A Data-Warehouse / OLAP Framework for Scalable Telecommunication Tandem Traffic Analysis

Qiming Chen, Meichun Hsu, Umesh Dayal

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

Abstract

In a telecommunication network, hundreds of millions of call detail records (CDRs) are generated daily. Applications such as tandem traffic analysis require the collection and mining of CDRs on a continuous basis. The data volumes and data flow rates pose serious scalability and performance challenges. This has motivated us to develop a scalable data-warehouse/OLAP framework, and based on this framework, tackle the issue of scaling the whole operation chain, including data cleansing, loading, maintenance, access and analysis. We introduce the notion of dynamic data warehousing for managing information at different aggregation levels with different life spans. We use OLAP servers, together with the associated multidimensional databases, as a computation platform for data caching, reduction and aggregation, in addition to data analysis. The framework supports parallel computation for scaling up data mining, and supports incremental OLAP for providing continuous data mining. A tandem traffic analysis engine is implemented on the proposed framework. In addition to the parallel and incremental computation architecture, we provide a set of application-specific optimization mechanisms for scaling performance. These mechanisms fit well into the above framework. Our experience demonstrates the practical value of the above framework in supporting an important class of telecommunication business intelligence applications.

1. Introduction

Telecommunication business intelligence applications typically require the mining of large volumes of Call Detail Records (CDRs) to generate system and customer behavior patterns [4]. In this paper, we describe tandem traffic analysis, a typical example of such an application. This class of application poses several challenging requirements to data warehousing and on-line analytical processing (OLAP).

The principal challenge has to do with data volumes and data flow rates. Typically, hundreds of millions of CDRs are created everyday, and the architecture must support loading and processing rates that match the input rate. A further challenge is to provide continuous, rather than one-time, analysis and mining of the CDRs. Since CDRs are continuously collected, it is important to mine them in “real-time” to dynamically detect trends and changes in traffic patterns. The results of tandem traffic analysis (which will be described in detail later) are represented by summary information at multiple levels of granularity, e.g., hourly, daily, weekly, and monthly. These summaries must be stored and incrementally updated. This requires an architecture that supports multilevel, incremental data mining and efficient maintenance and combination of partial results. Further, since it is not feasible to store raw CDRs for long periods of time, the architecture must support the retirement of data from the warehouse, integrated into the process of multilevel data reduction and summarization. The performance of the entire operational process, including data preparation, cleansing, loading, reduction, summarization and analysis, and retirement, must scale to keep up with the input data rates.

The above challenges have motivated us to develop a scalable framework on top of an Oracle-8 based telecommunication data-warehouse and a commercially available multidimensional OLAP server, Oracle Express [2]. Based on this framework, we have implemented a tandem traffic analysis engine with enhanced scalability,
maintainability and performance. Our approach has the following innovative features.

- We integrated data warehousing and OLAP technologies to provide a scalable data management and data mining framework. We provided multilevel and multidimensional data analysis based on OLAP technology. However, our framework is characterized by using OLAP servers as scalable computation engines rather than using them purely as front-end analytical tools.

- We introduced the notion of dynamic data warehousing to handle data staging and retirement, and thus control data life spans at multiple aggregation levels.

- We developed a parallel and incremental architecture to scale up OLAP.

- We introduced optimizations such as direct binning, and some application-specific optimizations such as avoiding two-leg correlations (these will be explained later), to reduce the computation load.

- The application, including the optimizations, was implemented by OLAP programming, i.e., in the scripting language provided by the OLAP server.

The proposed data-warehouse/OLAP framework and tandem traffic analysis engine have been implemented at HP Labs. Our experience has demonstrated that it is possible to use a data warehousing and OLAP-based architecture for incremental, scalable, and real-time tandem traffic analysis. We also showed that this approach significantly outperforms a direct SQL implementation of the application.

