2015

Using group strategy and pattern discovery for delta extraction in a limited collaborative environment

Zheng Lu
University of Wollongong

Using Group Strategy and Pattern Discovery for Delta Extraction in a Limited Collaborative Environment

A thesis submitted in fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Zheng Lu

School of Computing & Information Technology
August 2015

Dedicated to
My Father, Mother and My Son Yandong

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Zheng Lu
March 2, 2016

Abstract

It is very common that the large and multinational business organizations distribute their operational database systems over the wide area networks. The data integration from the distributed and highly autonomous operational database systems has to overcome many organizational and technical problems. Materialized view is the permanently stored relational table that contains integrated data from other data sources. Materialized view is an important technique applied to the implementation of data warehousing approach for data integration. In this approach, the data warehouse system extracts, transforms and loads data from heterogeneous sources into a single materialized view, so data residing in different sources are combined into a unified view.

The data sources change over times thereafter materialized view must reflect these changes and make the data up to date. These changes are referred to as *delta* and the process is called *view maintenance*. In a distributed environment, view maintenance by recomputing view definition can incur an enormous overhead associated with data transporting and loading. To address this issue, many incremental algorithms for view maintenance have been proposed to compute changes to a view only in response to changes to the data sources, in which it generally assumed that the delta somehow was made ready for use and therefore, the majority of research works have focused on how the data changes at the source systems can be integrated into materialized view. However, the problem of delta extraction from highly autonomous systems in a distributed environment is not a straight forward process nor an efficient one. The heterogeneity and distribution of the operational systems contribute to incompatibility and transmission problems. Additionally, a high level of autonomy of the local chapters in a large organization causes the problems of lack of organization-wide collaboration and support. Currently, only little recognition was given to the organizational and technical challenges associated with delta extraction in a distributed environment.

This work considers extracting delta in a distributed environment where the collaboration from highly autonomous operational database management systems is limited

to granting read only access on a set of selected relational tables. Because of inherently huge volume of data in data warehouse system, it is critical to minimize the volumes of data, i.e. communication costs, required to complete the task of delta extraction as much as possible. This research is based on the observations that usually, two consecutive snapshots are not very different and some of data in relational tables may not change frequently. It is proposed to maintain the statistics about remote data source in the data warehouse system, which would help to understand the essential characteristics of remote data. Based on the characteristics of remote data, we have developed the different group strategies to reduce the communication costs incurred during the process of delta extraction. In addition, to relax the underlying assumption of the group strategies, that is the changes to remote data are only caused by random events, we also defined a progression pattern to describe data changes with temporal regularities and developed a method for progression pattern discovery.

In this thesis, we have designed the 2-stage group hash method and the nested group hash method for the task of delta extraction. The experiments have been conducted to verify the accuracy and efficiency of the proposed approach. The experiment results have shown that by adopting the group strategies, the communication costs for the the task of delta extraction can be greatly reduced. In addition, the underlying cost estimation models of group strategies have been verified to be accurate and be consistent with the results collected from the experiment, which implies that the group strategy is capable of choosing an ideal group plan. On the basis of the results, we concluded that the group strategies are highly efficient solution to the problem of delta extraction. In addition, to cope with the computationally expensive task of progression pattern discovery, one of the inference principle we used to guide the discovery process is that the interesting update event observed from previous step is believed to be the consequent of the progression rule unless the contrary is proved in the succeeding process.

As the future work, we also consider to integrate sampling techniques into the group strategies. With this enhancement, firstly, the data warehouse does not require to maintain the statistics about the remote data source. More importantly, it provides the flexibility that the synchronization could run in arbitrary frequencies as desired. Moreover, considering various pricing models for the emerging trend of the cloud computing environment, as the future work, it would be helpful to design a mechanism that is used to balance the trade-off between data precessing that is measured by CPU

time and RAM usage and communication costs that is measured by the traffic allotment. According to the pricing model, a set of control parameters should be identified and defined, such that the process of delta extraction can be configurable in different settings according to the underlying pricing model. It also seems promising to extend our current optimization techniques to support NoSQL databases.

List of Publications

This is a list of referred papers that is related to this research work.

- Lu, Zheng., Yan, Jun. and Wang, Xinqin. (2015) 'Using Grouping Strategy and Pattern Discovery for Delta Extraction in a Limited Collaborative Environment', *International Journal of Business Intelligence and Data Mining*, In Press.

- Lu, Zheng., Liu, Haijun. and Yan, Jun. (2015) 'Delta Extraction in a Limited Collaborative Environment', *International Journal of Intelligent Information and Database Systems*, Vol. 9, No. 1, Pages 54 - 78.

- Lu, Zheng., Liu, Haijun. and Hyland, Peter. (2012) 'Delta Extraction Optimization for View Maintenance in a Limited Collaborative Environment', In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications*, Pittsburgh, Pennsylvania, United States, Pages 575 - 582.

- Lu, Zheng. and Getta, J. R. (2008). 'Identify and Extract Delta of Materialized View with Limited Collaborations', In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, PDPTA 2008, Pages 645-651.

The paper in preparation and to be submitted.

- Lu, Zheng., et al. (2015) 'Using Nested Group Algorithm for the Data Change Capture'.
  In preparation and to be submitted to *Data & Knowledge Engineering.*

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

When we submit online queries and search for the specified information such as airfares, local businesses and coupon offers, ideally, we would expect that a single interface that contains integrated data from all related data sources. In addition, more and more decision makers in business organizations interact with a data visualization tool that displays the current status of business metrics and key performance indicators (KPIs) for an organization. The displayed information by the data visualization tool provides insight into issues that contribute directly to the decision making process. In both scenarios, a data integration process must be designed, implemented and running behind the scene, to integrate, transport and populate the data from the data sources. Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [59]. The former scenario is to use the web portal interface [11], which is featured with the real time response to the queries. The latter is a front-end business intelligence application that provides the analytic functions and facilitates the identification of trends. Our focus in this research is to use data warehouse approach to integrate data for business intelligence environments. Typically, at the back-end, a very large and constantly growing historical data set are involved and the process of data integration is scheduled in a batch mode.

Data warehouse is a collection of integrated, nonvolatile, subject-oriented databases

designed to support the DSS function, where each unit of data is relevant to some moment in time, it contains atomic data and lightly summarized data [48]. Nowadays, it is very common that the large and multinational business organizations distribute their operational database systems over the wide area networks. Using the data warehouse approach to integrate the data from the distributed and highly autonomous operational database systems has to overcome many technical and organizational problems [14, 34, 36]. An important technique applied to the implementation of data warehouses is *materialized view*, which is defined as the permanently stored relational table that contains integrated data from other data sources [41]. The benefit of using materialized view is that the materialization of data can significantly accelerate the query processing when large volumes of data from multiple data sources are involved in the query processing. The data sources change over times, therefore the materialized view must reflect these changes and make the data up to date. These changes are referred to as *delta* and the process is called *materialized view maintenance*. In a distributed environment, the view maintenance by recomputing view definition incurs an enormous overhead associated with data transporting and loading. Alternatively, the incremental approach [18, 40] can be used, where the efficiency is achieved by only propagating the changed portions of the source data to the view without full recomputation. To use the incremental approach, we need to be able to extract delta from source systems. However, the choice of the right mechanism for delta extraction will depend on the availability constraints of the source systems.

Many proposed solutions to the problem of maintaining a materialized view in a distributed environment have some similarities [2, 21, 49, 98]. They all focused on the designing the efficient incremental algorithms and assumed that the data changes required by the algorithm somehow were made ready for use. The implication is that all related autonomous systems must be fully engaged in the process. Clearly, sometimes this assumption might not be realistic. The heterogeneity and distribution of

the operational systems contribute to incompatibility and transmission problems. Additionally, a high level of autonomy of the local chapters of a large organization causes the problems of lacking organization-wide collaboration and support. The problem of delta extraction from highly autonomous systems in a distributed environment is not a straight forward process nor an efficient one [56, 81, 93].

Furthermore, in light of these recent developments, big data [3, 20, 86] is to describe the the exponential growth volume of data, and cloud computing environments [8, 1, 29] are multi-domains environments in which each domain can use different security, privacy, and trust requirements. Despite the popularity Big Data and cloud computing, increasing numbers of traditional and non-traditional data sources are inundating organizations with data in volumes, access agreement and pricing model they may not have seen before.

The processes that facilitate the data integration, including the initial loading and the periodic refreshment of the data warehouse are commonly known as Extraction-Transformation-Loading (ETL) processes, which comprise the basis for data acquisition and organization of a data warehouse [50]. Quite often, the ETL process is tedious and very time consuming, which generally consumes 60% of the entire development time [50] and also has consistently made up 70% of the costs and risks of all data warehousing projects [32]. Particularly, when using data warehouse approach to integrate data, a well known issue in this process is the detection of changes in data source systems [53, 82, 93], where the problem arises when the source systems do not possess or do not provide the information about the changes that have occurred.

Currently, only little recognition was given to the challenges associated with delta extraction in a distributed environment. In this research, we aim to highlight an area important to the problem of data integration that has only received very limited research attention, namely, delta extraction with a limited degree of collaboration supported by a distributed environment.

To help understand the problem environment, we present a taxonomy of collaboration levels in a distributed environment. At the highest level, individual source systems take an active part in view maintenance by implementing the functions to log the data changes in source systems, to validate whether the changes are applicable to the current view, to compute the minimum change to the current view and to send the result to the data warehouse system. The source systems prepare all information required to refresh a materialized view and initiate communications with data warehouse system, but hide the low level implementation details. In this setting, the source systems and data warehouse form a global data warehouse system as a single seamless image.

The next level is characterized by the loosely coupled structure. Although the source systems are logically interrelated over a computer network and participate in a federation of a data warehouse system to make their local data sharable, they operate independently and are in fact separate systems. Individual source systems are engaged to view maintenance process by detecting the local data changes, notifying the modifications to the data warehouse system or exchanging information among other remote sites if required. The implementation techniques may include deployment of a database trigger or maintaining a transaction file, etc. The functions and resources owned by source systems are all made accessible to the data warehouse system. However, the data warehouse itself is responsible for coordinating the interactions among all related source systems, computing the final modification and applying the necessary changes to the current state of view.

The lowest level is characterized by the isolation, where individual source systems are stand alone DBMS. Individual source systems know neither the existence of others nor how to communicate with them. They only respond to the request from the data warehouse system, but not from other source systems. In addition, the source systems restrict their cooperation to granting read-only access rights to the selected relational tables and imposes limitations on the computational resources to the view maintenance

related applications. Furthermore, source systems do not provide the customized functions to support the view maintenance process, for example, the applications running on top of the source tables are altered to detect and notify of the data changes. The data warehouse system has to coordinate the view maintenance process among multiple source systems, to detect changes upon data sources and to make the necessary changes to the view.

When the collaborations are at the lowest level, the delta can be identified by comparing the current view with the data sources. Clearly, there are both processing costs and communication costs incurred this process. Processing costs usually are evaluated in terms of the number of disk accesses and CPU processing time, while communication costs are expressed in terms of total amount of transferred data. We are mostly concerned with the bottleneck of geographically distributed computer networks and propose a method to reduce the communication costs at the expense of some additional data processing. With the advent of more powerful processors and cheaper memories, we believe the issue of transportation of a large volume of data will be highlighted.

1.2 Contributions

In this research, we propose an approach for delta extraction if only a limited degree of collaboration is supported by a distributed environment. The limited collaborative environment refers to the environment where the collaborations are at the lowest level. In such an environment, many easy options to detect and extract data changes are unavailable. This work is trying to help to overcome the organizational and technical problems for the situation where the data integration needs to operate in a distributed environment and involves highly autonomous source systems. Because of inherently huge volume of data in a data warehouse system, it is critical to minimize communication costs as much as possible. Our original contributions here include:

1. Design a group hash algorithm for delta extraction. The algorithm is designed to identify the delta by comparing the current state of data in materialized view to the associated data sources. The data comparisons are performed on the hash value of a group of data elements, rather than on individuals. Consequently, this method significantly reduces the volume of data transfer between source systems and data warehouse system.

2. Develop the strategy that helps to create an optimal group plan. As the input, the group plan feeds into a group hash algorithm. The plan is created based on the statistics about remote data changes. The statistics are collected and maintained by a data warehouse system from the ongoing maintenance processes. We have considered two different scenarios about data changes in source systems, namely, uniformly distributed data changes and heterogeneous mixtures of data changes. Different strategies has been developed for each of these scenarios.

3. Propose a data change pattern, namely, progression pattern, which can be used to describe some underlying processes that have direct effect on the ongoing data evolution and to provide a useful prediction method for data evolution.

We believe the progression pattern is a necessary complement to the group hash algorithm that was designed to handle the situation where source data are only caused by random events and was not designed to recognize or to use any data pattern to deal with evolving data.

4. Develop a method for progression pattern discovery, which is defined to be a process of generating a plausible explanation for a given irregularity observed from ongoing data synchronization.

5. Design a nested group hash method, which is a multi-staged algorithm and is characterized by the condition that all groups included in the next stage must be chosen from a change group from the previous stage. The nested group hash method aims to seek the possibility of continual improvement. The improvement is in terms of efficiency and reliability, where the efficiency is measured by volume of data transfer required to complete the task, and reliability means how stable the underlined algorithm behaves in different settings.

We would like to point out that we are not going to claim that the proposed approach is the best solution to the problem of delta extraction. However, we believe that it is complementary to some other approaches, because it creates a possibility to maintain a materialized view when the higher level of collaborations are not easily available.

1.3 Organization of the Thesis

In Chapter 2, we discuss prior work in related fields, which covers the topics of data integration in general, the different classes of data change detection methods and the pattern discovery. We also draw attention to the strengths and limitations of some related methods through analysis.

In Chapter 3, we discuss our research design that was used to structure this research works. We describe our research process, selected research methods and discuss show how the major parts of the research process and selected research methods work together to address the central research questions in this study.

In Chapter 4, we present the preliminaries and describe the environment of view maintenance.

In Chapter 5, we present a two staged group method. Different strategies for creating an optimal group plan are developed for the scenarios of uniformly distributed data change and heterogeneous mixtures of data change respectively. We conduct the experiments to measure the efficiency and feasibility of our proposed methods. The experiment results are also compared with with other methods.

In Chapter 6, we begin with some discussions about the negative impacts that could be caused by an outlier of data irregularities. We proposed a data change pattern, namely, progression pattern, which is used to describe some underlying processes that have direct effect on the ongoing evolution of data and to provide a useful prediction method for data changes. In addition, we develop a method for progression pattern discovery, in which the pattern discovery is a process of generating a plausible explanation for a given irregularity observed from ongoing data synchronization.

In Chapter 7, we present a nested group hash method, in which a cost model is developed to estimate the volume of data required for the task of delta extraction. The experiments are conducted to verify the accuracy of the cost model, to evaluate the efficiency of the method and to compare with 2-staged group hash method in terms of

communication costs and reliability.

We conclude in Chapter 8 by summarizing the contributions of this thesis and discussing promising direction for future work in related fields.

Chapter 2

Background and Related Work

2.1 Data Integration

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [43, 59]. Currently, the volume, velocity, and distributed data sources in many organizations are growing at unprecedented levels, therefore the problem of designing the data integration process becomes more important in the information management disciplines. Data integration is characterized by a number of issues that are interesting from a theoretical point of view, and essentially, it needs to solve the problems of the data acquisition and schema mapping.

The data acquisition methods for data integration can be classified into two broad categories, namely, *lazy* and *eager* [94]. The lazy is an on-demand approach, which means the information is extracted from diverse sources only when the query is posed. In this approach, when the query is posed, according to on the source mappings [52, 97, 65], the query over the global schema is dynamically rewritten to determine the appropriated set of data sources and then fetch the data. The fetched data need to be translated, filtered or merged before returned as the results [13, 30]. In this approach, the programming components that re-formulate queries and combines results often is referred to as a mediator. Moreover, the global schema is a logical or virtual entity, so it is also called virtual data integration [71, 80].

In contrast to the lazy approach, in an eager approach, in advance of any query

execution, all data of interest are captured from the data sources and are materialized into a centralized data warehouse, therefore this approach is also called data warehousing approach. When a query is posed, the query is evaluated directly at the repository, without accessing the original information sources. The task of translating, filtering and merging on data sources when using eager approach is quite similar to that using lazy approach and it is commonly known as Extraction-Transformation-Loading (ETL) process.

A lazy approach is ideal when the required information changes rapidly or it is expected to be the most recent state of the information. However, if the vast amount of data involved in the query processing and transportation over the wide area networks, the inefficiency and delay might make this approach become not feasible. In the warehousing approach, because of the pre-materialization of data, it allows high query performance and more reliable query execution, but the query execution is not necessarily over the most recent state of the information. The data warehousing approach is ideal for the requirement of business intelligence environments that often contains a very large and constantly growing historical data set.

In the data warehousing approach, it requires techniques for keeping the integrated data in the warehouse up to date, i.e. refreshing the warehouse. This is hardly to be an efficient process, and heterogeneity and autonomy of the data sources introduce additional complications. Firstly of all, data integration is fundamentally about getting people to collaborate and share data [43]. Because of social and administrative reasons, quite often, the higher level of collaborations are not easily available [14, 34, 36]. Also as pointed out in [59] that one of the important related aspects of data integration problem that has not been addressed sufficiently is how to deal with possible limitations in accessing the sources. This focus in this thesis is related to the problem of data acquisition, particularly, when the support of collaboration among multiple source systems are at the limited level.

To design a system of data integration, it needs to clarify the required consistency

level of the integrated data supported in the context of the integration environment. The consistency level specifies how the state of the integrated data at any time corresponds to a collection of states of the data sources. It is important to the design of data integration systems in that, firstly, the users need to be aware of the supported consistency level, when using the integrated data to perform the transactional or analytical functions, and secondly, it is the system constraint that should be enforced at all times. According to [47], the three levels of consistency have been given as follows

- validity, this means that the state of the integrated data at time t should correspond to some state vector of the source databases.
- chronology, this means that the state of the integrated data at time t corresponds to the source databases at a times $\leq t$.
- order preserving, this means that successive states of the integrated data correspond to successive states of the source databases.

Although *order preserving* is a nice property, as it enforces the integrated data reflect source data exactly and completely, which may be not practical in most real world warehousing scenarios. The method of delta extraction in this thesis supports the consistency levels of validity and chronology.

