Data Quality Issues in Service Provisioning & Billing

Tamraparni Dasu & Theodore Johnson
AT&T Labs – Research
180 Park Avenue, Florham Park, NJ 07932

Introduction

Large corporations maintain a complex array of databases and data warehouses to record data when a customer orders a particular product or services and has recurring interaction with the corporation. The data are used for many core functionalities of the company, such as billing a customer, maintaining the customer's history of interaction with the company (e.g. complaints), billing history and history of purchases and usage. The warehoused data are analyzed and mined for customer segmentation, customer retention and acquisition activities (CRM programs), target marketing, advertising, developing new product and service offerings and many other activities (e.g. revenue assurance) that can save and generate significant revenues for the company. Thus, considerable money is spent on collecting, storing and analyzing the data.

However, the data collected are often riddled with problems and glitches. Bad data can lead to bad decisions, loss of revenues, loss of credibility and damage to customer goodwill. More recently, there is a pressure on corporations to provide a customer with electronic access to the customer's account for self-monitoring. Ideally, a customer would want to add, delete or update their order in real time, online. This requires a company to have an accurate and timely view of every customer's services, products and configurations. It is the pressure to provide such e-services to e-enable the customers that has forced many companies to examine the quality of their customer data and push for automatic flow of data seamlessly across various databases and warehouses.

In this paper, we discuss data quality issues as they arise in the process of recording a customer's data, starting from the point of sale, through provisioning of the service, to the mailing of a bill to a customer for products and services. We cannot discuss specifics due to proprietary reasons, however we illustrate with a hypothetical example. There are many interconnected stages involved such as recording the customer order, provisioning it, tracking customer care issues, computing recurring and non-recurring charges, and finally billing the customer. Typically, each stage is maintained by a different organization, in a different database, with poor communication between different databases. All these factors contribute to data quality issues. Using an illustrative example, we will describe potential problems at each stage and propose possible solutions. Many factors, such as legacy systems and decisions driven by the need for quick "time to market" remedies, make it difficult to solve data quality issues that arise in the service provisioning and billing process, especially in a large corporate setting.

The data quality problems and processes discussed in this paper motivated the development of a database browsing and profiling tool that we discuss in a separate paper [5]. Unfortunately, the paper is too long to include as an appendix to this paper, so we have tried to quote briefly from it or refer to it wherever possible.

Note: We do not consider the process prior to the placing of an order (telemarketing, sales) or after the bill has been rendered (bill collection etc.).

Description of the Process & Motivation

Consider the case where a customer calls to order a service with certain physical requirements (a T1 line) and service features (data transfer speed). Let us assume that a single provider can meet all the requirements without involving a third party, so that we do not have to consider the data generated by the interaction between the parent company and third party vendors.

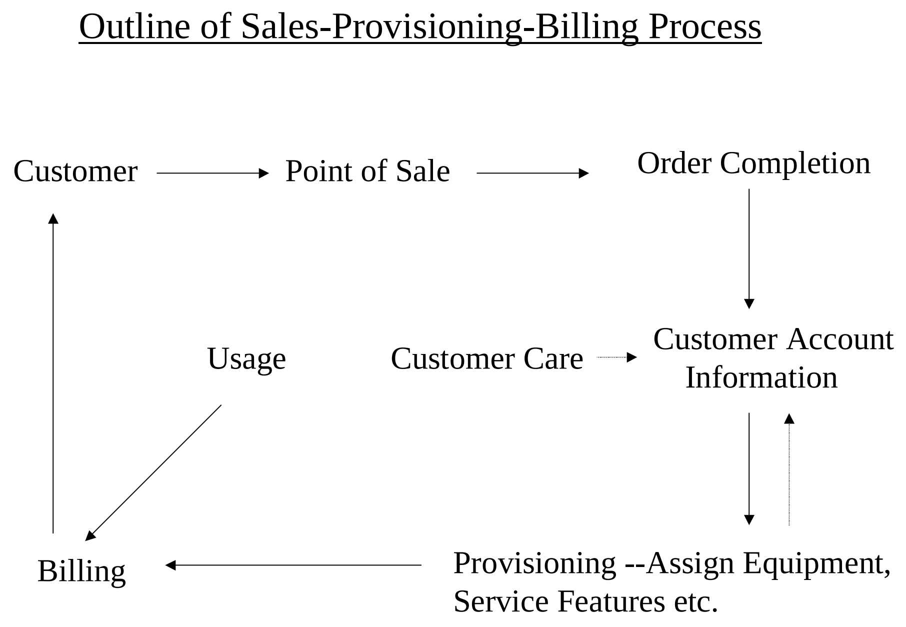
Outline of Sales-Provisioning-Billing Process
Customer
Point of Sale
Order Completion
Usage
Customer Care
Customer Account Information
Billing
Provisioning --Assign Equipment, Service Features etc.