We share the same view as [1,7,9-12] in taking advantage of OLAP technology for analyzing data maintained in data-warehouses. Particularly, we are in-line with the efforts described in [8] to use OLAP tools to support large-scale data mining. Our approach of integrating incremental OLAP with data life-span control is unique. Finally, we showed that in addition to a general architecture, it is important to look for application-specific optimization mechanisms that can provide significant improvements in scalability and performance.

Section 2 introduces the background of the tandem traffic analysis application. Section 3 discusses OLAP based tandem traffic analysis. Section 4 describes the scalable architecture. Section 5 introduces tandem-analysis optimizations and shows how the proposed mechanisms fit in the above framework. Section 6 shows some examples of tandem traffic analysis. Finally, Section 7 gives some conclusions and summarizes lessons learnt.

2. Tandem traffic analysis

In this section we shall briefly introduce the background of the tandem analysis application.

![Figure 1: Tandem Traffic](image)

In the telecommunication industry, voice trunks connect end offices (EO’s) that are connected to and controlled by the SS7 signaling network. By monitoring the SS7 network, call detail records (CDRs) are generated to represent information specific to each call attempt. Each CDR typically contains, among other things, the calling phone number, the called phone number, the start and end time of the call, as well as the point code of the originating end office, referred to as the originating point code (OPC), and that of the destination end office, referred to as the destination point code (DPC). Tandem traffic analysis studies the traffic volume between pairs of end offices.

To motivate the tandem analysis, we need to further introduce the categories of the end offices within the domain of a local carrier (Please refer to Figure 1.)

- There are 200 Local Access Transport Areas (LATAs) in US. (In Figure 1, the shaded oval represents a LATA). Inter-LATA services are provided by Inter-Exchange Carriers (IXCs). Intra-LATA services are provided by Local-Exchange Carriers (LECs). The physical access location interface between a LEC and an IXC network is called a Point Of Presence (POP), that is the point to which the telephone company terminates a subscriber's circuit for long distance services or leased line communications.

- There exist multiple LECs in a single LATA, where an Incumbent Local Exchange Carrier (ILEC) that is already in service as of the effective date of Telecommunications Act of 1996, may need to terminate calls originated from the service areas of Competitive Local Exchange Carriers (CLECs). So end offices are categorized into ILEC EO’s and CLEC EO’s.
In a LATA, signals may be transferred directly between EOs, or routed indirectly through a toll office called Tandem. There exist several reasons for using a tandem, notably, preplanned routing, switch volume overflow and alternative path provisioning for handling exception conditions. Since a tandem switch is very expensive, it is typically used for a call only when the direct route is not available or is overloaded.

We refer to ILEC’s regular end offices (i.e., not tandems) as internal point codes; the POPs and CLEC end offices external point codes; and the point code associated with the tandem switch as the tandem point code.

Traffic through EOs, including tandems, is measured by Centum Call Seconds (CCS), which is a traffic volume unit of measurement equivalent to 100 call-seconds. The basic task of tandem analysis is to produce tandem traffic summaries which document tandem-routed and non-tandem routed CCS, and in addition, number of call attempts, number of answered calls, etc, for each pair of point codes (internal or external), for each hour of the day. Once the tandem traffic summary is produced, various analysis measures can be derived. Below are two examples.

- **Overflow analysis:** Percentage of tandem-routed local-to-local traffic. The use of tandem switch to route a local-to-local call, for which a direct route is available, is considered an overflow. The overflow percentage for each pair of local point codes dimensioned by hour and day can be derived from the basic tandem traffic summary.

- **Balance analysis:** Percentage of inbound or outbound traffic over total traffic between each pair of local point codes dimensioned by hour and day. If this percentage is skewed significantly, there is a possibility that some form of data traffic (e.g., ISP traffic) is involved.

Tandem analysis is used to accomplish the following:

- monitoring network configuration,
- maximizing trunk-group usage and avoiding traffic-jam,
- discovering reasons for high tandem load (where switches are mis-translated resulting in tandem overflow before direct trunk-group are saturated), and
- improving QoS by better business and network planning (e.g. where to add direct trunk-group to accommodate changes of traffic flow).