Data integration is such a rich research field, a large body of research works range from simple, syntactic integration to complex, semantic integration [9, 17, 51]. Recently, the research interests has shifted to semi-automatically generating schema mappings. In general, automatic schema mapping is an AI-Complete problem and significant branch of the research works were proposed [24, 25, 45, 79] we will not discuss further in this thesis.

2.2 ACID

In the context of databases, four properties, atomicity, consistency, isolation, and durability (ACID), describe the major highlights of the transaction paradigm that emphasize aspects of reliability. The ACID has influenced many aspects of the development in traditional database systems and quite often, whether the transaction supported by a particular system satisfies the ACID properties becomes the tests of that system's quality. We therefore consider it is interesting to compare the our delta extraction method against these properties.

- Atomicity property states that database modifications must follow an all or nothing rule. Each transaction is said to be atomic, if one part of the transaction fails, the entire transaction is aborted, and the database state is left unchanged. Using the group strategies for the task of delta extraction, despite the initial comparison performed on the groups of elements, any groups that is detected to contain changes will also be subject to additional tests, therefore all individual changes are captured. The group strategies guarantee that all groups are processed thoroughly, only the detected changes are propagated into a data warehouse and no more no less.

- Consistency property ensures that any transaction will bring the database from one valid state to another. Like we discussed in subsection 2.1, the method of delta extraction in this thesis supports the consistency levels of validity and chronology.

- Isolation property is about the concurrency control which ensures that he concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially. Using the group strategies, the data extraction is always performed on the current state of data in materialized view, and the extracted delta is minimum to the current state, thus it guarantees serializability.

- Durability property ensures that once a transaction has been committed, the system must guarantee that completed transactions will persist. This property is also applied to the delta extraction method.

The group strategies are proposed as an optimized solution for the situation where delta extraction is only with a limited degree of collaboration supported by a distributed environment. Comparing against the properties of the ACID, the group strategies satisfy the properties of atomicity, isolation and durability. However, due to the nature of batch processing, the group strategies only support weaker level of consistency.

2.3 CAP Theorem

In the context of distributed computing, CAP theorem [15, 33] presents an idea that there is a fundamental trade-off between consistency, availability and partition tolerance. More specifically, it states that it is impossible for a distributed computer system to simultaneously provide all following three properties,

- Consistency (C). There must exist a total order on all operations or requests such that each operation or request looks as if it were completed at a single instant.
- Availability (A). Every request eventually receives a response about whether it succeeded or failed.
- Partition tolerance (P). The distributed computer system continues to operate despite arbitrary partitioning due to network failures. In other words, the network will be allowed to lose arbitrarily many messages sent from one node to another.

All three of these properties are desirable and expected from distributed systems. However, these properties form a special relationship, when it comes to designing and deploying applications in a distributed environment. According to CAP theorem, it is basically a 2 of 3 formulation, for example, it is impossible to reliably provide the availability and consistence of shared data when there are partitions in the network. Briefly speaking, there are three kinds of distributed system

- CP and forfeit A. Any request to the system has to wait for a response from the partitioned node which could result in a timeout error. Once the partition is healed and consistency can be verified, the the availability can be restored.
- CA and forfeit P. If the system is not designed to be fault-tolerant of a partitioning, the consistency and availability of the share data are able to be maintained by ignoring other partitions.

- AP and forfeit C. If the global consistency of shared data is forfeited, the system is able to return the most recent version of the data, which could be stale.

In this thesis, the group strategy is designed for the problem of delta extraction in a distributed environment. The data sources spread over a wide geographic area, therefore we cannot forfeit P and the choice is really between C and A. In addition, because the batch processing is used when refreshing the data, consequently it leads to a reduced level of data consistency. In summary, in a distributed environment, the design decision of group strategy is to prioritize availability with a reduced level of data consistency, i.e. following the pattern of AP and forfeit C.

2.4 Change Detection

In this thesis, the changes to data have been referred to as delta. In different contexts, the change detection is also called *data change capture* and *delta identification*. Change detection is important for various fields such as version management [42, 68, 74], data warehousing [56, 62, 81], active database [95] and cache replacement algorithms. In general, the problem of change detection can be categorized into three classes according to the objects being dealt with, i.e. flat file, hierarchical data and relational data.

2.4.1 File Synchronization

The rsync algorithm [90], a type of delta encoding, is used to minimize network usage when synchronizing the flat files on two connected computer systems. If two machines are connected by a low-bandwidth high-latency bi-directional communications link, the rsync algorithm is able to identify parts of the source file which are identical to some parts of the destination file, and only sends those parts which cannot be matched in this way. The optimization techniques used by rsync algorithm are the combination of rolling checksum and the standard MD4 hash algorithm, and the general assumption is that the source files and destination files have some similarities. In this method, the destination machine splits its copy of the file into fixed-sized blocks and computes two checksums for each block, namely, the MD4 hash, and a weaker but easier to compute rolling checksum. The hash values are sent to the source machine. The source machine must search source files for any blocks at any offset that possibly match the checksum of some block it received. Consequently, only the differential parts of file are sent back to the destination machine. The property of rolling checksum, originally invented by [23], makes it possible to calculate the succeeding checksum value very efficiently using the recurrence relations, such that the impact of the link latency can be minimized when considerable searches are required.

The optimization goal of rsync algorithm is very similar to the ours, in the sense

that to minimize network usage for the task of data synchronization. The optimization techniques in both works involve the use of the certain group strategies, where the rsync algorithm compares the file by using fix-size blocks, i.e. a group of characters in the file, and our delta extraction method performs the data comparison also on a group of elements. However, in the design of rsync algorithm [90], the question of *what is the optimal size of the block* was not being addressed, therefore it is not possible to evaluate and to determine what is the best option when choosing the block size. For example, the question remains to be answered is in order to minimize network usage, m-size block should be used or n-size block should be used. In this research, our method is based the probability theory, we not only develop a group hash algorithm for delta extraction, but also focus on the cost analysis aiming to find the optimal group size.

2.4.2 Hierarchically Structured Document

Hierarchically structured document allows users to describe data in hierarchically structured format, such as XML-like document and nest objects in databases. The changes to a hierarchically structured document are represented by the edit script, which includes a sequence of node insert, node delete, node update or the movement of a set of nodes, as the basic operations. The change detection for hierarchically structured documents is the problem of finding a minimum-cost edit script, i.e. the most compact deltas, that transforms one version of the file into another. When comparing two versions of hierarchically structured documents, typically, the documents are organized into the tree structures to facilitate the searching and comparison. The change detection problem is $\mathcal{NP}$-hard. Although the notion of a minimum cost edit is used, the reported results are often not proved to be minimal [19, 57, 92], but the optimal edit scripts. In addition, the cost model of the algorithms is also evaluated in terms of performance efficiency to cope with processing time required for a larger set of documents. Briefly, the change detection for hierarchically structured documents needs to design

the faster matching algorithm that generates the edit script guaranteed to be correct and close to the minimum. To improve the performance, the commonly used technique is to detect the copy and movement of a subtree, ie. a subset of nodes from an entire tree, whenever it is possible, instead of only searching the changes on individual nodes.

Between the problems of change detection for hierarchically structured document and delta extraction presented in this thesis, although the optimization goals are very different, the idea of optimization has some similarities. The optimization techniques in both works involves the use of the certain group strategies. In the problem of hierarchical change detection, the group strategy is to detect the copy and movement of a subtree, i.e. detecting operations on a group of nodes. In the problem of delta detection, the group strategy is to find an optimal group plan to efficiently identify the changes.

2.4.3 Relational Data

In this section, we discuss some data change detection methods for relational data that are currently available. We also draw attention to the strengths and limitations of each method through analysis.

Database Transaction Log. The previous works in [49, 81], proposed their data structures and algorithms for the distributed view maintenance, which is based on the assumption that the delta can be accumulated from transaction logs of remote sites. In DBMS environment, the transaction log is a proprietary format which logs a history of all changes made to the database. The transaction log based method can be very attractive, as it only poses very little impact on the source systems. However, this approach has its difficulties. Firstly, the format of log files are often proprietary to the DBMS and need to be parsed to obtain the relevant data modification. The challenge with here is the format may not be published by DBMS vendor. Also, typically the transaction file contains all changes made into entire database, due to the concerns of system security and data privacy, the administrators of remote sites may not be

willing to provide such accesses. Additionally, the format of transaction log file may change over time, which in turn requires the frequent changes of the view maintaining algorithm and the underlying data structure.

Database Trigger. Many view maintaining algorithms [70, 21] all have the assumption that every remote data source system is able and willing to send a message to notify of the data changes. After receiving the notification message, the data warehouse system issues a set of queries to all other remote source relations involved. The result of queries is used to compute the update to the current date of data. This approach requires *all* remote sites to implement database triggers upon the source relations to notify data changes. Using database trigger is an ideal way to capture data changes from remote source relations. It is an event based method, so theoretically every state change in source systems can be captured in real time. Not surprisingly, most of well-known DBMSs have included trigger as a standard function, such as Oracle, MySQL, Microsoft SQL Server, PostgreSQL and DB2. However, in a limited collaborative environment, because of extra costs on development and maintenance, the remote sites may not always be willing or able to make such commitments. Besides this, as pointed out by [81], if source systems are distributed and the business process does not execute as a global serializable execution, using trigger to capture a global transaction identifier along with the delta that guarantees uniqueness across the systems is a significant challenge. More importantly, to meet high demand of the performance and scale requirements of Big Data and cloud computing, the designs of many new emerging database systems such as MongoDB and Cassandra, adopt the strategy of prioritizing availability and horizontal scaling with a reduced level of data consistency. In this case, the standard trigger function is often not on the list of supporting functions, and clearly the data volumes these databases were designed to handle are very large. For more detailed survey, use case and performance comparison on these database systems, please refer to the works presented by [78, 39]. In addition, some legacy source systems may not offer the database trigger at all, such as Microsoft Access (JET), Microsoft

SQL Server Compact, LucidDB, Clustrix and RDM Embedded, etc. Also, the amount of data these databases can handle cannot be underestimated, and typically, the data capacities of a single table range from GBs to unlimited.

The environment of view maintenance with the assumptions listed above, namely, using database trigger and access to transaction log, is referred as a higher degree collaborative environment. Clearly, that is the desirable environment, but sometimes it is unachievable in reality.

Snapshot Differential Algorithm. The works in [56] formally defined the snapshot differential problem and proposed the algorithms by which snapshot differentials can be computed. A snapshot denotes a replica of a source relation. If materialized views contain integrated data and are created by distributed join over multiple remote data sources, the snapshot differential algorithm cannot be directly applied. In order to reduce the IO and communication costs, the snapshot differential algorithm used two optimization techniques to perform the data comparison, namely, *data compression* and *the window algorithm*. The data compression was proposed to be performed at individual tuple level, which certainly can reduce the volume of data transmission, but we believe the more effective way is to locate and compress tuples of a group based on the data change frequency, and consequently it can lead to further reduced data volumes and less data comparisons. This window algorithm assumes that the snapshots are not very different and matching records are physically nearby in the files. Clearly, the assumptions are often not true in a distributed environment. Source systems are autonomous, due to different data partition strategies, they can have very different file organizations. However, we believe that it is often the case there is a large amount of data that remains unchanged between two consecutive snapshots, i.e. two consecutive snapshots are not very different. To make this property very useful for data change capture, in stead of relying on the similarly physical file organization, we control the size of group to perform the comparison where the size is determined by the collected statistics on remote data from the ongoing maintenance process. Finally, we choose to

use the hash value to perform the data comparison rather than the compressed data, because the data compression generally is limited by a compression factor. In other words, the size of input message can greatly affect the size of output compressed message. On the other hand, most hash functions are designed to take a input message of any length and produce a fixed-length hash value. Clearly, hash value better suits the our approach, i.e. group based data comparison.

Complete Refresh. It is to ship all data of interest from all remote sites to a data warehouse system. The data from each remote site is referred to as a *data fragment*. After all data fragments are received, based on view definition, the data warehouse assembles these data fragments and reload them into materialized view. This approach has its difficulties. Firstly, because of the isolation of remote sites, there is no join operation allowed on multiple data fragments, thus the data shipped to a data warehouse may contain a large amount of redundancy. Secondly, because of the limitation of network bandwidth, transferring the large amount of data makes it either impractical or too expensive [37] .

2.5 Other Related Work

2.5.1 Pooled Sample Testing

In the fields of clinical and epidemiological studies, the pooled sample testing is a procedure for reducing the expected amount of testing needed to identify a rared virus in the population [26, 27, 91]. The testing procedure requires mixing part of pooled samples together and testing the mixture for existence of at least one infected element among them. If no such an element is indicated, i.e. negative pool, then no further testing is needed; otherwise, i.e. positive pool, further test is required to identify the individual infected elements. Similarly, the proposed methods of delta extraction is also to test a group of elements, rather than on individual one. However, their optimization goals are fundamentally different. In the pooled sample testing, the goal is to reduce the amount of testing. If translated into the problem of delta extraction, it is to reduce the number of queries on source systems to complete the task of delta extraction. Quite the contrary, the optimization goal for the delta extraction is to reduce the volume of data transfer *at the expense of additional data processing*. Secondly, in the pooled sample testing, an element is classified as either infected or good. For delta extraction, an data element is possibly being deleted, updated, inserted or unchanged, which means different testings have to be designed to each types of data changes. Particularly, in case of data change being detected, it may be necessary to decide which test to use next. Thirdly, in the pooled sample testing, generally it is assumed the probability of being infective is identical, for the problem of delta extraction, it also needs to consider the scenario of heterogeneous mixture of data changes.

2.5.2 Pattern Discovery

Th pattern discovery is to discover hidden patterns, unexpected trends, or to infer relationships of contextual and temporal arrangement in large databases. The techniques

of pattern discovery are the combinations of data mining, machine learning, statistics and data management techniques. In chapter 6, we will propose a data change pattern, namely, progression pattern and will also develop a method for the pattern discovery. The progression pattern needs to show some temporal regularities, therefore we mainly focus on the some background introduction to the temporal data mining.

Generally, the research works on the temporal data mining adopt Apriori-like approach [4] to generate a complete set of association rules, and then incorporate the temporal ordering information into the discovered association rules. Apriori algorithm originates from analyzing data from the transaction of supermarkets, in which each transaction contains a basket of purchased goods. The goal is to find a complete set of association rules, which indicate that the purchasing a set of items implies that another set of items is likely to be also purchased. For example, as one early study found, when diapers are purchased, beer is frequently purchased as well. The discovered rule is to describe the association of some items within the same transaction, but not among inter-transactions. Therefore, the ordering or timing information of transactions is not required in the mining process.

The first category of temporal data pattern is the the sequential pattern [5]. By extending the Apriori algorithm, the ordering label and the customer ID are incorporated into transactions, and consequently, the database is no longer just some unordered collection of transactions, but a sequence of transactions according to the identity and the ordering. For example, the transactions associated with a single customer can be viewed as a sequence of itemsets ordered by time. The sequential patterns of interest are shown as the ordered sequence of itemsets that frequently appear in the database. It can be used to predict after which events, an interesting event is most likely to occur. More advanced sequential pattern mining algorithms are reported in [12, 67], which are featured with the time window and the support of temporal constraints on multiple time granularities.

Another category of temporal data mining is to find the periodicity that is carrying

explicit time constraints. Periodicity detection is a classical problem in the field of signal processing for many years, where the well known technique is the Fourier transform. In the field of data mining, a typical example of the periodic event patterns is the cyclic association rule [75]. In which, an association rule is said to be cyclic if it occurs with a regular time interval within the entire length of the sequence, for example, at every first Monday of a month. In order to discover these rules, the transactions in the database need to be time-stamped in order to carry explicit time information. The discovery of cyclic association rule is firstly to perform an Apriori-like discovery, then to detect whether the cycles behind the discovered rules exist in the entire sequence of transaction.

The very tight temporal constraint of exact periodicity makes the cyclic association rule not very practical in the real world applications. To relax the constraint of exact periodicity, the partial periodic pattern has been proposed in [44, 69]. In [69], a statistical approach, namely, chi-squared test is used to detect and determine whether the candidate time period is a qualified frequent one. Also, the idea of partial periodic pattern is that the time tolerances are incorporated into the time constraints accounting for the imperfection. In [44], it studied methods for mining partial periodicity in a time series database, in which the partial periodicity indicates periodic behavior is associated with only a subset of all the time points. When designing partial periodicity mining algorithms, some useful properties related to partial periodicity are used to improve the performance. These properties include the Apriori property, the max sub-pattern hit set and shared mining of multiple periods.

The temporal pattern discovery methods above all ensure the completeness, in the sense that, to discover all the temporal patterns in the database that meet the predefined constraints. The central idea is to iteratively generate and verify the set of candidate patterns of length $k+1$ from frequent patterns of length k in the entire dataset. The progression pattern proposed in this thesis is used to facilitate the process of delta extraction, which is the part of process of refreshing warehouse. Typically, in

order to maintain the highest throughput of reporting and analytical functionality, the refreshing of data warehouse is limited to run at the the lowest possible priority level. Contradictorily, the completeness of pattern discovery only can be achieved through more processing. The approach that finds a complete set of the underlying data patterns in one go might be too time consuming, thus not a suitable in the context of refreshing warehouse. Alternatively, the discovery of progression pattern is designed to be a process of forming an explanatory hypothesis only in presence of irregularity being observed and requiring an explanation.

Chapter 3

Research Design

This work considers extracting delta in a distributed environment where the collaboration from highly autonomous operational database management systems is limited to granting read only access on a set of selected relational tables. Because of inherently huge volume of data in data warehouse system, this research aims to develop the strategies for delta extraction, in a way such that the communication costs can be minimized as much as possible. In this chapter, we will discuss our research design that has been used to structure this research works. More specifically, we will discuss our research process, selected research methods and will also show how the major parts of the research process and selected research methods including the definition of algorithms, implemented prototype, controlled experiments work together to address the central research questions in this study.

3.1 Research Process

The research process is a series of steps which lead to gradually revealing the complexity and progressively resolving uncertainty. It is an extremely cyclic process [58], which means later stages might necessitate a review of earlier work.

- Step 1. Identify field of interest. At the very beginning, *the method of maintenance of materialized views* and *the optimization of online analytical processing* were considered as the general field of interest which we intend to research.