Figure 1

Figure 1 represents the outline of a sales-provisioning-billing process. We show only the main steps, whereas in reality each stage involves a cluster of activities recorded in numerous databases. The process is set in motion when a customer orders a product or service. The sales representative who handles the call will typically record a minimal amount of information since his/her priority is to “sell, sell, sell”. Furthermore, there are **no standards or requirements** for data entry at this stage. Often, different GUIs are used in different sales branches, each with its own eccentric limitations, so that the amount, manner and type of data collected are dictated by such limitations. A follow-up call or mailing is made to the customer to gather more complete information and create a customer profile in the customer account database. Here again, there are several opportunities for incorrect information being entered – the customer might be unreachable, the person who placed the order might be different from the contact person who provides subsequent feedback and there might not be a strong communication between them, manual misrecording, etc. The customer’s order is then passed on for provisioning of physical, logical and service features (physical cable connection, IP address assignment, modem speed etc.). See Figure 1. Once the provisioning is completed, usage is monitored for recurring charges and the customer is billed regularly. Other databases such as customer care also begin to be populated.

All the databases are inter-dependent and need to communicate smoothly for efficient provisioning, maintenance, care and billing of a customer. As mentioned earlier, large customers would like to access their accounts to manage, update and request changes to their services online. In order to enable such services, it is important to maintain a complete, accurate and timely view of the customer's account.

Data Quality Demons

In the process that we described above, there are many opportunities for data problems to arise.

- Collecting customer data at point of sale. As mentioned earlier, there are opportunities for **manual errors** (180 Park Avenue typed as 108 Park Avenue), **limitations of the interface to the system** (will allow only 10 characters, will allow only 15 fields of information etc.) and **lack of incentive** (priority is to sell, not gather accurate information for subsequent analysis.) Optional data fields are seldom populated resulting in spotty availability of information that could be of value subsequently. It is difficult to address this problem since it is not desirable to tie down valuable selling time with data entry tasks.
- There are **no standards** that are followed across organizations. The Los Angeles branch might require the Industry Code to be populated for the sales person to close the sale, whereas the New York branch might not. Such a situation leads to large amounts of **missing data**.