The following aspects of the application introduce complications as well as opportunities for optimization, which will be described in a later section:

**Duplicate CDRs and multiple legs of the same call:**

The network can be monitored at many points. When the monitoring equipment is set to monitor both inbound and outbound traffic of an EO, with respect to a single call, duplicated CDRs are generated, one at the outbound point of the original EO, and the other at the inbound point of the destination EO, with slightly different timestamps. A CDR is also generated for each leg that the call travels in the network, and it records the point codes of the two end points of the leg as its OPC and DPC. So for a call via a Tandem T, two separate CDRs are generated, one representing the leg before T that takes T as its DPC, and the other representing the leg after T that takes T as its OPC. These two CDRs represent the two legs of the call, and each leg may be in turn duplicated when both inbound and outbound traffic are monitored at the EOs and at the tandem.

**Mapping between phone numbers and point codes:**

In a CDR, a phone number is recorded in three fields: NPA, NXX and LINE; e.g., a ten-digit US phone number, (650) 857 3060, has NPA=650, NXX=857 and LINE=3060. NPA-NXXs and point codes are different, but mappings between them are defined.

---

### 3. Data warehouse/OLAP based tandem analysis framework

There are two general tasks for tandem analysis. The first task is to compute the multi-dimensional summary information out of raw CDRs for measuring tandem traffic. The next task is to perform derivation and analysis based on such summary information. A tandem analysis framework also manages input data, i.e. CDRs, and the resulting tandem traffic summaries.

#### 3.1 Basic architecture

Our framework is based on data warehousing and OLAP technologies. We represent CCS and other summary information by cubes. A cube $C$ has a set of underlying dimensions $D_1,..., D_n$ 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 $D_1,..., D_n$. 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. For example, a CCS cube is dimensioned by original end-office, destination end-office, tandem, hour and day. Based on a CCS cube, various cubes can be derived using an OLAP tool to
represent CCS related information from different dimensions and at multiple levels. The multidimensional data cubes holding tandem traffic summaries are generated and analyzed using OLAP engines. The OLAP engine actually serves as a scalable computation engine for creating and updating CCS and other summary cubes, deriving patterns from these cubes, as well as analyzing and comparing those patterns. From the performance point of view, it supports indexed caching, reduces database access dramatically and extends main memory based computation. From the functionality point of view, it allows us to deliver solutions for telecommunication traffic analysis in a simple and flexible way.

Our infrastructure is built on top of an Oracle-8 based data-warehouse and Oracle Express, an OLAP server. CDRs and other reference data are stored in the warehouse. CDRs are fed to the data warehouse periodically or continuously, and dumped to archive after use, under certain data retirement control [4,6]. Various applications, including generation of tandem traffic summaries, are performed on the CDRs that are on-line. The basic functions of our data-warehouse/OLAP based tandem analysis engine are:

- Building the CCS cubes and other summary data cubes by processing CDRs in the data-warehouse using OLAP servers.
- Deriving multilevel and multidimensional patterns from CCS and other resulting cubes for analysis.
- Staging the CCS and other summary data between the data warehouse and the OLAP Multi-dimensional Database (MDB). The CCS cubes are stored back to the data warehouse, and may be reloaded later for analysis or refresh (see Section 4 for incremental tandem analysis).

3.2 Defining the CCS and other summary data cubes

For the tandem analysis, we generate the following three kinds of measures (cubes) from CDRs.
- CCS (Centum Call Seconds, a traffic volume unit of measurement equivalent to 100 call-seconds);
- NC (number of call attempts);
- NCA (number of answered calls).

All these measures are dimensioned by the follows:
- epc.o (from point-code, i.e., point code of the calling number);
- epc.d (to-point-code, i.e., point code of the called number);
- tpc (the tandem point code if the call is tandem-routed);
- day;
- hour.