- Step 2. Conduct field of study. To convert these general ideas into possible topics in detail, we have conducted extensive the field of study through a literature review. Surprisingly, we found that almost all proposed solutions to the problem of maintaining a materialized view in a distributed environment assumed that the data changes required by the algorithm somehow were made ready for use. We believe this assumption is not always realistic, especially when the source systems are highly autonomous and spread over the wide area networks. Consequently, the research problem was narrowing down as to develop a strategy for delta extraction in a limited collaborative environment. The key challenge has been identified as the huge volume of source data spread over a wide geographic area and the lack of collaboration between data warehouse and individual data sources in a distributed environment. The goal of research is to develop a strategy which is able to complete the task of delta extraction in an efficient way. By saying efficient, it means to minimize communication costs as much as possible.

- Step 3. Undertake feasibility study. Based on the an outlined field of study, This feasibility study was to assess and determine whether the problem can be resolved in an efficient way. To answer this question, according to the nature of research problem, we conducted the systematic literature review. From the

literature review, we have created the taxonomy of collaboration levels in a distributed environment, reviewed data change detection methods that are currently available, and also drew attention to the strengths and limitations of each method through analysis. We found that using database trigger and transaction log are ideal ways to capture data changes, however, in a limited collaborative environment, these handy options simply are not available. In addition, the snapshot differential algorithm uses two optimization techniques, namely, data compression and the window algorithm to generate a differential file. These two optimization techniques are performed on individual tuple level, therefore we believe the improvement can be achieved by developing a group strategy, particularly, when data sources do not change frequently. The group strategy was based on the well-defined probability theory.

- Step 4. Establish a framework. The framework of this thesis contains three key components, namely, *2-staged group hash method*, *using progression pattern* and *nested group hash method*.

    1. The goal of 2-staged group hash method is to minimize communication costs as much as possible. We focused on the model where the likelihood of any given tuple of data source to be changed is available prior to the design of the procedure of delta extraction. The design of group hash algorithm was based on the the analysis of expected changes, i.e. prior probability. We have considered two broad scenarios: one is when the changes are uniformly distributed and the other is to deal with the heterogeneous mixtures of data changes. In both cases, we have developed the cost-driven strategy. In the former, the idea is to find the group plan that requires the minimum volume of data, which has been solved by using extreme value theorem; in the latter, the idea is to find the group plan that yields maximum savings. Briefly speaking, it is to solve the problem that, given expected changes,

what is the group cardinality that would lead to maximized saving or lead to minimized costs.

2. Defining and using progression pattern with group hash method is to handle the outlier and relax the assumption that the data changes at source systems are only caused by random events. The progression pattern is defined to describe data change with temporal regularities. To maintain the highest throughput of reporting and analytical functionality in a data warehouse, instead of finding a complete set of the underlying data patterns in one go, the discovery of progression pattern was designed to be a process of forming an explanatory hypothesis from an observed irregularity.

3. The purpose of nested group hash method is to seek the possibility of continual improvement. The design approach of nested group hash algorithm is fundamentally different from the approach used for 2-staged group hash algorithm in that firstly, the nest group hash algorithm is a multi-staged algorithm and secondly, the dependencies across the different stages need to taken into account since, the later tests according to the result of previous tests. The efficiency is improved at the expense of additional data processing.

- Step 5. Confirming the validity. We proved that the all components in the framework are valid answer to the research problem by selecting appropriate methods. More details will be discussed in next subsection.

3.2 Research Methods

This research is characterized as an empirical discipline [22, 72], and the framework we used to establish the research question involves the definition of algorithms, statistical model and implemented prototypes. To fully understand the problem, as part of research design, we have adopted the combination of various research methods. We believe the selected methods would lead to acceptable evidence in response to our research problems, which include mathematical proof, comparative studies, controlled experiments and combinations of the above.

Comparative study is the act of comparing two or more subjects with a view to discovering the difference and to reveal the general underlying structure which generates or allows such a variation. A controlled experiment is an investigation of a testable algorithm where one or more independent variables are manipulated to measure their effect on one or more dependent variables [28]. We test the framework on synthetic data, which is the source of ground truth and on which the performance of algorithms can be assessed objectively.

We choose to use synthetic data in our experiments in that, firstly, synthetic data can be created in a controlled manner to meet specific needs or certain conditions that may not be found in the original, real data [60]. For example, we need to verify the performance of group hash algorithm at a range of specific data change rates respectively. In this case, the creation of synthetic data can be controlled to satisfy the requirement of these specific theoretical values. The created synthetic data here is used as a simulation of real data. Secondly, synthetic data is a subset of anonymized data, which would avoid to compromise the confidentiality of particular aspects of the data [64].

3.2.1 2-staged group hash method

1. Cause-effect relationship between group cardinality and communication costs. According to the formally defined cost model in Section 5.3, when data change is uniformly distributed, given a prior probability, there is a cause-effect relationship exists between *group cardinality* and *communication costs*. In Section 5.5.2, the controlled experiment was conducted to investigate the relationships between cause construct, i.e. group cardinality, and effect construct, i.e. communication costs. The design of the controlled experiment is summarized as follows

   - Independent variable: group cardinality.
   - Dependent variable: communication costs.
   - Controlled items: a prior probability, size of tuple, size of key and size of hash value.

   The controlled experiments reduced complexity by allowing only single independent variable and dependent variable while controlling all others. The experiment measured the effect of the treatment and shows precisely a cause-effect relationship exists between group cardinality and communication costs as expected.

2. Comparative studies. This comparative study was used together with the controlled experiment in Section 5.5.2, to answer the questions that among different methods, which one is more efficient in terms of communication costs, where are the differences and what are the tradeoffs. The methods to be compared are

   - Method 1: Complete refresh.
   - Method 2: Data compression.
   - Method 3. 2 stage group hash method.

   Based on the research question, the interesting properties that we predefined to record for each of the methods are summarized as follows

- Efficiency. It is measured in terms of *communication costs*.
- Tradeoff. It is measured by *no. of comparison required* indicating the costs of data processing.
- Controlled items. A prior probability, size of snapshot and size of delta file.

The results were shown in the form of tables and the differences became the focus of the studies.

3. Mathematical proof. In Section 5.4.1, an adaptive function has been developed to estimate individual prior probabilities when dealing with heterogeneous mixtures of data change. The function starts with an empirical value and also takes into account the dynamics in the course of data changes. The function has been generalized in equation (5.28). The Bernoulli's law of large numbers is used to prove that the estimation results generated by the function converge to a stable, desired state.

3.2.2 Using Progression Pattern

This comparative study was used together with the controlled experiment in Section 6.5.1, to answer the questions what would be the negative impacts caused by outlier. To understand the problem in precise terms, we have conducted two different comparisons, in terms of *communication costs* and *the costs of data processing* respectively. Both experiments have the same controlled setting, which includes a predefine prior probability, size of tuple, size of key and size of hash value. In addition, the set of deviation from the expected data change probabilities were chosen from 0.1 and up to 0.6. Regarding to communication costs, the experiment design was designed to compare between

- the minimized communication costs, i.e. the one with the deviation of zero and

- the communication costs that are associated with every deviation value listed above.

Regarding to the costs of data processing, the experiment was designed to compare between

- No. of queries required when the deviation is zero and
- No. of queries required that are associated with every deviation value listed above.

The results were in the form of graphs which was chosen to reveal trends of the negative impacts caused by outlier.

3.2.3 Nested Group Hash Method

This comparative study was used together with the controlled experiment in Section 7.4.3, to answer the questions that between 2-stage group hash method and nested group hash method, which one is more efficient in terms of communication costs, where are the differences and what are the tradeoffs. The methods to be compared are

- Method 1: 2 stage group hash method.
- Method 2. Nested group hash method.

To understand the problem in precise terms, interesting properties that we predefined to record for each of the methods are summarized as follows

- Efficiency. It is measured in terms of *communication costs*.
- Reliability. It is measured in terms of the standard deviation between the expected communication costs and the results collected from the actual executions.
- Tradeoff. It is measured by *no. of queries required* indicating the costs of data processing.

- Controlled item. It includes size of snapshot and size of delta file.

In addition, to have comprehensive view of the problem, the experiments have been conducted with a set of different prior probabilities setting, including 0.01, 0.03, 0.05 and 0.07. The results were in the form of tables and graphs, where the differences were highlighted and the trends were revealed.

Chapter 4

Preliminaries

4.1 Conceptual Model

To provide a visual representation of the overall framework of our approach, we present a high level conceptual model as shown in Figure 4.1. This conceptual model shows the fundamental system elements, processes and the interactions between them, from which we can see that the data warehouse and remote database systems are two major components of the problem domain. A materialized view is maintained in a data warehouse and contains integrated data from remote database systems. To reflect the data changes in the source systems and make the data up to date, the materialized view needs to be refreshed regularly or on demand. A differential file is used to refresh the materialized view, which is generated by the process of delta extraction. We focus on the situation where a differential file is inferred by comparing the current state of data in materialized view to the associated data sources.

Very often, a large volume of data is involved in the process of delta extraction, therefore the optimization goal is to minimize the communication costs. Firstly, 2-stage group hash algorithms are designed to be used in the data warehouse system as an optimized solution to the task of delta extraction. To seek the possibility of continual improvement, a nest group hash algorithm is also designed for the task of delta extraction. A nest group hash algorithm is a multiple staged algorithm, where at the expense of some additional data process, it can further reduce the communication

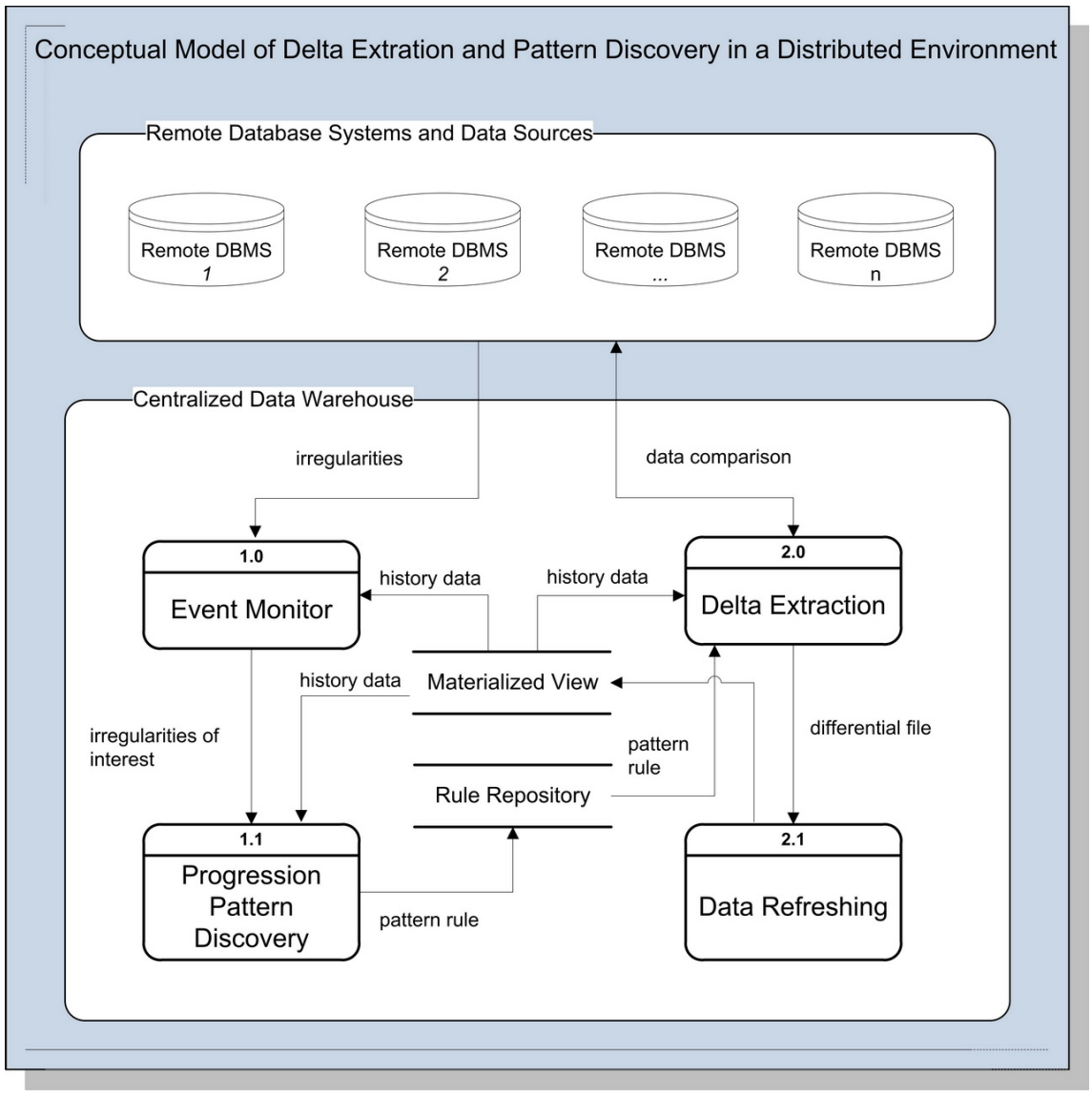

Figure 4.1: Conceptual Model

costs.

The group hash algorithms are an highly efficient solution for task of delta extraction. When using group hash algorithms, an important assumption was made that the changes at source systems are only caused by random events, therefore the algorithms were not designed to recognize or to use any data pattern when dealing with the evolution of data. To relax this assumption and accommodate to the data change patterns, the functionality of pattern discovery is also included in the framework. Basically, from the ongoing view maintenance processes, all data irregularities detected by event monitor will trigger the process of pattern discovery. The discovered data change pattern in turn will be used in the process of delta extraction to avoid the negative impacts caused by outlier data.

4.2 Data Warehouse Environment

In this research, we focus on the PSJ materialized view, i.e. PSJ expressions over source relations, which is defined by using the relational operators of **p**roject Π, **s**elect σ and **j**oin $\bowtie$. The PSJ views are the most commonly used for the purpose of data integration, therefore we believe it is a good problem area to be focused on initially. We assume that either the unique key or the primary key of source relation is included in the view definition, the read-only access to the remote source tables is granted and a standard hash function is available at the remote database systems. This assumption is realistic because of ready availability of source code implementations. For example, as listed in [73], Oracle Database provides standard hash functions such as SHA-1, SHA-2, MD4 and MD5, etc and DB2 hash routines [46] include MD5 and SHA-1, etc.

A data warehouse environment is viewed as $\mathcal{WH} = (\mathcal{V}, \mathcal{DB}, \mathcal{L}, R)$, where each $\mathcal{V}_i \in \mathcal{V}$ is a materialized view to be maintained, $\mathcal{DB}$ is a set of remote source systems involved and $\mathcal{L}$ is a set of communication links. Each link $l_i \in \mathcal{L}$ is described by the pair of nodes it connects, its capacity and unit cost. In a limited collaborative environment, a link only exists between the data warehouse system and a remote source system. In other words, the individual source systems know neither the existence of others nor how to communicate with them. Materialized view contains the integrated data from the set of source relations $R = \{R_1, R_2, ..., R_n\}$, which spread over the remote sites. Each source relation $R_j \in R$ belongs to a remote system $\mathcal{DB}_i \in \mathcal{DB}$ and each $\mathcal{DB}_i$ consists of at least one, possibly more source relations.

4.3 View Decomposition and Snapshot Projection

4.3.1 View Decomposition

We consider to decompose the view definition into a set of subqueries $Q_1, Q_2, ..., Q_n$ as a preliminary step. In work [96], a query was proposed to be decomposed into a sequence of irreducible queries, i.e. single variable queries. That method was designed for query processing in a centralized database. The view decomposition in this work is to decompose a view definition into subqueries, such that the data needed by each subquery are localized to each individual remote site. Accordingly, the granularity of decomposition is that the subqueries are local to individual remote sites, rather than to be irreducible queries. The first step of view decomposition is the data localization, which is to detach the source relations from a view definition, such that the detached subqueries are localized to individual remote sites. The next step is to push restriction and local join conditions into the detached subqueries if applicable. The distributed join is resolved by tuple substitution, which corresponds to a nested loop join, with a detached subquery as the outer operand and the matched value set as the inner. Finally, assigning the projection operator to ensure only attributes that are either in the list of the view definition or in the distributed join conditions are included. When the decomposition is complete, every decomposed subquery Q_i can be directly evaluated at a single remote site.

Considering the case that a university has a remote campus and maintains a materialized view about remote subject enrollment, a materialized view contains three relations as follows:

- Std_Personal_Detail (*sno*, sname, dob, address)
- Std_Course (*stdcrs_id*, sno, crs_name, crs_type)
- Remote_Enrollment(*enrol_id*, stdcrs_id, subj_name, mark)

Let us assume the all student personal information and student course information are maintained at the main campus, e.g. site 1, and the relation Remote_Enrollment is located at the site 2. The business rules say that a student can enroll one or more student courses and each of them is related to multiple subject enrollments. The view definition and the steps of view decomposition are shown as follows

```
--V_Remote_Enrollment
Select sno, sname, dob, stdcrs_id, crs_name, enrol_id,
       subj_name, mark
From   Std_Personal_Detail P,
       Std_Course C
       Remote_Enrollment E
Where P.sno = C.sno and
      C.course_type = 'UG' and
      C.stdcrs_id = E.stdcrs_id.
```

```
--Decomposition Step 1: Data Localization
Q1: Select sno, sname, dob,
           stdcrs_id, crs_name
    From   Std_Personal_Detail P,
           Std_Course C
    Where P.sno = C.sno and
          C.course_type = 'UG'

Q2: Select enrol_id, subj_name,
           mark
    From   Remote_Enrollment E,
           Svar S
    Where E.stdcrs_id = S.stdcrs_id
```

```
--Decomposition Step 2: Tuple Substitution
Q3: Select C.stdcrs_id
    Into   Svar
    From   Std_Personal_Detail P,
           Std_Course C
    Where  P.sno = C.sno and
           C.course_type = 'UG'

Q4: Select enrol_id, subj_name,
           mark
    From Remote_Enrollment E
    Where E.stdcrs_id in
    (Select stdcrs_id
       From Svar)
```

In the example above, Q1 and Q4 are the decomposed subqueries from view definition,

such that each of them can be evaluated at its local database system. However, the intermediate relations *Svar* needs to be transferred from site 1 to site 2 to complete the execution of Q4. This example is very simple, which just aims to provide an insight to the idea of view decomposition. With inconsistent data sources or with complex predicates, the view decomposition can be a very interesting research problem. In the research fields of query rewriting [38, 16] and distributed query processing [10, 54, 76, 88], these problems have already been addressed extensively and we will not discuss further in this thesis.