<u>Different Views of a Customer</u>

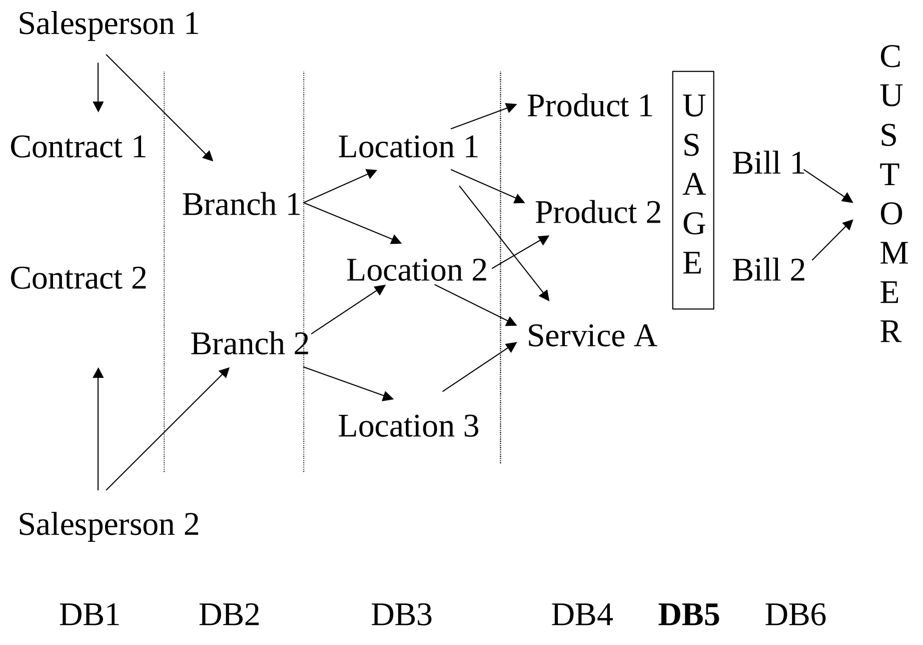

Figure 2

- Each organization has a different view of what a customer means. See Figure 2. For example, it could be a contract, a billing entity, a physical resource (a port) or the consumer of a particular product or service. The particular meaning drives the design and relationships within a database. It is **hard to correlate** these views since they often have hierarchies that result in many-to-many relationships that can be confusing. For example, a customer might have many branches scattered geographically, with different billing hierarchies, volume discount plans etc. In Figure 2, there can be arbitrary dependencies between the various views of a customer maintained in different databases. For instance:

  - It might be hard to establish that Bill 1 and Bill 2 belong to the same customer, making the **rendering of a single bill** (a very popular customer demand) difficult. Especially, if the two bills originate in parts of the company that have been recently acquired from outside.
  - Suppose the sales force is compensated on the basis of revenues generated from the products and services sold. In the above example, while the mapping between Sales Person (database DB1) and Product (DB4), and usage (DB5) and billing (DB6) might be accurate, the relationships between Product (DB4) and Usage (DB5) are difficult to maintain due to the complexity of the services provided. This makes it difficult to **determine the sales compensation**.

- Since each database is maintained by different organizations, with a high likelihood, there does not exist a common identifier for a customer across databases. Therefore, it is not easy to create a complete history of the customer from sales order to billing. Often, one has to rely on “soft keys” such as names and addresses to merge the databases to get complete information. Name and address matching can be expensive and sometimes inaccurate. For example, one database might list Ted Johnson while another might list Theodore Johnson or T. Johnson. Similarly, one database might list Park Avenue, while the other might list Park Ave. There are commercial vendors that sell software to address such issues.
- The feedback across organizations and databases is not strong so that the databases get out of sync rapidly. For example, the customer database might know that a customer has moved but might not notify the provisioning database so that they can de-allocate the resources assigned to the customer. As a consequence, valuable network resources might remain tied up, creating **spurious capacity saturation problem**s.
- When different companies merge or when one acquires another, they might have a large common customer base (e.g. when phone companies merge with cable companies) but nothing other than a name and address (in potentially different formats, conventions and database management systems) as a common identifier between the two companies’ databases. As mentioned earlier, merging databases based on name and address strings is difficult and error-prone.
- Furthermore, a company might inherit databases from an acquired company, but not with inadequate documentation and metadata. In such a case, it might be difficult to determine the key attribute (unique identifier) in any given table of the database. When there are hundreds of tables with thousands of variables, searching manually for a key-like attribute is impossible. (See our related paper [5] about designing an automatic database browser).
- Related to the previous point, it is difficult to find join paths between database tables to extract complete information about a customer. For example, finding the IP address assignment and the usage for the month of August for customer A might require joining several intermediate tables. While many database management software systems automatically generate schemas and diagrams to indicate the join paths, they are often based on field names and not necessarily on an interpretation of the content. Two tables might have completely different keys both of which are called “address”, which might then show up as a potential join path.