In Oracle Express Language these cubes are defined as follows.

```sql
define CCS variable int <sparse epc.o epc.d tpc day hour> inplace

define NC variable int <sparse epc.o epc.d tpc day hour> inplace

define NCA variable int <sparse epc.o epc.d tpc day hour> inplace
```

Note that the use of the keyword “sparse” in the above definitions instructs Oracle Express to create a composite dimension `<epc.o epc.d tpc day hour>`, in order to handle sparseness in an efficient way. A composite dimension is a list of dimension-value combinations. A combination is an index into one or more sparse data cubes. The use of a composite dimension allows Oracle Express to store sparse data in a compact form similar to relation tuples without restriction on rolling up or drilling down the measures along any individual dimension.

A CCS cube is populated by means of binning. A CDR contains fields whose values have to be mapped to values of dimensions of the cube. This mapping is referred to as binning. For example, ‘01Feb98 16:44am’ is mapped to hour-bin ‘16’ and day-bin ‘01Feb98’. A call made at ‘01Feb98 16:44am’ from end-office ‘249-133-1’ to end-office ‘249-133-6’ via tandem ‘249-133-100’ falls into the cell corresponding to these dimension values.

Various measures can be defined as formulas over the basic measures above. The ability to use formulas to define measures over a multi-dimensional space is a powerful feature of OLAP tools. Below are some more examples.

- Number of failed calls
  ```sql
  define NCF formula (NC - NCA) int <epc.o epc.d tpc day hour>
  ```
- CCS of all calls (tandem or non-tandem routed), by end points and by day and by hour
  ```sql
  define CCS.ee formula (total(CCS, epc.o epc.d day hour))
  ```
- CCS of direct calls between end-points (not via tandem)
  ```sql
  define CCS.dir formula (total(qual(CCS, tpc '_'), epc.o epc.d day hour))
  ```
- Total CCS of tandem-routed calls, by end points and by day and by hour
  ```sql
  define CCS.toll formula (CCS.ee - CCS.dir) int <epc.o epc.d day hour>
  ```
• Percentage of tandem CCS over total CCS, by end points and by day and by hour

\[ \text{define POU.toll formula } \frac{\text{CCS.toll}}{\text{CCS.ee}} \text{ dec}<\text{epc.o epc.d day hour}> \]

3.3 Multilevel tandem analysis

Cubes holding tandem traffic summary information can be rolled up along one or more hierarchical dimensions to support multilevel aggregation. Values of a dimension may form a hierarchy. A cube may roll up along a hierarchical dimension, say \( D \), such that a cell corresponding a high-level value of \( D \) holds the sum of the values of cells corresponding to lower level values of \( D \). As an example, the point code can form a hierarchy as shown in Figure 2.

With the availability of reference data describing this dimension hierarchy, the OLAP engine automatically calculates aggregate traffic measures over these categories, e.g., total tandem-routed CCS from Sprint to CLEC1, or total local-to-AT&T calls as a percentage of total tandem-routed calls, in the study area. Supporting dimensional hierarchy is another powerful feature of using OLAP for telecommunication business intelligence.

More formally, 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 \( C \). \( D \), 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. In the applications discussed in this paper, the following hierarchies are introduced.

The point code hierarchy is made up of the following objects.

- **pointcode**: dimension with values at pc level (e.g., ‘249-133-1’), carrier level (e.g. ‘249’ for SBC, ‘244’ for MCI, ‘254’ for ATT), carrier_kind (e.g. ‘IXC’ for Inter-exchange Carrier, ‘CLEC’ Competitive Local Exchange Carrier), and a special value ‘top’ at top-level.
- **pcLevel**: dimension with values ‘pc’, ‘carrier’, ‘carrier_kind’, ‘top’.
- **pc_pc**: parent relation (pointcode, pointcode) mapping each value to its parent, e.g.

\[ \text{pc-pc(pointcode '249-133-1')} = '249' \]

\[ \text{pc-pc(pointcode 'top')} = \text{NA} \]

- **pcLevel_pointcode**: level relation (pointcode, pcLevel) mapping each value to its level, e.g.

\[ \text{pcLevel_pointcode(pointcode '249-133-1')} = 'pc' \]

\[ \text{pcLevel_pointcode(pointcode '249')} = 'carrier' \]

\[ \text{...} \]

**Figure 2: Point Code Hierarchy**

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 \( time\_level\_time \).