4.3.2 Snapshot Projection

Snapshot projection is an operation performed at data warehouse system, which is to generate a set of snapshots corresponding to the decomposed subqueries. For a decomposed subquery Q_i, its snapshot $\mathcal{S}_i$ at data warehouse system can be acquired by assigning a projection operator to the materialized view and making the attribute list identical to one that the subquery Q_i holds, that is $\mathcal{S}_i = \Pi[\alpha_i](\mathcal{V})$. Every $\mathcal{S}_i$ represents the data fragment that is being included as the part of current materialized view and has the identical schema as Q_i. In what follows, we refer to $\mathcal{S}_i$ as the snapshot of a decomposed query Q_i. Using the example above, the corresponding snapshots of the decomposed queries Q1 and Q4 are obtained from

```
--S1  the snaoshot of Q1
Select distinct sno, sname, dob, stdcrs_id, crs_name
From   V_Remote_Enrollment

--S4  the Snaoshot of Q4
Select enrol_id, subj_name, mark
From V_Remote_Enrollment
```

4.4 Differential File

In the relational model, a relation R_j is a set of n-tuples $\{t_1, t_2, ..., t_n\}$. We use the vertical bars to indicate set cardinality, i.e. $|R_j|$ represents the number of tuples in R_j. A tuple $t_i \in R_j$ is a pair $t_i = (k, v)$, where $t_i.k$ and $t_i.v$ refer to key and non-key attributes respectively. Insertion, deletion and update on data sources cause the inconsistencies of materialized view. Based on the key constraints, three types of inconsistencies are interpreted as follows:

- insertion, denoted by t_j^+, that is $\exists t_j \in Q_i, \forall t'_j \in \mathcal{S}_i$ such that $t_j.k \neq t'_j.k$.
- deletion, denoted by t_j^-, that is $\exists t'_j \in \mathcal{S}_i, \forall t_j \in Q_i$ such that $t_j.k \neq t'_j.k$.
- update, denoted by $t_j^\pm$, that is $\exists t_j \in Q_i, \exists t'_j \in \mathcal{S}_i$ such that $t_j.k = t'_j.k$ and $t_j.v \neq t'_j.v$.

A differential file $\Delta(R_i, R'_i)$ is the representation of delta, which contains the differences between relations R_i and R'_i and is in the form of

$$\Delta(R_i, R'_i) = (\Delta^+(R_i, R'_i), \Delta^-(R_i, R'_i)) \tag{4.1}$$

where $\Delta^+(R_i, R'_i) = R_i - R'_i$ and $\Delta^-(R_i, R'_i) = R'_i - R_i$. Both R_i and R'_i are query expressions that have an identical relation schema. A differential file is often used for the purpose of data synchronization [84]. Given R'_i and $\Delta(R_i, R'_i)$, R_i can be reconstructed as follows:

$$R_i = (R'_i \cup \Delta^+(R_i, R'_i)) - \Delta^-(R_i, R'_i) \tag{4.2}$$

A differential file $\Delta(Q_i, \mathcal{S}_i)$ can be inferred by comparing a decomposed query Q_i to its snapshot $\mathcal{S}_i$ and this process is referred to as *delta identification*. Delta identification is a sub-process of *delta extraction* that is also responsible for transporting differential files to the data warehouse. *Data synchronization* is to apply the differential file to the current materialized view, such that the materialized view becomes consistent with the current state of data sources.

4.5 Ideal Properties of the Hash Function

Hash function $h : \mathcal{M} \longrightarrow \mathcal{O}$ is a well-defined mathematical function which maps an arbitrary finite length of message $\mathcal{M}$ to a fixed length hash value $\mathcal{O}$. A hash value is used as a compact representative image of an input message, or a hash value can also be used as if it were uniquely identifiable with that input message. This property is referred to as *compact message and collision resistance* [83]. Let $size(h)$ represent the length of a hash value, typically, two randomly chosen inputs would yield the same output with probability $2^{-size(h)}$. As an example, the classic hash algorithm SHA1 [87] takes an input message of up to 2^{64} bits and produces a 160-bit hash value. This property makes it possible to use much smaller of data to perform the data comparison without sacrificing much precision. The second property of the hash function we are interested in is referred to as *flexible input size*, which means that hash function can take an arbitrary finite length of input. The properties mentioned above make the hash function very ideal for the purpose of data comparison over the wide area networks. For the task of delta extraction, in terms of minimizing the volume of data transfer, the hash function is used to make the message in a compact form, while our focus is to develop the group strategies that are able to reduce the number of messages required.

Chapter 5

Group Hash Method

5.1 Overview

By using a group hash method, the delta is identified by comparing the hash value of a group of tuples, rather than by comparing each of the individual tuples. Typically, a large portion of data that remains unchanged between two consecutive snapshots [56]. The general assumption when we design the group hash algorithm is that usually the unchanged portion of data in a snapshot vastly outnumber the changed one. The basic idea of the group hash method is to send the hash value of a group of tuples along with key identifiers to the remote site for the group comparison. To explain its benefits, assuming that a group contains 10 tuples, the average tuple size is 200 bytes, the average key size is 7 bytes, and the size of hash value is 16 bytes. Provided that no tuple in this group has been changed, only $7 \times 10 + 16$ bytes, i.e. 4.3% of total amount of data volume, are required to complete the task of delta extraction. Comparing to the method of complete refresh, we can expect to save $200 \times 10 - (7 \times 10 + 16) = 1914$ bytes , and the saving ratio is $\frac{1914}{200 \times 10} = 95.7\%$. In this example, we see that, when no change occurs in a group, the communication costs for the task of delta extraction can be significantly reduced. Some questions also arise immediately, by taking account of the changed data existing in a snapshot,

1. How will the group hash method perform?

2. What is the most efficient size for the groups and how to decide it?

5.2 Basics of Group Hash Method

5.2.1 Problem Formulation

Given the decomposed query Q_i and its snapshot $\mathcal{S}_i$, let G_j be a set of tuples and $G_j \subseteq \mathcal{S}_i$, and $G_j\kappa$ be all keys of G_j. Also let $G'_j \subseteq Q_i$, and $G'_j\kappa$ be all keys of G'_j.

- *Peer Groups* refer to two groups G_j and G'_j that are about to compare against each other. When used in the singular form, we can say that group G' is the peer group of group G_j, and vice versa.

- *Matched Groups*, denoted by $\ominus_j$, refer to peers groups G_j and G'_j that have a set of identical keys, i.e. $G_j\kappa = G'_j\kappa$. Being matched groups implies that two groups have the same cardinality, i.e. $|G_j| = |G'_j|$, and there is a one to one correspondence between $G_j\kappa$ and $G'_j\kappa$. In the singular form, we can say that group G' is the matched group of group G_j, and vice versa.

- *Identical Groups*, denoted by $\equiv_j$, refer to the match groups that have identical hash values, ie. $h(G_j) = h(G'_j)$. In the singular form, we can say that group G' is the identical group of group G_j, and vice versa.

- *Changed Group*, denoted by cg, refers to any group $G_j \subseteq \mathcal{S}_i$ that does not have any identical group.

The information about which tuples in a snapshot $\mathcal{S}_i$ has been changed can be viewed as a vector

$$\mathbf{x} = (x_1, x_2, ..., x_n) \tag{5.1}$$

where the possible value for every element x_j is one of the binary values 0 or 1. More specifically, $x_j = 1$ if the tuple t_j is either deleted or updated, else $x_j = 0$. The task of delta identification is to find out the positions of 1's and 0's in the vector. In the group hash method, the snapshot is partitioned into a set of groups and the hash value

of each partitioned group is sent to remote site and compared with its peer group, which means that, initially, the group is used as an unit to be compared, rather than individual tuples.

The function of group comparison, denoted by $cf(G_i, G_i')$, formally is defined as

$$cf(G_i, G_i') = \begin{cases} 1 & \sum_{k=1}^{|G_i|} x_k \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

which means that if there is any tuple in G_j either being deleted or updated, then the function returns 1, i.e. identified as a changed group. The implementation of this function is to evaluate the hash values of peer groups, i.e. if $h(G) = h(G')$, thus in the case the value 1 is returned from the function, it is still not known that which one causes the change and how many are changed. On the other hand, if the value 0 is returned, it concludes that the identical groups are established.

5.2.2 Goal and Challenges

The goal of group hash method is to unambiguously identify each of the changed tuples in a snapshot and in an efficient way. By saying efficient, it means to minimize the communication costs. The motivation of using the group hash method is that two consecutive snapshots are not very different. However, when designing an efficient group hash algorithm, we also need to consider the negative impacts caused by the changed data. For example, the deletion causes the failure of establishing matched groups, and the update causes the failure of establishing the identical group even if the matched groups have been established. To ensure that the goal that can be achieved, in the data warehouse system, a group strategy must be developed.

The group strategy is a critical part of the overall optimization, in which the main task is to make a group plan. A group plan specifies how to divide a snapshot into a set of disjoint groups, such that the volume of data required for delta extraction is optimized. The properties of group have made the generation of group plan an

interesting problem. When generating a group plan, one option is to use larger group cardinality, i.e. including more tuples in a group, then intuitively, it would lead to more data saving when the identical groups could be established. However, at the same time, the larger cardinality also implies the less chance of establishing identical groups. Another option is to use smaller group size, which would increase the chance of data saving, but at the same time, lead to less data saving per group. Briefly, when designing a group hash algorithm, the expected saving per group and the likelihood of yielding the saving are two correlated factors that have to be considered.

To resolve this inherent dilemma, firstly, the statistics about change probabilities of remote data need to be maintained at the data warehouse system. Maintaining the statistics certainly requires some extra efforts in the data warehouse system, but the collected statistics can be used to construct a simple model of the prior probabilities, which will help us to capture the essential characteristic about remote data. The essential characteristic of data reflected by the statistical model in turn will allow us to conduct the analysis in a systematical manner. We have considered two broad scenarios: one is when the data changes are uniformly distributed and the other is the heterogeneous mixtures of changed data at data sources. When dealing with uniformly distributed, the idea is to find the group plan that requires the minimum volume of data to complete the task of data extraction. On the other hand, for the second scenario, the idea is to find the group plan that yields maximum savings. We start by introducing the processes of delta identification.

5.2.3 Process of Delta Identification

In this chapter, as shown in Figure 5.1, we are using a 2-staged delta identification process. Firstly, the notions of Q_i and $\mathcal{S}_i$ are used to represent a decomposed query and its corresponding snapshot respectively.

Stage 1. At a data warehouse system, a snapshot is partitioned into a set of disjoint and nonempty groups. It aims to identify all changed groups, but at the same time,

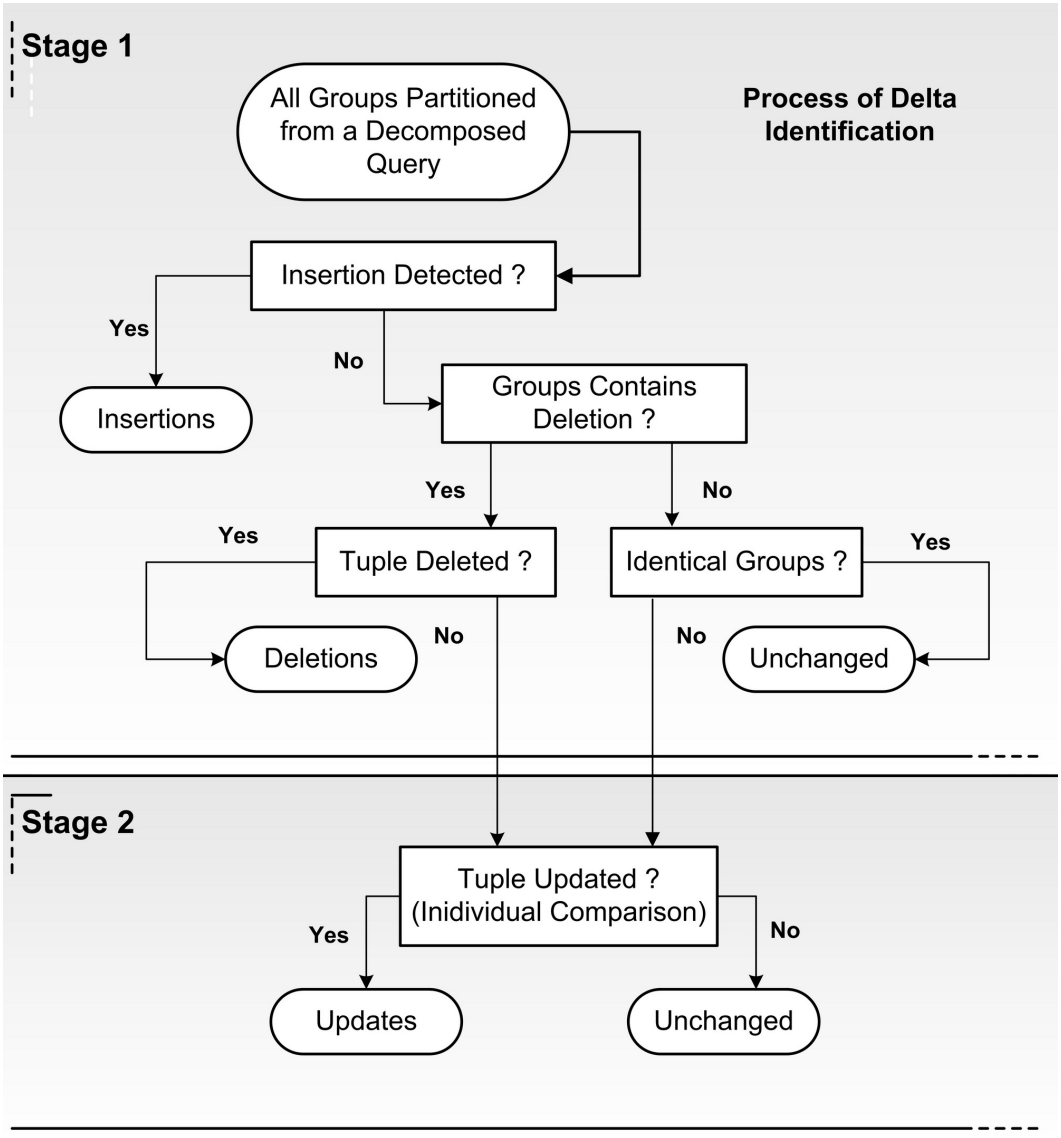

Figure 5.1: Process of Delta Identification

all insertion, deletion and identical groups are also identified. For an individual group G_j, the information that needs to be sent to remote site for the group comparison consists of all keys in the group and the hash value of the group, which form a basic data unit for the comparison and is represented by $\langle G\kappa_j, h(G_j)\rangle$. All data units are sent to the remote site where the associated decomposed query is located. Let $Q_i\kappa$ and $\mathcal{S}_i\kappa$ represent all keys in the decomposed query and the snapshot respectively.
The insertion, denoted by $\mathcal{S}_i^+$, is obtained from

$$\mathcal{S}_i^+ \longleftarrow \sigma\ (Q_i \bowtie_{k=k} (Q_i\kappa - \mathcal{S}_i\kappa)) \tag{5.3}$$

The deletion, denoted by $\mathcal{S}_i^-$, is obtained from

$$\mathcal{S}_i^- \longleftarrow \mathcal{S}_i\kappa - Q_i\kappa \tag{5.4}$$

Both $\mathcal{S}_i^+$ and $\mathcal{S}_i^-$ are part of the differential file $\Delta(Q_i, \mathcal{S}_i)$ and more specifically

$$\mathcal{S}_i^+ \subseteq \Delta^+(Q_i, \mathcal{S}_i) \tag{5.5}$$

$$\mathcal{S}_i^- \subseteq \Delta^-(Q_i, \mathcal{S}_i) \tag{5.6}$$

The insertions do not affect the result of the data comparison for all group units composed from the snapshot. On the other hand, for the detected deletions, without further testing, the conclusion can be made that the group that contains the deletion is a changed group. By removing all the groups that containing deletion, the remaining groups, denoted by η, need to be compared with their peer groups in order to verify which one contains the update. Let η_c denote the changed groups in η, finding $\eta_c \in \eta$ is straightforward and is described by Algorithm 1. To perform the group comparison, the first step is to establish the matched groups, the function to locate the matched groups of G_j is defined as:

$$G'_j \longleftarrow \sigma_{G_j\kappa \subseteq Q_i\kappa}(Q_i) \tag{5.7}$$

Algorithm 1 Group Comparison Algorithm

```
    procedure IdentifyChangedGroup(Q_i, η)
2:      η_c ← ∅                                   ▷ Changed groups as output
        for each group G_j ∈ η do
4:          G'_j ⟵ σ_{G_jκ ⊆ Q_iκ}(Q_i)
            if cf(G_j, G'_j) = 1 then             ▷ Update identified
6:              η_c ← η_c ∪ {G_j}
            end if
8:          i++
        end for
10: end procedure
```

Stage 2. At this stage, it is to resolve all changed groups identified from the previous stage. All tuples in the changed groups need to do the individual comparison in order to identify the updated tuples. The updates are also part of a differential file, which is in the form of $\mathcal{S}_i^{\pm} = \mathcal{S}_i^{+'} \cup \mathcal{S}_i^{-'}$, where $\mathcal{S}_i^{-'}$ denotes the expired data that are to be deleted from the snapshot and $\mathcal{S}_i^{+'}$ denotes the new data to be incorporated into the snapshot.

When the 2-staged process is complete, a differential file can be generated by using equations (5.8) and (5.9) as follows:

$$\Delta^{+}(Q_i, \mathcal{S}_i) = \mathcal{S}_i^{+} \subseteq Q_i \cup \mathcal{S}_i^{+'} \subseteq Q_i \tag{5.8}$$

$$\Delta^{-}(Q_i, \mathcal{S}_i) = \mathcal{S}_i^{-} \subseteq \mathcal{S}_i \cup \mathcal{S}_i^{-'} \subseteq \mathcal{S}_i \tag{5.9}$$

In the process of delta identification, both locating a group and locating an individual tuple at source data are using the key identifier. Typically, the indexes have been created and built on the key attribute in a relational database, therefore the performance of the process should be reasonably efficient.

5.2.4 Properties of Group

The statistics about data sources are maintained at the data warehouse system as follows, for each process of data synchronization,

- update rate per hundred tuples in a snapshot, which is used as the probability of a randomly selected tuple being updated, represented by $P(t^{\pm})$.
- deletion rate per hundred tuples in a snapshot, which is used as the probability of a randomly selected tuple being deleted, represented by $P(t^{-})$.