Potential Solutions

The problem of making information flow from sale to billing (and beyond) is a difficult one, especially in old, legacy systems. There are commercial ventures that enable migration into modern, easy to manage databases. However, the fundamental problem of insufficient documentation and lack of communication between related but independently run databases remains. Often, in a hurry to take a service to market, old technology is re-used in the context of new applications, creating predictable as well as unforeseen problems.

While we cannot address organizational and political issues in this paper, we will outline a few technical solutions that can be employed to facilitate a smoother flow of data.

- Create the infracstructure for knowledge and expertise sharing, as elucidated by Kuan-Tsae Huang, Yang W. Lee, Richard Y. Wang in their chapter on Network Knowledge Infrastructure. See [4] for details.
- As Nigel Turner mentioned in an IQ2000 paper [6], centralize and document metadata and domain expertise. While this is an organizational/process suggestion, the interfaces used by sales personnel and database managers can be setup to require the entry of certain information. This can be enforced through appropriate incentives (compensation tie-ins, other rewards, recognition) as well as easy and **efficient design of interfaces**. The source of many data quality problems can be traced back to a lack of metadata, documentation, interpretation and domain expertise.
- In the absence of good documentation, use heuristics to **automatically generate keys** and potential join keys. For example, **any field that has unique or almost unique values is a potential key**. Similarly, if a combination of two keys results in a unique or near-unique set of identifiers, then we have identified a two-way key. The key finding algorithm computes the counts of unique values of pairs, triples, etc. of the fields of the table. Therefore as a side effect the key finding algorithm finds *approximate dependencies*, in which one set of fields determines another. The key finding algorithms will not find all such dependencies (because it avoids doing all counts), and does not check the quality of the dependence. These dependencies are marked accordingly when they are entered into the profile repository. See our related paper [5] for details about data profiling. The paper is too lengthy to include in the appendix of this paper.
- Once we have rapidly identified keys within each table, we can start identifying "join paths", namely ways of joining different tables to extract comprehensive information about any given entity, e.g. a customer. Note that this task is difficult to perform manually, given that a database has hundreds of tables with dozens of attributes in each table. Without proper documentation, finding join keys can be very hard. Using "similarity measures", we can reduce the set of potential "join keys" to a manageable number. For example, using **string-matching** algorithms (see [3]) we can determine that a 9-digit-key in one table is similar to the two-way key ZIP+4 in another table. In practice, the algorithm has been highly successful in matching databases based on names and addresses. Similarity measures can be used to establish the confidence in the result of merging two tables.
- As a next step, we can **validate the joins** that we have performed using the keys that we have found. We can do the validation by using redundant information that resides in the

two tables. As a simplistic example consider: if we have joined the two tables using name+address, we can use other attributes for **asymmetric validation**. Therefore, if we see that there is revenue in one table, but the service indicator for that particular service is not populated, then there is a reason to believe that the two parts of the merged record do not belong to the same customer. If the information is consistent, then we need to do further checks to validate the match. The asymmetry is due to fact that, if the validation is unsuccessful we have good information that the record has not been properly merged, but if the validation is successful we are not certain of success of the merge. Let is consider a slightly more sophisticated example. Suppose one table contains usage and another table contains billed amount. We can verify an approximate join based on name+address, using **regression validation**. That is, we can use the matches that we are confident about (**labeled examples** to use machine learning terminology) to build a regression type model that estimates billed amount based on usage, and check whether the approximate joins are consistent with this model by comparing the actual billed amounts with the estimated billed amounts derived from the regression model (**supervised learning**). Another form of validation is based on **mutual information**. We use the labeled examples to determine attributes that have high mutual information. Loosely, mutual information is a measure of the extent of association between two attributes. If the mutual information is high, under certain circumstances, one attribute can be used to guess the value of the other attribute. We can then use the guessed value with the actual value to validate the join. See [1] for details on mutual information.