For storing, combining and updating cubes, only the bottom level of each dimension is necessary. For analysis, we allow them to rollup along any hierarchical dimensions. The dimension \( epc.o \) (and \( epc.d \)) is a pointcode dimension, therefore a CCS cube covering a local area where multiple carriers coexist, can be rolled up along dimension \( epc.o \) (and \( epc.d \)). A cube derived from CCS can be defined in Oracle Express by the following.

\[ \text{define CCS.day formula } <epc.o epc.d tpc time> \text{ int EQ unravel(total(CCS, epc.o epc.d tpc day))} \]

This cube, CCS_day, can be rolled up along time dimension as well. Thus for example,

\[ \text{CCS.day(epc.o '249-133-1', epc.d '249-133-2', tpc '249-133-100', time 'July98')} = 98900000 \]

says that in July, 1998, the CCS between end-offices ‘249-133-1’ and ‘249-133-2’ via tandem ‘249-133-100’ is 98900000.

4. Architecture-based scalability enhancements

Using OLAP engines, together with MDBs, as a computation platform is a starting point to scale up tandem analysis. To reduce the data transfer between RDB and MDB, and to take advantage of the
programming capability of OLAP scripts, we adopt a technique called *direct binning*. Further, to enhance scalability, we have introduced the notion of *dynamic data warehousing* to handle data staging and control data life span at multiple aggregation levels. To enhance the computation performance, we have developed a parallel and incremental OLAP architecture.

### 4.1 Direct binning vs. cache-in

Very often, a MDB based OLAP (MOLAP) application involves two general steps: loading data to MDB to create raw data cubes, and then using OLAP operations to generate target cubes for analysis. In this way, the targeting data cubes are computed from raw data cubes. We refer to this process as *cache-in*.

#### Figure 3: Direct Binning

For scalability and efficiency, our approach is characterized by building target cubes directly from the data retrieved from relational data warehouse. CDRs retrieved from the RDB are processed (possibly dropped, repaired, extended and split as explained later) using scripts implemented in the OLAP server’s scripting language (we refer to this as “OLAP programming”), and contribute to the appropriate cells (bins) of the target cubes. In the tandem traffic study, as the CCS cube is dimensioned by time, a CDR across time bins contributes to more than one CCS cell. We refer to this process as *direct binning*, and we implement it using Oracle Express Language(Figure 3).

Populating and updating CCS cubes directly without first loading raw CDRs into a MDB is a simple but significant solution to scalability improvement since it reduces I/O load, memory load and computation load.

### 4.2 Incremental and continuous tandem analysis

There are two ways to carrying out the tandem traffic study. One way is to do is a one-shot analysis results from a given batch of CDRs. The other way is to incrementally compute the analysis results from CDRs that are periodically or continuously fed from the data warehouse. We refer to the former as *static* and the latter as *dynamic*. To the best of our knowledge, all the existing tandem analysis efforts are static. On the contrary, our goal is to provide dynamic tandem analysis.

Dynamic tandem analysis has the following benefits.

- It permits periodic, continuous or even real-time system monitoring, with the CCS analysis reflecting the current status and trends.
- It enables multilevel tandem analysis, which requires summarizing multiple partial results. For example, CCS distribution over a month cannot be concluded from a CCS cube that contains only one-day traffic volume. In fact, multilevel mining is incremental by nature.
- It enhances scalability, since it does not require the mining of CDR sets of arbitrary size. Incremental and distributed mining have become a practical choice.

#### Figure 4: Periodically Building CCS Cube

Periodically updating a CCS cube (e.g. daily) typically includes the following steps(Figure 4).