The probability of a randomly selected tuple remains unchanged, i.e. not being updated, denoted by $P(t^{\cong})$, is derived from

$$P(t^{\cong}) = 1 - P(t^{\pm}) \tag{5.10}$$

The probability of a randomly selected tuple free from deletion, denoted by $P(t^{\ominus})$, is derived from

$$P(t^{\ominus}) = 1 - P(t^{-}) \tag{5.11}$$

Symbols	Description
$1 : t^{+}$	tuple added
$2 : t^{-}$	tuple deleted
$3 : t^{\pm}$	tuple updated
$4 : t^{\ominus}$	tuple not deleted
$5 : t^{\cong}$	tuple not updated
$6 : -$	deletion exists in a group
$7 : \pm$	update exists in a group
$8 : \ominus$	matched groups
9: $\cong$	no update exists in a group
10: $\equiv$	identical groups

Table 5.1: Symbol Explanation

In a relational database, typically, the fact that deleting or updating a tuple t_i gives no information about any other tuple t_j, thus deletion and update in this work are considered as *independent event*. For a randomly selected group G_j, during the process of delta extraction, whether deletion would be identified in this group, has two possible outcomes $\{-_j, \ominus_j\}$, where $-_j$ represents the deletion is detected and

$\ominus_j$ represents no deletion. The probability of $\ominus_j$ is an important parameter to our optimization algorithm, denoted by $P(\ominus_j)$, and is understood as the likelihood that: $\forall t_i \in G_j$, such that $t^{\ominus}$, where $t^{\ominus}$ means a tuple is not deleted. From above, $P(\ominus_j)$ is obtained from:

$$P(\ominus_j) = P(t_1^{\ominus}) \times P(t_2^{\ominus}) \times ... \times P(t_n^{\ominus}) \tag{5.12}$$

Similarly, whether update would be identified in this group also has two possible outcomes $\{\pm_j, \cong_j\}$, where $\pm_j$ represents at least one tuple in group G_j is updated and $\cong_j$ represents no update is detected. The probability of $\cong_j$, denoted by $P(\cong_j)$, is understood as the likelihood that: $\forall t_i \in G_j$, such that $t^{\cong}$, where $t^{\cong}$ represents a tuple remains unchanged. From above, $P(\cong_j)$ is obtained from

$$P(\cong_j) = P(t_1^{\cong}) \times P(t_2^{\cong}) \times ... \times P(t_n^{\cong}) \tag{5.13}$$

For peer groups G_j and G'_j, only if $\ominus_j$ holds, they are matched groups. In the case they are matched groups, only if $\cong_j$ holds, then they are identical groups, denoted by $\equiv_j$. The probability of $\equiv_j$ can be derived by using equations (5.12) and (5.13) as follows:

$$P(\equiv_j) = P(\ominus_j) \times P(\cong_j) \tag{5.14}$$

The probability of a randomly selected group G_j being a changed group, denoted by $P(cg)$, is understood as that the likelihood that $\exists t_i \in G_j$, such that $t^- \vee t^{\pm}$, where t^- and $t^{\pm}$ represent a tuple deleted and updated respectively. From above, $P(cg)$ is derived from equation (5.14) as follows:

$$P(cg) = 1 - P(\equiv_j) \tag{5.15}$$

5.3 Uniformly Distributed Data Change

According to the process of delta identification specified in Section 5.2.3, the cost model will be specified to estimate the incurred communication costs. The estimated costs will be used as the basis when selecting the optimized group plan.

5.3.1 Cost Model

The first scenario we consider is that the data changes at source systems are uniformly distributed. A snapshot with cardinality $|\mathcal{S}_i|$ is partitioned into a set of disjoint groups, called *initial groups*. In this case, the group plan is characterized by all initial groups having the equivalent cardinalities, denoeted by $|G|$. The number of initial groups is

$$N = \frac{|\mathcal{S}_i|}{|G|} \tag{5.16}$$

Let $sz(k)$ and $sz(h)$ represent the lengths of key and the length of hash value respectively, in the process of delta identification, the communication costs are incurred at the stage 1 are

$$C_1 = (sz(k) \times |G| + sz(h)) \times N \tag{5.17}$$

At stage 2, it is to resolve the changed groups identified from the stage 1 by conducting the individual comparisons. Firstly, given the collected statistics of $p(t^{\pm})$ and $p(t^{\pm})$ of the snapshot, out of N initial groups, from equation (5.15), the expected number of change groups is estimated as

$$N_{cg} = N \times P(cg) \tag{5.18}$$

The individual tuples that are included in the changed groups identified at stage 1 are required for the further comparison. The number of these individual tuples is estimated as

$$N_{idv} = N_{cg} \times |G| \tag{5.19}$$

The estimation of communication costs incurred at stage 2 is obtained from

$$C_2 = (sz(k) + sz(h)) \times N_{idv} \tag{5.20}$$

Note that in this 2-staged procedure, the costs incurred at stage 1, i.e. C_1, is a predetermined value, while the costs incurred at stage 2 are the expected values based on the probability theory. By combining the costs incurred at stage 1 and at stage 2, the total communication costs for the task of delta identification are:

$$C = C_1 + C_2 \tag{5.21}$$

5.3.2 Group Plan Selection

Based on the cost model, the problem of group plan selection is that: when the data changes are uniformly distributed, given the probabilities of $P(t^{\pm})$ and $P(t^{-})$ of snapshot $\mathcal{S}_i$, to find the group cardinality x, such that the selected values of x lead to the minimized value of C.

Method 1. We use the *extreme value theorem* to solve this problem. Let $f(x)$ represent the cost function and let $f'(x)$ be the derivative of $f(x)$ with respect to the variable x. Basically, the minimized value of C occurs at critical points where either $f'(x) = 0$ or $f'(x)$ is undefined. To find the optimized group cardinality, i.e. value of x, firstly combining equations (5.17) and (5.20)

$$f(x) = (sz(k) \times x + sz(h)) \times \frac{|\,\mathcal{S}_i\,|}{x} + (sz(k) + sz(h)) \times (|\,\mathcal{S}_i\,| \times (1 - (P(t^{\cong})^x) \times P(t^{\ominus})^x) \tag{5.22}$$

Let C be the value of $f(x)$ and let symbol d read as *change in*, then $f'(x)$ is

$$f'(x) = \frac{dC}{dx} \tag{5.23}$$

To find the optimized group cardinality, firstly, set $f'(x) = 0$, then the value of x can be either observed from the graph of the equation or going through the process of solving the equation.

Method 2. Algorithm 2 uses a different approach to find the cost-effective group plan, which simply chooses the best one among different group plans. For each different group plan, the costs are calculated from group cardinality being one, incrementally

until the evidence is shown that no need to proceed further. When the optimized value of costs C is found, the associated group cardinality is considered to be the cost-effective group plan. A threshold is also predefined, which is set to the costs when the cardinality is one. The algorithm terminates, if it costs more than the predefined threshold. A function $f_c : x \rightarrow C$, is defined to calculate the costs C in the situation of $|\, G \,| = x$.

Algorithm 2 Group Selection Algorithm

```
1:  procedure GroupSelection
2:      x ← 1
3:      |G| ← 1
4:      notFound ← True
5:      C_min ← f_c(x)
6:      while notFound = True do
          x ← x + 1
7:          if f_c(1) < f_c(x) then
8:              exit                          ▷ Terminates the process
9:          end if
10:         if C_min ≥ f_c(x) then
11:             C_min ← f_c(x)
12:         end if
13:     end while
14: end procedure
```

5.4 Heterogeneous Mixtures of Data Change

It is common that in a relational table, some portion of data may never be changed once or may only infrequently be changed after being created. For example, a table called student-personal-details in an university, when a person is enrolled in a course provided by the university, a data entry is created in the table, which records this student's first name, middle name, last name, title, date of birth, contact phone number, physical address, mailing address..., etc. The records in this table are updated for many reasons. Some records in the table are frequently updated, because students frequently change their addresses or contact phone numbers. On the other hand, some records are relatively static after being created. Furthermore, year after year, many students complete their degrees, but their records are still kept in this table for historical-reference purposes. These records will remain unchanged unless students register a new course. The proportion of this kind of data in the table will be increased gradually. In other words, this table may include a significant amount of data being static. In the case that a large amount of static data existing in data sources, it is inefficient to treat static and frequently changed data in the same way.

The design of the group hash algorithm for heterogeneous mixtures of data change is credited with accommodating the individual priors of update, which aims to find the optimized group plan for the situation where all tuples in a snapshot are with individual prior probabilities of update. Note that the individual priors are only maintained for the update, but not for the deletions, because deletion is not an ongoing event. Instead, the probability of deletion is still maintained as uniform distribution.

5.4.1 Estimation of Probabilities

When dealing with heterogeneous mixtures of data change, the data warehouse system needs to keep track of the probabilities of data changes for all individual tuples in a snapshot from the ongoing maintenance processes. Our method of estimating the

individual prior probabilities can be viewed as an adaptive function that starts with an empirical value and also takes into account the dynamics in the course of data changes. A tuple $t_j \in \mathcal{S}_i$ will be used as an example to explain the method. The notation $P_{j[n]}$ denotes the probability of updating tuple t_j at the nth synchronization process. Initially, an empirical value $P_{j[0]}$ is given to t_j. After the first time, it has

$$P_{j[1]} = \begin{cases} P_{j[0]}\frac{1}{2} & \text{if } t_j^{\cong} \\ P_{j[0]}\frac{1}{2} + \frac{1}{2} & \text{if } t_j^{\pm} \end{cases} \tag{5.24}$$

Every time when new evidence is available, the estimation will be adjusted to ensure it converges to a stable, desired state. The estimation of $P_{j[n]}$ is obtained from

$$P_{j[n]} = \begin{cases} P_{j[n-1]}\frac{n-1}{n} & \text{if } t_j^{\cong} \\ P_{j[n-1]}\frac{n-1}{n} + \frac{1}{n} & \text{if } t_j^{\pm} \end{cases} \tag{5.25}$$

Suppose t_j has been monitored $[n]$ times, then based on equation (5.25), the estimation of t_j at $[n+1]$ would be

$$P_{j[n+1]} = \begin{cases} P_{j[n]}\frac{n}{n+1} & \text{if } t_j^{\cong} \\ P_{j[n]}\frac{n}{n+1} + \frac{1}{n+1} & \text{if } t_j^{\pm} \end{cases} \tag{5.26}$$

By using this evolutionary approach, the recurrence equation (5.26) for any estimation P_j from $[n]$ to $[n+1]$ can be generalized as:

$$P_{j[n+1]} = P_{j[n-1]}\frac{n-1}{n+1} + \frac{\lambda_{[n\to n+1]}}{n+1} \tag{5.27}$$

where $\lambda_{[n\to n+1]}$ denotes the number of times $t_j^{\pm}$ has happened from $[n]$ to $[n+1]$, i.e. $0 \leq \lambda_{[n\to n+1]} \leq 2$. Furthermore, any estimation P_j from $[0]$ to $[n+1]$ can be generalized as:

$$P_{j[n+1]} = P_{j[0]}\frac{1}{n+1} + \frac{\lambda_{[0\to n+1]}}{n+1} \tag{5.28}$$

where $\lambda_{[0\to n+1]}$ denotes the number of times $t_j^{\pm}$ has happened from $[0]$ to $[n+1]$, i.e. $0 \leq \lambda_{[0\to n+1]} \leq n+1$. To prove this estimation approach is a valid one, it has:

$$\lim P_{j[n+1]} = \lim P_{j[0]}\frac{1}{n+1} + \lim \frac{\lambda_{[0\to n+1]}}{n+1} \tag{5.29}$$

$$\lim_{n\to+\infty} P_{j[0]}\frac{1}{n+1} = 0 \tag{5.30}$$

$$\lim_{n\to+\infty} \frac{\lambda_{[0\to n+1]}}{n+1} = \frac{\lambda_{[0\to n+1]}}{n+1} \tag{5.31}$$

thus

$$\lim_{n\to+\infty} P_{j[n+1]} = \frac{\lambda_{[0\to n+1]}}{n+1} \tag{5.32}$$

and according to Bernoulli's law of large numbers, the estimation results generated by this approach as shown in equation (5.28) converge to a stable, desired state. The process of arriving at the conclusion is based upon the patterns of inference, which is of form that, after observing some evidence, the resulting posterior probability can then be treated as a prior probability, and a new posterior probability computed from new evidence.

5.4.2 Group Plan Generation

In the case of the heterogeneity of data changes, the generation of group plan is based on the analysis of expected benefits. A random selected group G_j in a snapshot will be used an example for the explanations. Firstly, the full size of a group G_j is:

$$sz(G_j) = sz(t) \times |\, G_j \,| \tag{5.33}$$

Using the group hash method, the volume of data required for the group comparison is

$$hz(G_j) = sz(k) \times |\, G_j \,| + sz(h) \tag{5.34}$$

In the case that identical groups are established, i.e. $\equiv_j$, by using the group hash method, the reduced communication costs can be derived as follows:

$$rs(G_j) = sz(G_j) - hz(G_j) = (sz(t) - sz(k)) \times |\, G_j \,| - sz(h) \tag{5.35}$$

Considering the probability of peer groups being identical groups, i.e. $p(\equiv_j)$, equation (5.35) is extended as:

$$ES(G_j) = rs(G_j) \times p(\equiv_j) \tag{5.36}$$

In addition, in the case that any update or deletion are detected, i.e. identical groups not being established, then $hz(G_j)$ used for the comparison is considered to be the

data losses. The chance of data losses is $1 - p(\equiv_j)$. By extending equation (5.36), the expected benefits can be obtained from:

$$EB(G_j) = ES(G_j) - hz(G_j) \times (1 - p(\equiv_j)) \tag{5.37}$$

Algorithm 3 Group Selection Algorithm

procedure GROUPSELECTION($\mathcal{S}_i$)
  $j \leftarrow 1$
  $i \leftarrow 1$
  $G_{tmp1} \leftarrow \emptyset$
  $G_{tmp2} \leftarrow \emptyset$
  $\ell_i \leftarrow f_{\Downarrow p(t^{\pm})}(\mathcal{S}_i)$ ▷ Sorting on probability descending
  while $j < |\ell_i|$ **do**
    $G_{tmp1} \leftarrow f_{\uplus}(G_{tmp1}, t_j \in \ell_i)$
    if $EB(G_{tmp1}) \leq 0$ **then**
      exit ▷ Terminates the process
    end if
    $G_{tmp2} \leftarrow f_{\uplus}(G_{tmp1}, t_{j+1} \in \ell_i)$
    if $EB(G_{tmp1}) > EB(G_{tmp2})$ **then**
      $G_i \leftarrow G_{tmp1}$ ▷ Output: group selected
      $i \leftarrow i + 1$
      $G_{tmp1} \leftarrow \emptyset$
    else
      $G_{tmp2} \leftarrow \emptyset$
    end if
    $j \leftarrow j + 1$
  end while
end procedure

According to the cost model, the algorithm that generates an optimized group plan is shown in Algorithm 3. The idea of generating an optimized group plan is that from the current data set, always find the group with the maximum expected benefits as the next group. Two functions have been used in the algorithm, namely, inclusion function and sorting function. The inclusion function is defined as: $f_{\uplus} : G_a \rightarrow G_b$, where $G_b = G_a \cup \{t_i\}$ is produced by adding a new tuple t_i into a group G_a. The sorting function is defined as: $f_{\Downarrow sv} : (R_i) \rightarrow \ell_i$, where sv is a sorting value, ℓ_i is a list of tuples sorted by sv in the descending order. In the algorithm, the sorting value is the update probabilities of individual tuples. There are two circumstances that cause the

algorithm to terminate. Firstly, when the expected benefits of the new group become negative, in this case all following groups can only have negative values. Secondly, all tuples have been allocated into different groups.

5.5 Experiments

In order to measure the accuracy and feasibility of our proposed methods, we have conducted some experiments and simulations. As we will see, the results demonstrate that frequency estimation function is accurate and the group selection algorithm are quite promising.

5.5.1 Estimating Probabilities

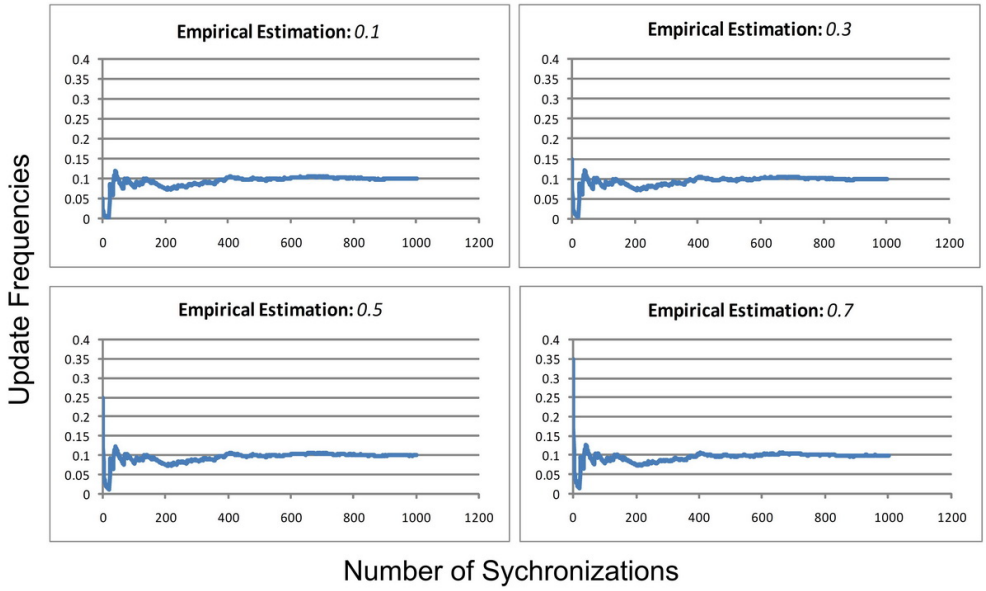

Figure 5.2: Estimation Results

The simulation is designed to verify the estimator function we presented in the Section 5.4.1. For a given initial empirical value assigned to a tuple, 1000 random numbers that are generated by the Oracle function *dbms_random.value* are used to simulate the outcomes of 1000 times of synchronization. Among these numbers, based on a predefined value, 100 numbers are allocated to represent *a tuple updated*, and another 900 numbers represent *not updated*. In other words, in this setting, the update frequency is 0.1. We have run four simulations with initial empirical values 0.1, 0.3, 0.5 and 0.7 respectively. Figures 5.2 shows that the proposed estimator function handles

all these empirical values very well. In all cases the original estimation is adjusted until converges to a stable and desired state.

