Representation

Calling pattern cubes are derived from profile cubes and used to represent *the calling behavior* of *individual customers*. In order to represent such calling behavior at multiple levels, Dimensions *dow*, *time* and *duration* are defined as hierarchical dimensions, along which the calling pattern cubes can be rolled 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* such as a volume-based calling pattern cube. *D*, together with *DL*, *DL\_D* and *D\_D*, fully specify a dimension hierarchy. They provide sufficient information to rollup cube *C* along dimension *D*, that is, to calculate the total of cube data at the upper levels using the corresponding lower-level data. A cube may be rolled up along multiple underlying dimensions. For example, the *dow* hierarchy is made of the following objects.

- ? *dow*(day of week): dimension with values 'MON', ... 'SUN' at the lowest level (dd level), 'wkday', 'wkend' at a higher level (ww level), and 'week' at the top level ('week' level).
- ? *dowLevel*: dimension with values 'dd', 'ww', 'week'
- ? *dow\_dow*: relation (dow, dow) for mapping each value to its parent value, e.g.
  - dow\_dow*(dow 'MON') = 'wkday'
  - ...
  - dow\_dow*(dow 'SAT') = 'wkend'
  - dow\_dow*(dow 'wkday') = 'week'
  - dow\_dow*(dow 'wkend') = 'week'
  - dow\_dow*(dow 'week') = NA
- ? *dowLevel\_dow*: relation (dow, dowLevel) for mapping each value to its level, e.g.
  - dowLevel\_dow*(dow 'MON') = 'dd'
  - ...
  - dowLevel\_dow*(dow 'wkday') = 'ww'
  - dowLevel\_dow*(dow 'wkend') = 'ww'
  - dowLevel\_dow*(dow 'week') = 'week'

Analogously, 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 *duration hierarchy* is made up of dimension *duration*; dimension *durLevel* with values 'dur\_bin' and 'dur\_all'; parent relation *dur\_dur* and level relation *durLevel\_dur*.

For profile storage, combination and updating, only the bottom levels are involved, therefore rolling up profile cubes such as PC is unnecessary. Rolling up is only applicable to calling pattern cubes for analysis purposes.

### 4.3 Calling Pattern Cubes

A calling pattern cube is associated with a *single customer*. As the calling behavior of a customer may be viewed from different aspects, different kinds of calling pattern cubes may be defined. These cubes are commonly dimensioned by *time*, *duration* and *dow* (day of week), and in addition, for those related to outgoing calls, dimensioned by *callee*, and for those related to incoming calls, dimensioned by *caller*. Their cell values represent the number of calls, the probability distributions, etc. Calling pattern cubes are derived from profile cubes, say, PC, and then may be rolled up.

**Volume based calling patterns.** Cube CB.o represents the outgoing calling behavior of a customer. In Oracle Express that is defined by

```
define CB.o variable int <sparse <duration time dow callee>> inplace
```

Similarly, cube CB.d representing incoming calling behavior is defined by

```
define CB.d variable int <sparse <duration time dow caller>> inplace
```

The cell values of these cubes are the number of calls falling into the given 'slot' of time, duration, day of week, etc. When generated, CB.o and CB.d are rolled up along dimensions *duration*, *time* and *dow*. Therefore,

*CB.o(duration 'short', time 'morning', dow 'MON')*

measures the number of short-duration calls this customer made to each callee (dimensioned by callee) on Monday mornings during the profiling interval. Similarly,

*CB.o(duration 'all', time 'allday', dow 'week')*

measures the number of calls this customer made to each callee (total calls dimensioned by callee) during the profiling interval.

Cubes representing probability distribution based calling patterns are derived from volume-based pattern cubes. Depending on the application requirements various cubes may be derived. We list below two kinds of calling pattern cubes for outgoing calls. Calling pattern cubes for incoming calls can be defined similarly.

**Probability distribution on all calls.** Cube P\_CB.o for a customer represents the dimensioned probability distribution of outgoing calls over *all* the outgoing calls made by this customer, and is derived from CB.o in the following way

*define P\_CB.o formula decimal <duration time dow callee>*

*EQ (CB.o/total(CB.o(duration 'all', 'allday', dow 'week')))*

where *total(CB.o(duration 'all', 'allday', dow 'week'))* is the total number of calls this customer made to all callees (remember that CB.o has already been rolled up, hence we can use its top-level value). The value of a cell is the above probability corresponding to the underlying dimension values.

**Probability distribution on calls to each callee.** Cube P1\_CB.o is dimensioned by duration, ... and callee, and represents the probability distribution of a customer's outgoing calls over his total calls to the corresponding callee, and is also derived from CB.o as specified in the following

*define P1\_CB.o formula decimal <duration time dow callee>*

*EQ (CB.o/total(CB.o(duration 'all', 'allday', dow 'week'), callee))*

where *total(CB.o(duration 'all', 'allday', dow 'week'), callee)* is the total number of calls this customer made to each callee (dimensioned by callee). The value of a cell is the above probability corresponding to the underlying dimension values.

#### 4.4 Calling Pattern Similarity Comparison

Calling pattern comparison is important for such applications as fraud detection. Since the similarity of customer behavior can be represented from different angles, we compare calling patterns derived from customer calling behavior profiles, rather than comparing profiles directly. For example, some calling patterns might be similar in the volume of calls to the same set of callees, others might be similar in the time of these calls such as late nights. Our objective, therefore, is to enable the comparison of calling patterns along multiple dimensions and at multiple levels of the dimension hierarchies.