- A fresh batch of CDRs are loaded into RDB tables in the data warehouse, and then loaded to the OLAP server to generate a *CCS-snapshot cube*.
- In parallel with the above step, a previously computed CCS cube with the same dimensions is retrieved from the data warehouse.
- The CCS cube is updated by merging it with the CCS-snapshot cube.
- The updated CCS cube is stored back to the data warehouse. The frequency of data exchange between the data warehouse and the OLAP server is controlled by certain data staging policies.

In Oracle Express, merging CCS cubes is very straightforward. Let $CCS$ be the previously calculated cube and $CCS_1$ the newly calculated one. Their merge is simply expressed as

$$CCS = CCS + CCS_1$$

In this way, the tandem analysis results are combined and updated incrementally.
Continuously updating a CCS cube is based on continuously feeding the CDRs from the data warehouse to the OLAP computation engine. If the speed at which data is fed can match the speed of CDR generation, real-time Tandem analysis becomes possible. To accomplish this, we use a separate OLAP engine as an observer to read the current state of the evolving CCS cube, without halting the CCS cube generation. CCS display, and pattern derivation and analysis, are handled by the observer OLAP engine, as shown in Figure 5.

**4.3 Dynamic data warehousing**

An important maintenance issue is controlling the life span of both CDR data and derived aggregates such as tandem analysis results. As the CDR data sets are huge, keeping them on-line is impractical. We have developed a dynamic data warehousing approach for handling data at different aggregation levels, by allowing data at lower aggregation levels to retire from the data warehouse (archive to tapes) at a pace faster than the data at higher aggregation levels.

Unlike most conventional databases, a CDR database may never hold all the data on-line. In our system, a relational table typically contains multiple million CDRs and resides in the database only for a couple of days. Deriving summary information, such as CCS, from raw CDRs, is important not only for knowledge discovery but also for data reduction. CDRs and certain summary information must be retired periodically from the data warehouse to free up space for new CDRs. We call such a warehousing technique dynamic data warehousing. The key features are:

- incremental data reduction using the OLAP engine to generate more summarized and meaningful information;
- handling FIFO data with different life-spans at different aggregation levels;
- control of data operations based on information state, stage and age.

**4.4 Parallel OLAP**

To scale the architecture to handle a large volume of CDRs with reasonable performance, it is necessary to use parallel processing. Our architecture supports both data partitioning and task partition, as illustrated in Figure 6.

**5. Application-specific optimization**

Our experience shows that scalability should be tackled by considering the whole operation chain, including. In addition to designing the architecture for scalability (as discussed in the previous section), it is important to exploit opportunities for enhancing performance. Based on this principle, we have investigated tandem-analysis specific optimization mechanisms, and successfully implemented these mechanisms using “OLAP programming”. This section illustrates the diverse optimization opportunities in the whole operation chain in a real data analysis application.

**5.1 Basic tasks for generating CCS cubes**

We will first list the required tasks for tandem analysis, and then show the proposed mechanisms for eliminating certain expensive steps.
Conceptually, computing tandem analysis measures requires the following steps.

- Identify the CDRs relating to the local LATA and filter the irrelevant ones. This is done by comparing the OPC and DPC fields of each CDR with the reference data to identify the corresponding ILECs and CLECs.
- Remove duplicate CDRs that have the same call origin and destination but are generated at different locations of the SS7 network, and thus have slightly different timestamps. The cost for computation might reach O(N^2), or O(NlogN) if indices are provided (however, the index space would be fairly large).
- Repair incomplete CDRs by filling in NPA-NXX based on mappings from local point codes (OPC, DPC) to NPA-NXX.
- Identify the legs of the same call via tandem, T, typically by correlating the in-bound leg (with the point-code of T as its DPC) and the out-bound leg (with the point-code of T as its OPC). In SQL, this could be a very expensive task that requires a semi-join operation over large relations or fragments.
- Break down each CDR into the multiple time bins it overlaps with, to compute the traffic summary between each pair of EOs in those time periods. Multiple CDRs may contribute to the same CCS time bin. This process could be very expensive if implemented as database operations.
- Generate reports and visualize multidimensional CCS and other measures. Once the cubes are produced, various analysis measures are derived easily with an OLAP tool.