5.5.2 Efficiency Evaluation

The primary goal of the experiments is to evaluate the efficiency of the proposed delta extraction method. The experiments have been performed on test data based on the example given in Subsection 4.3. We are using 64-bit Oracle Database server version 11.2 that is running on 64-bit Solaris 10 with 4 CPUs and 8G memory. The test data contains three sites and two communication links, where

- the materialized view *V_Remote_Enrollment* with 6 million records is located at the site $\mathcal{DB}_0$.
- the remote data source of *student course* is located at the site $\mathcal{DB}_1$ and the decomposed subquery $Q1$ can be used to query the current data of student course.
- the remote data source of *remote student subject enrollment* is located at the site $\mathcal{DB}_2$ and the decomposed subquery $Q4$ can be used to query the current data of remote subject enrollment.
- the communication link l_1 connects the data warehouse and data source of *student course*, i.e. between $\mathcal{DB}_0$ and $\mathcal{DB}_1$. Similarly, l_2 connects the sites $\mathcal{DB}_0$ and $\mathcal{DB}_2$.

Corresponding to the two decomposed subqueries, two snapshots are generated at site $\mathcal{DB}_0$, represented by $S1$ and $S4$, where each of them has different parameter setting with respect to the delta identification. The conducted experiments cover the processes of generating differential files $\Delta(Q1, S1)$ and $\Delta(Q4, S4)$ by the proposed group hash method.

Parameters	Value
1: No. of Tuples	500,000
2: Size of Tuple	400 Bytes
3: Size of Key	7 Bytes
4: Size of Hash	16 Bytes
5: Tuple Update Probability	5%

Table 5.2: Group Parameters of Q1

5.5.2.1 Snapshot Refresh of S1 from Q1

Firstly, using the Oracle function *dbms_random.value* to randomly change 5% of records in the $Q1$ at the site $\mathcal{DB}_1$. The snapshot $S1$ is projected from *V_Remote_Enrollment* at the site $\mathcal{DB}_0$. The parameters associated with the delta extraction are specified in Table 5.2. Based on the estimation model, the optimized group plan was created. The delta identification between $Q1$ and $S1$ is to execute the group selection plan. The hash function we used is SHA-1 by Oracle *dbms_crypto.hash_sh1*, where the hash value of 128 bits ensures enough distinct hash values are generated to avoid collision.

	Snapshot (MB)	Optimum Group	$p(\equiv)$	C1 (MB)	C2 (MB)	C (MB)	Usage Ratio
Estimation	191	5	0.774	4.86	2.48	7.34	0.038
Execution				4.86	2.48	7.34	0.038

Table 5.3: Delta Identification (Q1 and S1) - Estimation and Execution Results

Table 5.3 demonstrates the results generated by both the cost estimation model and the actual execution of the plan. The unit of measure is Megabyte (abbreviated MB). In the table, the column *Snapshot* represents the size of the snapshot table, i.e. the size of $S1$. The column *Optimum Group* represents the selected optimized group cardinality. By the current setting, the cardinality 5 is chosen as an optimized solution. The column $p(\equiv)$ represents, given group cardinality, its probability of being identical groups. The columns *C1, C2* and *C* represent the costs in the stage 1, stage 2 and the

total costs of delta identification. The column *Usage Ratio* is obtained from $\frac{C}{Snapshot}$. In the table, the first row shows the estimated values based on the cost model and the second row shows the results collected from the plan execution. It was observed that the results collected from the execution support the proposed cost model. Both results are consistent and the slight difference is not shown when rounded to two decimals. The selected group plan is proved to be cost-effective. To complete the task of delta identification of Q1 and S1, when choosing the optimized group cardinality of 5, only around 3.8% of the total size of data is required to transfer from site $\mathcal{DB}_0$ to site $\mathcal{DB}_1$ over the link l_1.

	Snapshot (MB)	C (MB)	Delta (MB)	Refresh Costs (MB)	No.of Comparison
Complete Refresh		N/A	N/A	191	N/A
Data Compression	191	26.26	9.55	35.81	500,000
Group Hash		7.34	9.55	16.89	113,110

Table 5.4: Snapshot Refresh (Q1 and S1) - Comparisons with other Methods

Additionally, we also conducted the comparisons with other traditional methods (See also Section 2.4.3) in terms of *Refresh Costs* and *No. of Comparisons Required*. The other methods we have identified are *complete refresh* and *data compression*. The results of comparison are listed in Table 5.4. As reported by [56], they used *lossy compression* and the compression factors are between 0.15 and 0.10. Here 0.12 is used as the basis of our comparisons. To be consistent, the transmission costs of key attributes are calculated by the original size. Again, the column C indicates the size of data required to complete the task of delta identification. The column *Delta* represents the size of the differential file. In this case, the size of the differential file is 9.5 MB. The column *Refresh Costs* represents total costs of delta identification plus the differential file, which are obtained from $C + Delta$. The last column of the table indicates the number of comparisons is required to complete the task of delta extraction. We can see that both the data compression and the group hash method significantly reduced

communication costs required for delta identification at the expense of some additional data processing. Between the data compression and the group hash method, the table also shows that the latter requires less than 30% of communication costs for the task of delta identification, less than 50% of communication costs for the task of snapshot refresh and also imposes less data comparisons on the source system.

5.5.2.2 Snapshot Refresh of S4 from Q4

Parameters	Value
1: No. of Tuples	6,000,000
2: Size of Tuple	200 Bytes
3: Size of Key	7 Bytes
4: Size of Hash	16 Bytes
5: Tuple Update Probability	10%

Table 5.5: Group Parameters of Q4

As the example given in Section 4.3, Q4 is the source data of *remote student subject enrollment*. S4 is the snapshot of Q4 and is projected from the materialized view *V_Remote_Enrollment*. Comparing to the previous experiment, Q4 contains much more records and is with higher change frequency, etc. The relevant parameters are specified in Table 5.5. The following experiment is conducted with different parameters to further evaluate and validate efficiency of the proposed method.

	Snapshot (MB)	Optimum Group	$p(\equiv)$	Tuple Subs (MB)	C (MB)	Usage Ratio
Estimation	1,144	3	0.729	3.34	109.58	0.096
Execution				3.34	109.59	0.096

Table 5.6: Delta Identification (Q4 and S4) - Estimation and Execution Results

As in the example, to extract the delta of Q4 and S4, the *tuple substitution* is used to resolve the distributed join of attribute *stdcrs_id*. Table 5.6 demonstrates

the results from both the estimation and plan execution. As shown in the example before, Q3 represents the tuple substitution. The column *Tuple Subs* indicates the communication costs incurred by the tuple substitution, which can be acquired by the number of distinct *stdcrs_id*s multiplies its size. The values shown under column C also includes the costs of *tuple substitution*. The table shows that the results of both the estimation and the plan execution are consistent and the communication costs are optimized as expected, i.e. less than 10% of total size of data required.

	Snapshot (MB)	C (MB)	Delta (MB)	Refresh Cost (MB)	No.of Comparison
Complete Refresh		N/A	N/A	1,144	N/A
Data Compression	1,144	177.33	111.40	288.73	6,000,000
Group Hash		109.59	111.40	220.99	3,626,483

Table 5.7: Snapshot Refresh (Q4 and S4) - Comparisons with other Methods

Also, in terms of communication costs required for refreshing the snapshot S4, the comparisons with *complete refresh* and *data compression* are shown in Table 5.7. Comparing to the complete refresh, the group hash method only requires 20% of total data volume, i.e. around 80% communication costs can be saved. Comparing to the data compression, the group hash method is also an ideal option because around 38% of communication costs can be saved and the number of comparisons imposed on remote system can be significantly reduced.

5.5.2.3 View Refresh

Now, we will show the summary of experiment results regarding to the view maintenance of *V_Remote_Enrollment*. Figure 5.3 lists the communication costs and the number of data comparisons that are associated with different options for view refresh, namely, *complete refresh, data compression and group hash method*. Briefly, the view contains 6 million records, the total data volume is $3,433$ MB and consists of two data fragments, i.e Q1 and Q4. Q1 and Q4 contain the data volume of 191 MB and $1,114$

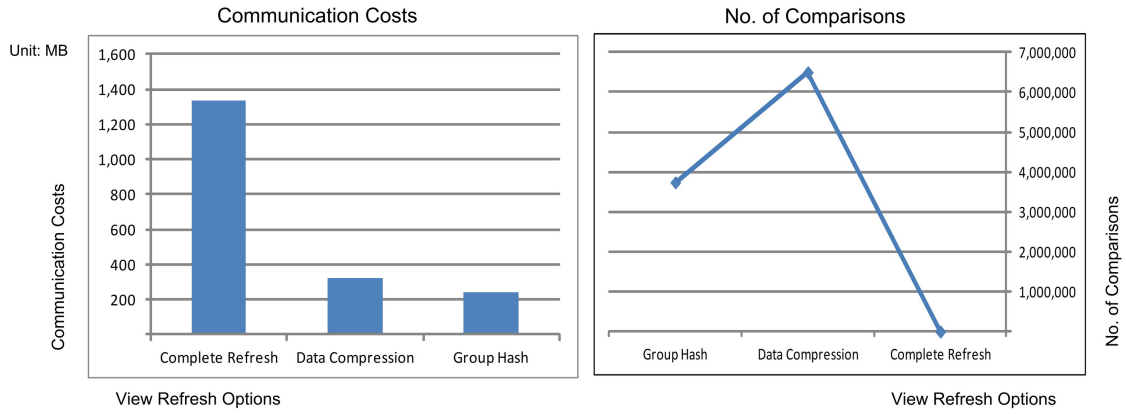

Figure 5.3: View Refresh - Comparisons with other Methods

MB and are located at remote sites $\mathcal{DB}_1$ and $\mathcal{DB}_2$ respectively. The left hand side of the figure shows the sum of communication costs incurred by snapshot refresh of two fragments. The costs of individual fragments have already been listed in Table 5.4 and Table 5.7 respectively. The right hand side of the figure shows the number of comparisons required to complete the view maintenance. Comparing to complete refresh method, at the expense of some additional processing, the group hash method only needs 17.8% of data volume to complete the view maintenance. Comparing to data compression method, the group hash method not only reduced the communication costs but also reduces the number of data comparisons imposed on remote sites to complete the view maintenance.

5.6 Summary

In this chapter, we studied the group strategies for delta extraction in a limited collaborative environment. The goal is to minimize communication costs as much as possible for the task of detecting all data changes in a snapshot. The criterion for design of group hash algorithm is based on the the analysis of expected changes in the snapshot, which motivates, at the data warehouse system, to maintain the statistics about data sources. In addition, we have considered two broad scenarios: one is when the changes are uniformly distributed and the other is to deal with the heterogeneous mixtures of data changes. In both cases, we have developed the cost-driven strategy. In the former, the idea is to find the group plan that requires the minimum volume of data; in the latter, the idea is to find the group plan that yields maximum savings.

By considering the dynamics in the course of data changes, we proposed the estimation function which allows an empirical value is predefined, then gradually adjusted as the new evidence is learnt. We also performed simulations to measure the practicality of the estimation functions. Based on the results of testing four empirical values, it was observed, in all cases, that the results converge to a stable and desired state. More importantly, we have conducted the experiments of delta extraction process. It was observed that the proposed cost models has been proved to be accurate and the outcome of the optimization is satisfactory. We concluded that the proposed group strategy is capable of choosing an ideal group plan.

Chapter 6

Progression Pattern

6.1 Introduction

6.1.1 Motivation

In previous chapter, we developed the group strategies for the delta extraction in a limited collaborative environment, in which the optimization technique relies on the estimation of expected changes in a snapshot. When designing the group hash algorithm, an important assumption was made that the changes at source systems are only caused by random events, therefore the algorithm was not designed to recognize or to use any data pattern when dealing with the evolution of data. However, the real-life data often do not evolve randomly. It has been observed that the changes in nature of the data could be minor fluctuations of the underlying probability distributions, steady trends, random [55], sometimes could be changed systematically or by batches of records [31]. To improve the reliability of the algorithm, these situations during the evolution of data should be taken into account, particularly, when the evolution of data results from routine business logic [89], in which the impact is significant enough to establish a data change pattern.

Using the data change pattern to improve the optimization techniques was based on the following considerations: in the previously proposed group hash method, reliable estimation of expected data changes is the key to an optimal result, where the estimation is based on the assumption of randomly data changes. If a sizable portion

of data in source systems owns the certain features that have a direct effect on the data changes, then the assumption of randomness is no longer valid and it also implies the cost model will not be accurate. This portion of data is referred to as *outlier data*. Obviously, the outlier data should be identified and excluded, so that the remaining data contain randomness. Secondly, if the features of outlier can be found and are associated with a certain degree of temporal regularities, then this combination forms a data change pattern and the formed pattern can help to predict the future data behavior. In this research, the pattern we are interested in has following characteristics

- the pattern with temporal regularities. Informally, the temporal regularity means temporal arrangements of related events repeat with an approximately same interval.
- the pattern shows enough statistical support, i.e. the pattern can be applied to the evolution of large amount of data.

Consider a student enrollment system at university, let us assume its transaction table *t_enrollment* is about student enrollment containing following attributes: student no, student type, course name, calendar year, session, subject name, campus, funding type, enrollment status and grade mark. Also assume the update rate per hundred tuples in the table is 0.1. Undergraduate students enroll subjects for a spring session, and the initial enrollment status is *enrolled*. After 4 weeks, 90% of students choose to stay enrolled while another 10% of students choose to withdraw. When the audit date is reached, unless withdrawn earlier, the status of all active enrollment records are automatically updated to *in progress*, which indicates being liable for the tuition fees. For this category of data, previously proposed group hash algorithm expects 10% update but has to deal with 90% of updated records. Clearly, in this situation, the previous algorithm is unable to produce the optimal group plan, and may even produce the unique worst possible solution. Moreover, for those enrollment records with the status of in progress, after 8-9 weeks, this enrollment will become complete, i.e.

enrollment status is updated to complete. The enrollment progression starts from being enrolled, then in progress, and finally terminated as complete. From the perspective of data change, the attribute of enrollment status is greatly affected by enrollment progression process, while the changes to other attributes could be campus change from regional to main campus or funding type changes from international to domestic. The latter is considered as random data change, while the former is considered as *progression pattern*. The features of this pattern consist of undergraduate, spring session and active enrollment. The effect of the pattern is that 90% of enrollment records will be updated. The temporal associations between the progression features and the effect indicates the pattern behavior is predictable.

6.1.2 Overview

In this chapter, we define a data change pattern, namely, progression pattern that aims to achieve more reliable estimation of data changes at source systems in the situations where the change is resulting from routine business logic and its impact is significant enough to establish a data change pattern. A data change pattern is used to

- describe some underlying processes that have direct effect on the ongoing evolution of data; and
- provide a useful prediction method for the evolution of data.

We believe a progression pattern is a necessary complement to the group hash algorithm in that it helps to relax the assumption that data changes are only caused by random events.

Inspired by abductive reasoning by [77, 66], we also developed a method for discovering a progression pattern. Unlike the classic association rule problem [4], it is to find *all* rules existing in a database with respect to minimal thresholds on certain quality measures. Abductive reasoning is defined to be a process of forming an explanatory hypothesis from an observation requiring explanation. Out of the consideration

of minimizing the overhead of data processing, the discovery of progression pattern is designed to be a process that, in the presence of irregularities observed from the ongoing maintenance processes, generates plausible explanations why they are present, and testing the plausible explanations.

6.2 Basics of Progression Pattern

6.2.1 Data Model

To enable the progression pattern discovery, at data warehouse system, the timestamps need to be added into the data model. Data synchronization is an ongoing process and consists of a sequence of operations. Correspondingly, the timestamp is an ordered collection of occurrence times $\tau = \tau_1, \tau_2, ..., \tau_n$, which indicates database system clock time at which information is posted into the database, i.e. transaction time. We assume that data synchronization is carried out at a regular interval and each one is associated with a timestamp. In this data model, firstly, let R denote a base relation of the standard relational data model. A transaction-time relation T_R over a base relation R is referred to as t-relation, which is maintained at data warehouse system and aims to keep a complete change history. Also note that the value of key attribute is tuple unique in a base relation R, but not in a t-relation T_R. The composite of (τ, k) is tuple-unique in a t-relation. A base relation is interpreted as containing data that are valid only at a given time point, which can be retrieved by a time function as follows:

$$R(\tau_i) \stackrel{def}{=} T_R \ \ at\ time\ \tau_i \tag{6.1}$$

$$R \stackrel{def}{=} T_R \ \ at\ time\ Now \tag{6.2}$$

We also define two types of time-scales that are used to specify the temporal aspects of different events

1. elapsed time, the expression $[\tau_s, \tau_e]$ specifies the length of time since the event first occurred at time point τ_s, and continuously without interruption at every point until the time point τ_e;

2. time-at, the expression τ_i indicates the time point at which the event occurs.

6.2.2 Progression Pattern

Progression pattern appears as repeated occurrences of data changing event, which consists of two correlated sub-events, namely, *progressing event* and *update event*. A progressing event $\mathcal{P}$ is specified by progression variables (also abbreviated p-variables) and the associated elapsed time. P-variables under selected attributes $A_i, ..., A_m$ with domain $D_{A_i}, ..., D_{A_m}$ are a subset of $D_{A_i}, ..., D_{A_m}$, represented by $x_i, ..., x_m$. The element of p-variables contains two types of information, the attribute and the value of the attribute. P-variables are expressed as conjunctions in which every element is restricted to be a ground formulae. Using the example of the University enrollment system, the selected p-variables read as follows

$$student\_type = undergrad \wedge session = spring \wedge status = enrolled$$

A progressing event $\mathcal{P}$ on p-variables X for an elapse time $[\tau_s, \tau_e]$ is denoted by $\mathcal{P}(X_{[\tau_s,\tau_e]})$. The elapse time over p-variables specify that, to be a valid progressing event, the p-variables must hold true for the entire length of period $[\tau_s, \tau_e]$. To handle certain variations from the typical temporal association, our definition of progression pattern includes the concept of a time tolerance, denoted δ_τ, which specifies an allowed level of temporal discrepancy and data latency, i.e. $[\tau_s, \tau_e] \pm \delta_\tau$. The idea is that progressing event is valid, even if it is varied within a specified limit of the expected duration.