Given two input calling pattern cubes, say  $C_1$  and  $C_2$ , the output of the comparison is a *similarity cube*, say  $C_s$ , rather than a single value. The similarity cube  $C_s$  can be dimensioned differently from cubes  $C_1$  and  $C_2$  being compared. Each cell of  $C_s$

represents the similarity of a pair of corresponding sub-cubes (slices or dices) of  $C_1$  and  $C_2$ .

To support such cube similarity comparison, the following should be provided.

- ? The mapping from a cell of  $C_s$  to a pair of corresponding sub-cubes of  $C_1$  and  $C_2$ .
- ? The algebraic structure for summarizing cell-wise comparison results of a pair of sub-cubes to a single similarity measure to be stored in one cell of  $C_s$ .

For the latter, we have introduced the following two approaches. One treats a sub-cube as a bag, and summarizes cell-wise comparison results based on *bag overlap*. The other treats a sub-cube as a vector, and summarizes cell-wise comparison results based on *vector distance*.

Bag-overlap based approach is primarily used for comparing volume-based cubes, while vector-distance based approach can be used for comparing both volume-based and probability-based cubes. The similarity of volume-based calling patterns is meaningful only when they cover the same time-span. This limitation can be eliminated in measuring the similarity of probability-based calling patterns. This is especially useful in comparing a preset calling pattern with an ongoing one in real-time. For example, the following cube measures the similarity of probability-based outgoing calling patterns

*define P1SIM.o variable decimal <durLevel, timeLevel, dowLevel> inplace*

An instance of P1SIM.o is illustrated in Figure 2.

```

DOWLEVEL: week
-----P1SIM.O-----
-----DURLEVEL-----
TIMELEVEL   dur_all   dur_bin
-----
time_all    1.00      0.71
time_bin    0.94      0.70

DOWLEVEL: ww
-----P1SIM.O-----
-----DURLEVEL-----
TIMELEVEL   dur_all   dur_bin
-----
time_all    0.95      0.72
time_bin    0.92      0.71

DOWLEVEL: dd
-----P1SIM.O-----
-----DURLEVEL-----
TIMELEVEL   dur_all   dur_bin
-----
time_all    0.77      0.63
time_bin    0.73      0.61

```

**Fig. 2.** P1SIM.o: Multilevel and multidimensional similarity cube

P1SIM.o is calculated by comparing two probability-based calling pattern cubes based on *vector-distance* using a *cell-to-subcube mapping*. It takes two calling pattern cubes, P1\_CB.o and P1\_CB2.o (defined in the same way as P1\_CB.o) as input

(since P1\_CB.o and P1\_CB2.o are “views” of CB.o and CB2.o, the latter can also be considered input cubes).

Let us look at the following two cells that are based on the same dimension values as the cells shown in the above P1SIM.o examples.

? Cell *P1SIM.o(durLevel 'dur\_bin', timeLevel 'time\_bin', dowLevel 'dd')* says that there is 61% similarity between a corresponding pair of probability-based sub-cubes of P1\_CB.o and P1\_CB2.o. These sub-cubes are based on low-level values of dimension *duration*, *time*, *dow* and all values of dimension *callee*. The value of the above cell is the vector-based summarization of cell-wise comparison of the above pair of sub-cubes.

? Cell *P1SIM.o(durLevel 'dur\_all', timeLevel 'time\_all', dowLevel 'week')* says that there is 100% similarity of a pair of sub-cubes of P1\_CB.o and P1\_CB2.o that are based on high-level values of dimension *duration*, *time* and *dow*, and all values of dimension *callee*.

For details of multidimensional calling pattern similarity comparison, see [6].

## 5 Conclusions

The problem of customer behavior profiling occurs in many applications such as telecommunications and electronic commerce. In this paper we have developed a data-warehouse and OLAP based framework for customer behavior profiling, and illustrated its use in a telecommunication application. A prototype has been implemented at HP Labs. Our work demonstrates the practical value of using OLAP server as a scalable computation engine for creating and updating profiles, deriving calling patterns from profiles, as well as analyzing and comparing calling patterns. We plan to introduce parallel data warehousing and OLAP architecture to further scale the profiling engine.

## References

1. S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, S. Sarawagi: On the Computation of Multidimensional Aggregates, Proc. VLDB'96 (1996)
2. Q. Chen, U. Dayal, Meichun Hsu: A Distributed OLAP Infrastructure for E-Commerce, International Conference on Cooperative Information Systems (CoopIs99), UK (1999)
3. H. Garcia-Molina, W. Labio, J. Yang: Expiring Data in a Warehouse, Proc. VLDB'98 (1998)
4. S. Geffner, A. El Abbadi, D. Agrawal, T. Smith: Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes, Proc. ICDE'99 (1999)
5. J. Han: OLAP Mining: An Integration of OLAP with Data Mining, Proc. IFIP Conference on Data Semantics (DS-7), Switzerland (1997)
6. Q. Chen, U. Dayal, Meichun Hsu: Multidimensional and Multilevel Calling Pattern Similarity Comparison, Tech Rep. HP Labs, 1999.