### 5.2 Avoid two-leg correlation with CDR elimination and extension

A non-tandem routed call is recorded in a single CDR, while a tandem-routed call is recorded in two separate CDRs: the inbound one to tandem (with the tandem as its DPC) and the outbound one from tandem (with the tandem as its OPC). Although in both CDRs the calling number and called number remain, tandem analysis focuses on the traffic between end offices identified by point codes rather than phone numbers. These two CDRs represent two legs of the call but miss either the real OPC or the real DPC. Therefore, they normally need to be correlated to form a single logical call with the real OPC and DPC.

As CDR correlation is very expensive, we opt for an approach that avoids the need to perform correlation. This approach, referred to as **CDR elimination and extension**, is characterized by dropping one of the two legs, and extending the CDR of the remaining leg with a local point code that is mapped from the NPA-NXX of the calling (called) number. Such mapping can be accessed from Line Information Database (LIDB) or discovered by mining sufficiently large numbers of CDRs.

In short, our approach consists in

- replacing the DPC that is recorded in the inbound CDR as TPC, by the DPC corresponding to the called number, or
- replacing the OPC that is recorded in the outbound CDR as TPC, by the OPC corresponding to the calling number.

After such replacement, only one leg is needed. The proposed CDR elimination and extension approach identifies which leg to eliminate, and which leg to extend by point code replacement.

Given a LATA, tandem-routed calls fall into one of the following cases: local-to-local, external-to-local, local-to-external, and external-to-external. Legs fall into one of the following cases: tandem-to-local, local-to-tandem, tandem-to-external, and external-to-tandem. We ignore a rare case, tandem-to-tandem, and for external to external calls of a LATA (which form a very small fraction, less than 4%, of the calls in our test area), we consider only the total traffic.

For each case, the strategy for dropping and keeping the legs is listed below; the CDR of a kept leg will be extended.

<table>
<thead>
<tr>
<th>Tandem-routed call</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local-to-local</td>
<td>Extend the first leg (local to tandem) and drop the second leg (tandem to local).</td>
</tr>
<tr>
<td>External-to-local</td>
<td>Extend the first leg (external to tandem) and drop the second leg (tandem to local).</td>
</tr>
<tr>
<td>Local-to-external</td>
<td>Extend the second leg (tandem to external) and drop the first leg (local to tandem).</td>
</tr>
<tr>
<td>External-to-external</td>
<td>Use both legs and reported separately as external-to-tandem as well as tandem-to-external traffic.</td>
</tr>
</tbody>
</table>

In general, when the many-to-one mappings from local NPA-NXXs to local point codes are available, this approach is applicable. It can significantly cut down the volume of CDRs to be processed (e.g. all the CDRs from tandem to local point codes can be ignored), and reduce the computation load for CDR correlation.

### 5.3 Avoid duplicate removal
Depending on how the SS7 signaling network is monitored, each leg of a call may be recorded in two CDRs: at the originating point, and at the terminating point. These two CDRs are essentially identical, except for some very small difference in timing. Duplicate removal is also very expensive even using an efficient algorithm. Instead, we opt to include all duplicated CDRs in computing the summaries, and then divide the summary measure in each of the cells of the resulting cube by 2.

5.4 Limit CDR repair

Many CDRs contain incomplete information on calling or called numbers. Through data mining, we found that the omission is generally made by some default handling mechanisms. For example, if OPC is recorded in a CDR, and it is for a non-tandem local EO, then the corresponding NPA or NPA-NXX may be missing, provided that it can be recovered from the mappings between point codes phone numbers. When a particular NPA is used most in a LATA, it might be ignored for the phone numbers in that LATA.