An update event $\mathcal{U}(E_{\tau_u})$ is to specify that update of attributes E happened at a given time point τ_u. In this research, we are only interested in what attributes are about to be updated, not what the updated values would be. As the correlated sub-events, $\mathcal{P}$ and $\mathcal{U}$ are bound by both common keys and a temporal association across their occurrences. Firstly, let $\mathcal{U}\kappa$ be all keys in the dataset related to $\mathcal{U}(E_{\tau_u})$, $\mathcal{P}\kappa$ be all keys in the dataset related to $\mathcal{P}(X_{[\tau_s,\tau_e]})$, then $\mathcal{U}\kappa \subseteq \mathcal{P}\kappa$. Secondly, by using the interval representation of time developed in [6], the temporal association of $\mathcal{P}$ and $\mathcal{U}$ is described by the relation *meet* as follows

- $\mathcal{P}$ always precedes $\mathcal{U}$;
- $\mathcal{P}$ and $\mathcal{U}$ do not overlap, that is given the elapse time of $\mathcal{P}$ is $[\tau_s, \tau_e]$, then τ_u of $\mathcal{U}$ is τ_{e+1}.

Also note that the elapse time of $\mathcal{P}$ does not represent the entire period the p-variables hold true, but terminates immediately before $\mathcal{U}$ occurs, which means the past history and future possible values of p-variables are not important, i.e. before τ_s or after τ_e are not important.

6.2.3 Progression Rule

A progression pattern is to describe the the evolution of data and specify how these data evolve over time, which is expressed by a progression rule in terms of antecedent, consequent and control parameters as follows:

$$\mathcal{P}(X_{[\tau_s,\tau_e]}) \longrightarrow \mathcal{U}(E_{\tau_u})[\delta_\tau, min\_conf]$$

A rule antecedent is a progressing event representing a progression condition. The p-variables X are chosen because if the p-variables hold true for a specified elapse time, then they suffice to describe the progression condition. A rule consequent is the update event specifying the progression effects, which means if the rule's antecedent is supported, attribute E is updated at the time point τ_u. Progression patterns are not always perfect, thus two control parameters $\mathcal{C} = \langle \delta_\tau, min\_conf \rangle$ are specified to ensure the quality of a discovered pattern, where δ_τ is the time tolerance and *min_conf* is the threshold of minimum of confidence a progression rule needs to pass. The *confidence* indicates the strength of the rule and is understood as the conditional probability of the consequent occurring given the antecedent. A tuple t_i is said to contain X if X is a subset of t_i. A function *ct-k* over a t-relation is defined as $ct\_k(X_{min\_p})$, which is to count distinct keys in a t-relation that satisfies the conditions: containing X and X hold true for the minimum of period min_p. The function $ct\_k(X_{min\_p} \cup E_{\tau_u})$ is also

a count function. It also requires that the elapse time of progressing event meets the time point the update event occurs. The minimum of the length of period min_p is derived from $[\tau_s, \tau_e] \pm \delta_\tau$. Also notice that both $ct\_k(X_{min\_p})$ and $ct\_k(X_{min\_p} \cup E_{\tau_u})$ are to count distinct keys in a set of tuples, rather than to count the number of tuples. The rule confidence and min_conf are used in a progression rule as follows

$$min\_conf \leq \frac{ct\_k(X_{min\_p} \cup E_{\tau_u})}{ct\_k(X_{min\_p})} \tag{6.3}$$

Progression rules are used for the prediction of update event, thus the rules with a high confidence are often the most interesting and useful ones.

6.3 Pattern Discovery

The progression rule is interpreted as a strong explanatory co-occurrence relationship between antecedent and consequent.

6.3.1 Problem Environment

The problem of pattern discovery is a tuple

$$\langle E, X_{all}, e, \mathcal{C}, f_p, E^{\top} \rangle$$

- E is an evidence attribute at which the update event shows some irregularity with respects to overall observations.
- X_{all} is a finite, non empty set of possible atomic hypotheses.
- e is a relation of $2^{X_{all}}$ and E, i.e. $e \subseteq 2^{X_{all}}$. The relation e connects the observed evidence attribute with set of hypotheses which explain it.
- $\mathcal{C}$ holds the quality criteria of evaluation.
- f_p is a discovery function. The function f_p assesses e by referencing the elements in $\mathcal{C}$ and mapping from $2^{X_{all}}$ to a particular subset.
- $E^{\top}$ indicates the irregularity to be present.

An exhaustive search for the mining progression rule is to compute the confidence and elapse time for the power set of all atomic hypotheses. This approach is prohibitively expensive because exponential growth of possible rules to be extracted when the number of variables increases. To cope with this computationally expensive task, one of the inference principles guiding the discovery process is that the observed irregularities of update event are believed to be the consequent of the progression rule, i.e. the irregularities can be explained, unless the contrary is proved in the succeeding process.

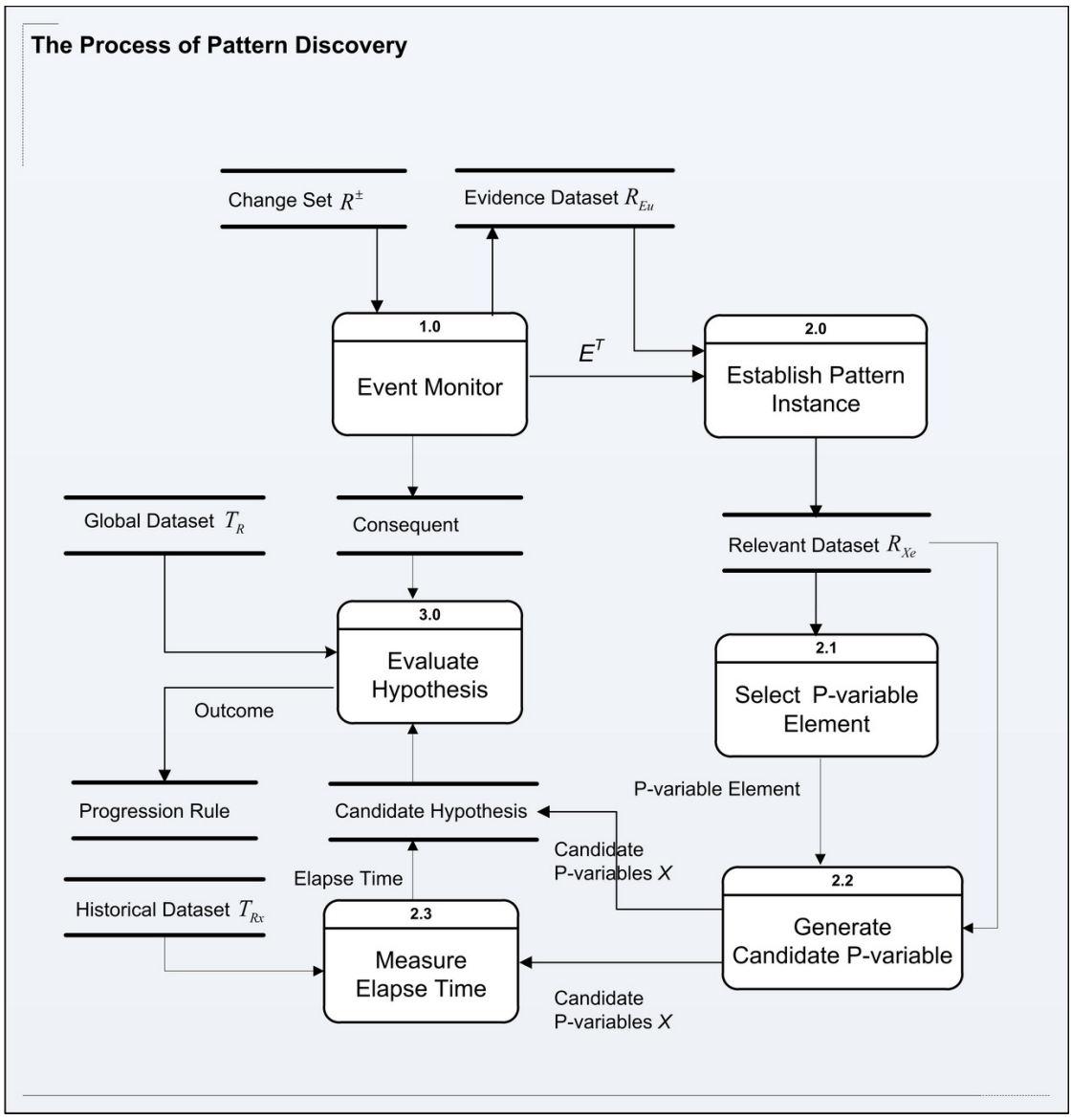

Figure 6.1: The Process of Pattern Discovery

Inspired by abductive reasoning [77, 66], we have developed the heuristic techniques for finding the candidate rules, as shown in Figure 6.3.1, which consists of three steps:

6.3.2 Event Monitoring

Event monitoring is to identify the update event with irregularity. An update event will be identified if it introduces irregularities from established normal habits. The event monitoring is conducted in a change set when a synchronization process is complete. A *change set*, denoted by $R^{\pm}$, is a subset of differential file $\Delta(R_i, R_i')$, instead of containing

all information about insert, delete and update, a change set only contains *all updates* (See also Section 5.2.3).

The main assumption underlying pattern discovery approach is that a unique irregularity is present, which means every time the observed irregularity is associated with only one progression rule or possibly none. The technique of event monitoring is to use *the normal approximation to the binomial distribution of frequency domain*. Let $\mid R \mid$ represent the cardinality of base relation, let P_{A_i} be the update probability of attributes A_i, and $|A_{iu}|$ be the number of actual updated tuples of attribute A_i, then the mean of the number of the updates of attribute A_i is

$$\mu = \mid R \mid \times P_{A_i} \tag{6.4}$$

using normal approximation to the binomial distribution, the standard deviation is obtained from

$$\sigma_\mu = \sqrt{\mid R \mid \times P_{A_i} \times (1 - P_{A_i})} \tag{6.5}$$

As a general rule, whenever both $\mid R \mid \times P_{A_i}$ and $\mid R \mid \times (1 - P_{A_i})$ exceed 5, the binomial distribution approximately follows the normal distribution. The irregularity is identified by calculating the z-score, where a z-score of 0 indicates the score is the same as the mean and a positive z-score indicates a datum above the mean in the units of the standard variation.

$$z = \frac{|A_{iu}| - \mu}{\sigma_\mu} \tag{6.6}$$

Substituting into the transformation formula

$$z = \frac{|A_{iu}| - \mu}{\sqrt{\mid R \mid \times P_{A_i} \times (1 - P_{A_i})}} \tag{6.7}$$

We use an example to explain how to use z-score to decide whether a change set is believed containing the update event with irregularity. Suppose that the base relation R contains 1000 tuples in which 315 tuples found being updated under attribute A_i, and let $P_{A_i} = 7\%$. Since both $\mid R \mid \times P_{A_i} = 1000 \times 0.07 = 70$ and $\mid R \mid \times (1 - P_{A_i}) =$

$1000 \times 0.93 = 930$ exceed 5, the the normal distribution can be used to approximate the binomial as follows

$$z = \frac{315 - 70}{\sqrt{1000 \times 0.07 \times (0.93)}} = \frac{245}{65.1} = +3.76 \tag{6.8}$$

By referring to the standard normal distribution table, the probability of $z \geq 3.76$ is $1 - 0.99992 = 0.00008$. If a significant level is set to 0.005, then the irregularity is identified. $E^{\top}$ indicates the irregularity is present. When multiple attributes are found associated with irregularities, the attribute with the maximum of $|A_{iu}|$ is designated as the evidence attribute E.

6.3.3 Candidate Hypothesis Generation

Dataset	Explanation
1: $R(\tau_e)$	R at the time point τ_e
1: $R^{\pm}$	A change set
2: R_{E_u}	All tuples with evidence attribute E updated at the time τ_u
3: R_{X_e}	The relevant dataset of R_{E_u} at the time point τ_e
4: T_{R_x}	Historical dataset of R_{X_e}

Table 6.1: Symbol and Explanation

The update event with irregularity represents the actionable state that triggers the succeeding action of explanatory analysis, i.e. triggering the action to find all candidate hypothesis that might explain the irregularity. The hypothesis is in form of rule's antecedent consisting of p-variables and elapse time. The candidate hypothesis needs to be the full explanation of the observation, where the hypothesis is the full explanation if its p-variables has no super-set that is also the candidate hypothesis. In this case, the p-variables are referred to as *maximal p-variables*.

We start by introducing the concept of *min_relevance*. The relational data model itself does not offer to represent prior knowledge regarding to the current observations, therefore *min_relevance* is defined in terms of uncertainty measures, aiming to establish

the evidential link between the observed irregularities to the relevant dataset of prior state. Provided that $E^{\top}$ is found to be present at time point τ_u, R_{E_u} is a subset of a change set $R^{\pm}$, which only contains the tuples with the evidence attribute E updated. The relevant dataset of prior state of R_{E_u} is defined as

$$R_{X_e} \leftarrow \sigma(R(\tau_e) \bowtie_{key=key} \Pi_{key}(R_{E_u})) \tag{6.9}$$

Essentially, the dataset R_{E_u} contains the observed irregularities, while the dataset R_{X_e} contains the relevant knowledge of prior state regarding to R_{E_u}. *min_relevance* is to establish the evidential link between R_{E_u} and R_{X_e}. The union of two datasets i.e. $R_{E_u} \cup R_{X_e}$ is referred to as *pattern instance*.

- According to the definition of progression rule, $\tau_e = \tau_u - 1$ and conceptually, *min_relevance* is understood as

$$min\_relevance = \frac{ct\_k(X_{\tau_e} \cup E_{\tau_u})}{|E_{\tau_u}|} \tag{6.10}$$

  where $|E_{\tau_u}|$ is the number of updated tuples associated with evidence attribute E at the time τ_u. Given $|E_{\tau_u}|$, *min_relevance* indicates the base rate of the p-variables of interest X to be expected at the time point τ_e with respect to an evidence attribute E at the time point τ_u.

- Based on the concept of *min_relevance*, the minimum number of tuples expected to hold the p-variables of interest in the dataset R_{X_e} is

$$min\_thresh = |E_{\tau_u}| \times min\_relevance \tag{6.11}$$

- Based on the update probability of evidence attribute E, denoted by P_E, the *min_thresh* is estimated as follows:

$$min\_thresh \approx |E_{\tau_u}| - |\,R\,| \times P_E \tag{6.12}$$

  Basically, it means that after eliminating the randomly updated tuples from R_{E_u}, the rest is considered as the direct consequent of progression rule occurring at

the point τ_u. *min_thresh* is an approximation that allows to produce a heuristic and computationally efficient solution to the problem of candidate hypotheses generation. The value of *min_thresh* can be adjusted, i.e. increase or decrease, where the adjustment indicates the degree of willingness to risk fitting noise.

Enrol_Id	S#	Name	Type	Subject	Session	Status	Funding
Enr_101	901	James	Undergrad	Math101	Spring	Complete∗	Int.
Enr_102	901	James	Undergrad	CSCI204	Spring	Complete∗	Int.
Enr_103	902	Alice	Undergrad	ACCY100	Spring	Complete∗	Dom.
Enr_104	903	John	Postgrad	THES924	Annual	Enrolled	Spnsr.∗
Enr_105	902	Alice	Undergrad	ACCY200	Spring	Withdrawn∗	Dom.
Enr_106	904	Jason	Undergrad	CSCI315	Spring	Fail∗	Dom.
Enr_107	906	David	Undergrad	CSCI401*	Anual	Incomplete	Dom.

Table 6.2: $R^{\pm}$: A Change Set of Student Subject Enrollment

Enrol_Id	**S#**	**Name**	**Type**	**Subject**	**Session**	**Status**	**Funding**
Enr_101	901	James	Undergrad	Math101	Spring	Enrolled	Int.
Enr_102	901	James	Undergrad	CSCI204	Spring	Enrolled	Int.
Enr_103	902	Alice	Undergrad	ACCY100	Spring	Enrolled	Dom.
Enr_105	902	Alice	Undergrad	ACCY200	Spring	Enrolled	Dom.
Enr_106	904	Jason	Undergrad	CSCI315	Spring	Enrolled	Dom.

Table 6.3: R_{X_e}: Relevant Dataset of Evidence Attribute at Prior State

Table 6.2 shows an example of change set captured at the time point τ_u, in which *Enrol_Id* is the key identifier and the symbol $*$ shown in the table is to indicate the updated values. In the table, in the case that the attribute *Status* is chosen to be an evidence attribute, then the dataset R_{E_u} contains *Enr_101*, *Enr_102*, *Enr_103*, *Enr_105* *Enr_106*. Given the evidence attribute being chosen, Table 6.3 shows the example of relevant dataset R_{X_e} at the time point τ_e. The searching for p-variables is conducted in a pattern instance, i.e. $R_{E_u} \cup R_{X_e}$, while to ensure comprehensibility and accuracy, the evaluation of candidate hypothesis at a later stage is conducted in the full range

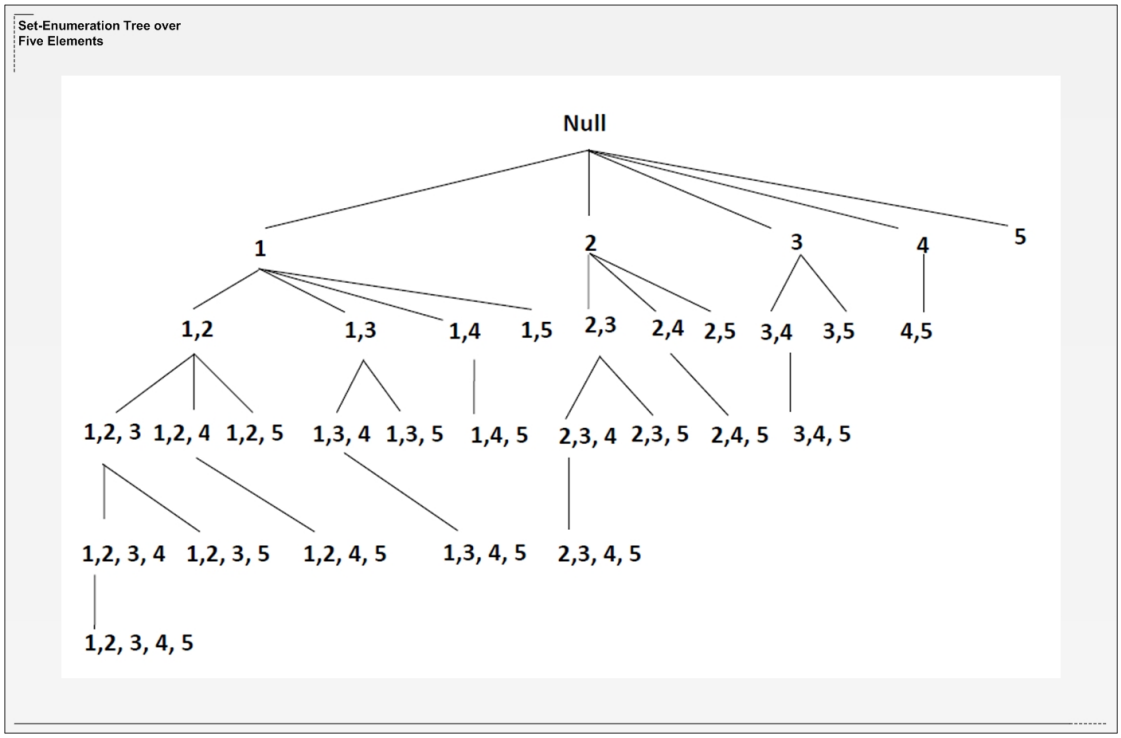

Figure 6.2: Complete set-enumeration tree over five elements

of data. In the absence of any information to the contrary, R_{E_u} is the dataset that is believed to contain the effect of the rule, which allows us to make inferences upon partial information by establishing the evidential link to its relevant dataset of prior state, and to retract of the previous conclusions if the contradictory information appears during the evaluation of candidate rules. More importantly, the method of inference helps to effectively reduce the search space when searching for candidate hypothesis.