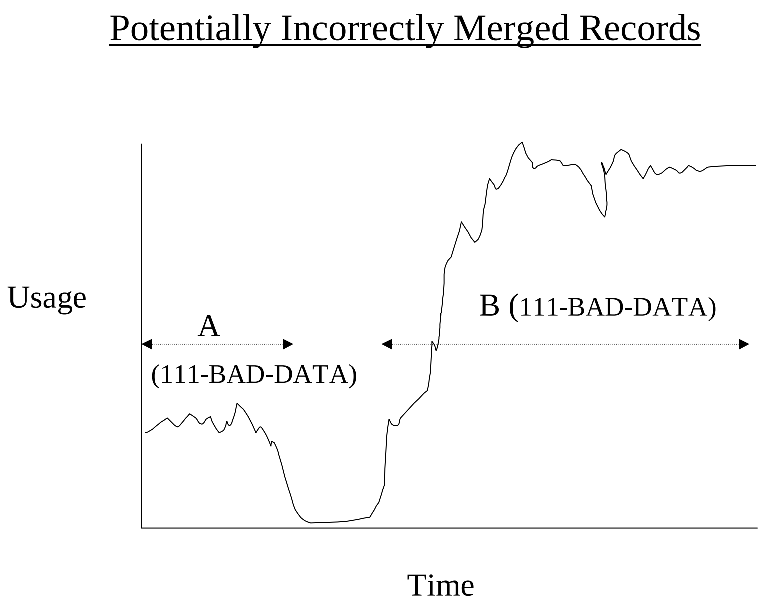

Figure 3

- We can extend the above concept to **validate time series** joins. For example, in the telecommunications industry, telephone numbers are re-assigned after alarmingly short waiting periods. So when usage tables are joined based on a telephone number (111-

BAD-DATA), it is quite possible to get spurious matches between customer A's history and a new customer B's history. See Figure 3. It is easy to screen out obvious mismatches and check them using more rigorous techniques. (See [2]). The rapid screening of time series to isolate a subset for further investigation is again an asymmetric validation technique. We can use linear models to investigate deviation from expected values.

- While it is not possible to mandate the use of a common identifier across organizations in a company, it might be possible to put in place a mechanism where the mapping between a common identifier and an individual organization is kept current. There is no burden on a particular organization to use a centrally dictated identifier that might not meet the specific needs of the organization. We have found this strategy to be successful in practice.

Conclusion

In this paper, we have focused on the data flow from sales order to billing of a customer. We have documented potential sources of problems and proposed technology based solutions. Other solutions based on political and organizational criteria are not considered in this paper. We refer to [5] for greater detail on the implementation of some of the technical solutions and the resulting tool.

Bibliography

[1] Thomas Cover & Joy Thomas : Elements of Information Theory, (1991), Wiley Series in Telecommunications. John Wiley & Sons.
[2] Tamraparni Dasu, Theodore Johnson & Eleftherios Koutsofios : "Hunting Glitches in Massive Time Series Data", IQ 2000, MIT, Boston, MA.
[3] Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan and Divesh Srivastava : "Approximate string joins in a database (almost) for free". In Proceedings of the International Conference on Very Large Databases (VLDB), 2001.
[4] Kuan-Tsae Huang, Yang W. Lee, Richard Y. Wang : Quality Information and Knowledge Management, (1998), Prentice Hall.
[5] Theodore Johnson & Tamraparni Dasu : "Database Browser" accepted to IQ 2001.
[6] Nigel Turner & John Hodges : "Deploying Information Quality Tools in a Federated Business" IQ 2000, MIT, Boston, MA.