In order to perform the point code mapping, it is necessary that some of the CDRs be repaired. However, it is not necessary to repair all the CDRs; only the CDRs that (a) contain a tandem point code, and (b) are not for the leg to be dropped, need to be repaired.

6. Analysis examples

The prototype was tested with multiple millions of real CDRs from a test area. The following table is the result of an Oracle Express query on the CCS.ee cube, the total CCS between pairs of point codes by hour by day. In this example, we show the result for calls from a local point code x-y-1 (we suppress the actual point codes for privacy reasons) to other local point codes (x-y-2 through x-y-6), for hour 3-10 on the day of April 2, 1998.

The following two tables are the results of querying on the ratio of traffic volumes from point code X to point code Y vs. that from Y to X. They are used for traffic balance analysis. If the ratio is close to 1, then the traffic of the two directions is relatively balanced. If the ratio is much larger than 1 (which implies that it would much smaller than 1 in the other direction), then the traffic is very unbalanced.

In the first table, we show the result of a query for these ratios based on total CCS (both tandem-routed and directly routed) between local point codes at hour 19. Based on the sample data shown in this table, we can see that the traffic among the first five local end-offices (identified by point codes) appears to be relatively balanced. The traffic between the first five and the sixth end-office is very unbalanced (most of traffic is directed towards the sixth end-office, vs. the other way round.)

In the second table, we show the result of a query for these ratios based on tandem-routed CCS between local point codes at hour 19. Based on the sample data shown in this table, there appears to be additional imbalance in the tandem-routed traffic to the second point code from most other point codes.

7. Conclusions

We recognize that data warehousing solutions must take into account the specific requirements of the applications. We have identified several important scalability issues and described an architecture that addresses these issues. We have introduced the notion of dynamic data warehousing, using OLAP servers as computation engines, and incremental and parallel OLAP processing. We have developed a prototype implementation of this architecture for SS7 traffic analysis, with enhanced scalability, maintainability and performance. In this paper, we reported our experience in building a tandem traffic analysis application based on the proposed architecture.
Unlike most OLAP-based applications that treat OLAP servers purely as front-end analysis tools, we use the OLAP server as a computation engine, and support information staging between the data warehouse and the OLAP MDB. The incremental and parallel OLAP architecture further supports scalability. This work also shows the experience of using “OLAP programming” to provide a unified platform for data access, processing and analysis.

We have also tackled the maintenance issue by introducing an approach to dynamic data warehousing, which provides information staging and retirement, allowing different time-spans for information at different abstraction levels. We plan to introduce a system component to automate such control.

The performance of the tandem analysis application is significantly enhanced by the above architecture as well as by our specific optimization mechanisms such as call leg elimination and extension. We emphasize that to achieve the scalability and performance requirements, it is important to look for optimization opportunities through the entire operation chain: data preparation, data cleansing, data loading, data maintenance, data access and data analysis

In our experiments, we achieved processing rates of 1 million CDRs (each 210 bytes in length) per hour to generate a set of CCS, NC and NCA cubes using a single UNIX machine. These cubes are dimensioned as described before, and each of them contains 181,056 cells. Using OLAP, a simple cube construction task at this machine and at this scale is about 45% faster than using an embedded-SQL based C program to compute the CCS, NC and NCA. (Using the parallel architecture can significantly improve performance, although we didn’t actually measure this in our experiments. The actual improvement depends on the data size for cube export/import between MDBs.) Furthermore, avoiding two-leg correlation reduces data load by at least 25%. In the presence of duplicate CDRs (this is often the case), avoiding duplicate CDR removal reduces data processing to half. Also, only 37.5% of the incomplete CDRs needed to be considered for repair, and in practice even fewer actually needed to be repaired.

Our experience demonstrates the flexibility, simplicity and power of the proposed architecture for supporting large-scale applications. We plan to build a complete benchmark to compare the performance of different OLAP tools and different OLAP architectures in the context of telecommunication business intelligence.

References


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