P-variable Element Selection. The hypothesis generation starts from the p-variable element selection. In the dataset R_{X_e}, only if an attribute consists of at least *min_thresh* of identical values, then the value is selected to be an element of p-variable. The element selection aims to eliminate the irrelevant attributes and variables to be considered at early stage and consequently it will help to reduce the candidates to be examined. Non-existence of element leads to the termination of the discovery process. According to equation (6.12), as shown in Table 6.3, if the estimated *min_thresh* is 4, then the qualified attributes would consist of *Type*, *Session* and *Status*.

Candidate P-variables Generation. After the elements are selected, the next step is to verify the truth value of the conjunctions of these selected elements that are contained in dataset R_{X_e}. To be a qualified candidate p-variable, the conjunction of these elements also needs to pass the test of *min_thresh*. In addition, the candidate p-variables need to be maximal (See page 81 for more details). This task uses the data structure of generic set enumeration tree (abbreviated se-tree), which enumerates elements of a power set using a pre-imposed order on the underlying set of elements. Figure 6.2 illustrates a se-tree for the complete power set of {1,2,3,4,5}, where each subset of elements is called node and the arcs between nodes are called branches. The root node of a se-tree is always an empty set. A leaf is a node without branches underneath it. Every single element of a se-tree is a parent node. A tree path is a subset of a se-tree, consisting of an array of nodes from a leaf to its parent node. Without pruning, a se-tree search for candidate hypothesis needs to explore every node over the element power set. To avoid highly redundant exploration, the binary search strategy has been adopted to divide and conquer every tree path. By taking the property of maximal p-variables into account, the search algorithm uses bi-directional pruning rules for reusing and pruning nodes. Given the *min_thresh*, the bi-directional pruning rules consist of

- upward pruning, if a probed node meets the threshold, then all the nodes upward to its parent node also meet the threshold.
- downward pruning, if a probed node fails to meet the threshold, then all the nodes downward to the leaf also cannot meet the threshold.

The pseudo-code description of p-variables generation is presented in Algorithm 4. Given a set of p-variable elements, the algorithm employs a depth-first approach to searching a se-tree. The binary search is conducted at every tree path, i.e. *tree_path(i)*, such that every iteration eliminates half of the remaining nodes. Algorithm 5 is the

Algorithm 4 Generate P-Variables

```
    procedure GEN_P-VARIABLES(tree_path)
 2:     k ← 0
        p_var ← ∅
 4:     max_list ← ∅
        min_list ← ∅
 6:     the_path ← ∅
        for i ← 1 .. tree_path.last do
 8:         the_path ← tree_path(i)
            if max_list ≠ ∅ or min_list ≠ ∅ then
10:             the_path ← Trim_Tree_Path(the_path, max_list, min_list)
            end if
12:         p_var ← Search_Tree_Path (the_path, min_thresh, start_idx, end_idx)
            k ← k + 1
14:         max_list(k) ← p_var
            update_min_list
16:     end for
        return max_list
18: end procedure
```

pseudo-code description of the tree path search algorithm we use to process an individual tree path and the algorithm returns the qualified p-variables of a tree path.

In addition, the basic algorithm is augmented with two data structures, namely, *max_list* and *min_list*. The *max_list* keeps all the maximal p-variables that pass the support count test of *min_thresh* and are used for upward pruning. For example, after processing the left most tree path, let the node $\{1, 2, 3\}$ be the maximal p-variables adding into max_list, for its succeeding tree path, i.e. the second left most tree path, any node that is the subset of $\{1, 2, 3\}$ is exempt from the redundant testing, and consequently the only node needs to be tested is $\{1, 2, 3, 5\}$. On the other hand, the *min_list* keeps the minimal p-variables that fail to pass the support count test of *min_thresh* and are used for downward pruning. For example, after processing the left most tree path, let the node $\{1, 2, 3\}$ be the minimal p-variables adding into min_list, for its succeeding tree paths, any node that is the super set of $\{1, 2, 3\}$ is exempt from the further testing, and consequently processing of tree path is complete without

Algorithm 5 Search Tree Path

```
    procedure SEARCH_TREE_PATH(the_path, min_thresh, start_idx, end_idx)
 2:     if start_idx = end_idx then
            if ct_k(the_path(start_idx)) ≥ min_thresh then
 4:             return the_path(start_idx)
            else
 6:             return the_path(start_idx − 1)
            end if
 8:     else
            m_idx ← midpoint(start_idx, end_idx)
10:         if ct_k(the_path(mid_idx)) ≥ min_thresh then
                return Search_Tree_Path(the_path, min_thresh, start_idx, m_idx − 1)
12:         else if ct_k(the_path(mid_idx)) < min_thresh then
                return Search_Tree_Path(the_path, min_thresh, m_idx + 1, end_idx)
14:         end if
        end if
16: end procedure
```

Algorithm 6 Trim Tree Path

```
    procedure TRIM_TREE_PATH(the_path, max_list, min_list)
 2:     for i ← 1 .. the_path.last do                    ▷ all nodes in a tree path
            for k in 1 .. min_list.last do                ▷ minimal subset failed
 4:             if the_path(i) ⊇ min_list(k) then
                    while i ≤ the_path.last do
 6:                     the_path(i).delete               ▷ delete all nodes downwards
                        i ← i + 1
 8:                 end while
                end if
10:         end for
        end for
12:     for i ← the_path.last .. 1 do
            for k in 1 .. max_list.last  do               ▷ maximal subset failed
14:             if the_path(i) ⊆ max_list(k) then
                    while i ≥ 1 do
16:                     the_path(i).delete               ▷ delete all nodes upwards
                        i ← i − 1
18:                 end while
                end if
20:         end for
        end for
22:     return the_path
    end procedure
```

accessing the dataset at all. The pseudo-code description of using max_list and min_list to trim a tree path is presented in algorithm 6. Finally, the max_list holds a set of all the candidate p-variables passing the test of *min_thresh*.

Measure of Elapse Time After candidate p-variables are selected, the elapse time of the candidate p-variables needs to be measured to further confirm their validity. Firstly, the dataset we use to find the elapse time of candidate p-variables is as follows

$$T_{R_X} \leftarrow \sigma(T_R \bowtie_{key=key} R_{X_e}) \tag{6.13}$$

The dataset T_{R_X} is the type of t-relation containing the historical data of R_{X_e}. Basically, the elapse time of the selected candidate p-variables X is the mean of a set of individual elapse times extracted from T_{R_X}. An individual elapse time is associated with a distinct key. Since τ_e was already known, i.e one time unit before τ_u, to find an individual elapse time associated with k_i, it only needs to find τ_{s_i}, that is the first time both k_i and X appear in T_{R_X}. Given candidate p-variables X, suppose n individual elapse times are extracted from the dataset, then the elapse time of X is obtained from

$$elapse\_time = \frac{1}{n}\sum_{i=1}^{n}(\tau_e - \tau_{s_i}) \tag{6.14}$$

The mean obtained above gives us the information about the central value of a discrete set of elapse times from the history. The absolute deviation function together with time tolerance are used to assess the degree of dispersion of individual elapse times. The function outputs a non-negative real number that is zero if the individual elapse times are the same and increases as they become more diverse. Also, the time tolerance δ_τ sets the limit of the degree of dispersion as follows

$$\delta_\tau \geq \frac{1}{n}\sum_{i=1}^{n}|elapse\_time - [\tau_s, \tau_{s_i}]| \tag{6.15}$$

Only passing the testing on the degree of dispersion, the p-variables and its associated elapse time are considered to be a acceptable candidate hypothesis. In addition, the degree of dispersion can be used to prioritize the candidate hypotheses for the evaluation process in next step. Low degree of dispersion is more ideal as it stands for uniformity.

6.3.4 Hypothesis Evaluation

To establish a candidate hypothesis into progression rule, we must evaluate its strength. The evaluation is carried out in the t-relation T_R and it is the only step that needs to access full range of data. A evaluation function follows the minimum confidence equation (6.3), which is to assign a degree of confidence to hypothetical conclusion $X_{[\tau_s,\tau_e]}$ on the basis of evidence E with a confidence between 0 and 1. To be accepted as a progression rule, the confidence must be over the predefined threshold min_conf. This evaluation function generalizes the proposed progression rule from classical notion of logical entailment, i.e. non-fallacious arguments, to the partial entailment inference rule with quantified probabilities.

6.4 Using Progression Pattern

Incorporating the discovered progression pattern into the process of data synchronization is straightforward. Given a progression rule, an event-based monitor is used to capture the first occurrence in transactions, which contains p-variables and a distinct key, then registering the transactions to the progression rule, i.e. logging progression start time as now and estimating progression end time. A time-based monitor is used to read from the registered transactions, when the minimum elapse time is reached, it notifies the next data synchronization of update event. The confidence level of the progression rule will override the update probability collected from general statistics and will be used as the update probability to participate in the grouping selection.

6.5 Experiments

6.5.1 Negative Impacts Caused by Outlier

Parameters	Value
1: No. of Total Tuples	6,000,000
2: Length of Tuple	200 Bytes
3: Length of Tuple's Key	7 Bytes
4: Length of Hash	16 Bytes
5: Tuple's Update Frequency	5%

Table 6.4: Group Selection Parameters

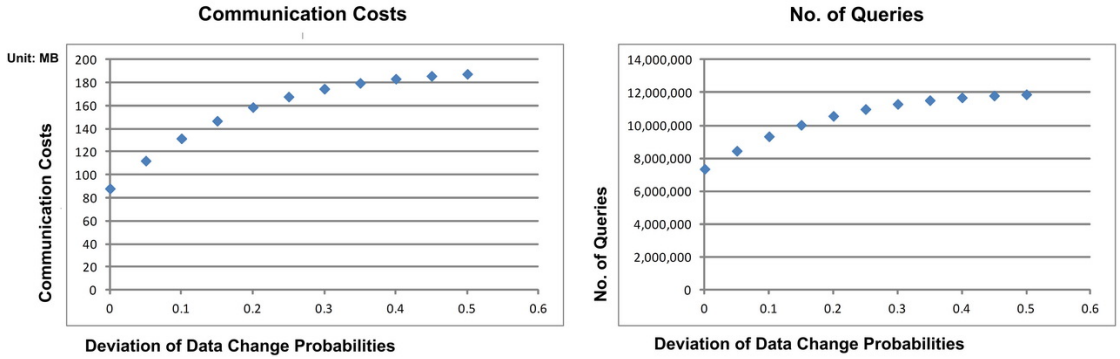

Figure 6.3: Negative Impacts Caused by Outlier

Previously, we have introduced the concept of *outlier* and we also briefly discussed the negative impact of outlier in the motivation example. To give the proof and to show how the negative impacts can be caused by outlier data when using the group hash algorithm, we have conducted the simulations and analysis. Table 6.4 lists the parameters. In Figure 6.3, the graph to the left represents the negative impacts caused by outlier data with respect to communication costs, in which the x-axis represents the deviation from the expected data change probabilities and the y-axis represents the communication costs. The size of snapshot is $1,144$ MB. In the graph, by using group hash algorithm, the minimized communication costs are the one with the deviation of zero, more specifically, 88.13 MB of data volume is required when choosing

the optimized group cardinality of 5. The value of 0.3 shown on x-axis represents that the deviation of 0.3 from the expected data change probability, i.e. 35% of data are changed. Because of the deviation, the communication costs incurred have been doubled and are more than expected, i.e. 174.70 MB. The graph to the right shows the negative impacts caused by outlier data with respect to total queries required to impose on a source relation. Figure 6.3 shows that the outlier that is not handled causes the negative impacts on both the communication costs and data processing.

6.5.2 Efficiency Evaluation

The experiment is designed to evaluate the efficiency of the optimization techniques for candidate p-variables generation. The task of candidate p-variables generation is to verify the truth value of the conjunctions of the selected p-variable elements that are contained in dataset R_{X_e} (See also subsection 6.3.3). The efficiency is measured in terms of *data processing* and *run time*. The experiments conduct on the different sizes of synthetic datasets R_{X_e} and more specifically, it includes 5 datasets with size of 20,000, 50,000, 100,000, 150,000 and 200,000. Each of these 5 datasets is described by 7 attributes and consists of a mixture of random values and p-variables, in which true p-variables are known in advance. A 7-tuple dataset is a relation of 2^7, which consists of all attribute combinations and the number of these combinations is obtained from

$$\sum_{k=1}^{7} \binom{7}{k} = 2^7 \tag{6.16}$$

In a dataset, each attribute combination in 2^7 is associated with a set of value combinations. For example, the attribute combination of A1, A2, A3, in a dataset, may be associated with multiple value combinations *abc, adc and adb,...*, etc. To find a complete set of candidate p-variables, all of these value combinations need to be tested. The number of value combinations in a dataset is denoted by $\mathcal{N}$.

In this experiment, all the synthetic datasets of R_{X_e} are generated in a controlled manner. Table 6.5 describes the distinct values per attribute. For example, the table

A1	A2	A3	A4	A5	A6	A7
3	10	3	20	3	40	3

Table 6.5: Distinct Values Per Attribute

shows that attribute 1, i.e. A1, contains 3 distinct values and attribute 6, i.e. A6, contains 40 different values. This setting has consistently applied to all 5 datasets. Table 6.6 shows the value distributions in all 5 different datasets and also specifies the value of *min_thresh* for each dataset. The value of *min_thresh* sets the criteria that a value combination need to pass to be accepted as a candidate p-variables. In the table, for the dataset with 20,000 rows, A1 has the most distinct value of 19,500 and all other values of A1 are randomly selected by using the Oracle function *dbms_random.random*. By referring to Table 6.5, we know that all other values of A1 are randomly chosen from 3 different values. Finally, the column *M.Thresh* specifies the value of *min_thresh*.

Data Size	A1	A2	A3	A4	A5	A6	A7	M.Thresh
20,000	19,500	18,000	19,500	18,000	18,000	18,000	19,500	19,000
50,000	49,000	47,000	49,000	47,000	47,000	47,000	49,000	47,500
100,000	99,000	94,000	99,000	94,000	94,000	94,000	99,000	95,000
150,000	149,000	140,000	149,000	140,000	140,000	140,000	149,000	142,500
200,000	199,000	185,000	199,000	185,000	185,000	185,000	199,000	190,000

Table 6.6: Value Distribution of Each Attribute

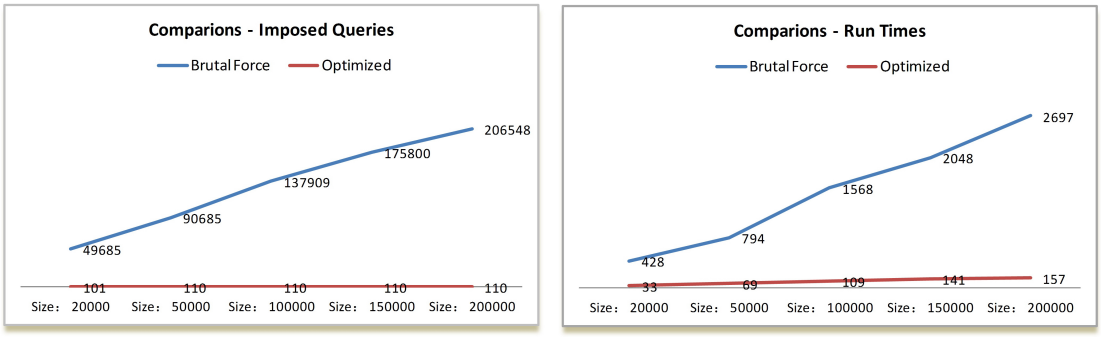

Figure 6.4: Performance Comparisons

Our focus is to evaluate the performance when generating candidate p-variables by comparing the exhaustive search and the optimized algorithm. The optimized techniques include *upward pruning* and *downward pruning*. The upward pruning is implemented by using the max_list, while downward pruning is implemented by using the min_list. Figure 6.4 shows the performance comparisons between the exhaustive search and optimized algorithm for the task of candidate p-variables generation. Generally speaking, comparing to min_list and max_list, the dataset itself is much larger. Instead of testing all value combinations in a dataset, the efficiency can be achieved by reducing the queries imposed on the dataset and querying the min_list and max_list whenever it is possible. According to the results collected from all 5 different executions, the min_list contains 79 to 81 records and max_list contain 3 to 5 records only. Clearly, both of them are much smaller that the dataset.

The graph to the left shows the comparisons in terms of the number of queries imposed on the dataset. When using the exhaustive search on the dataset range from 20,000 to 200,000, the number of queries were imposed on the datasets are 46,985, 90,685, 137,909, 175,800 and 206,548 respectively and these numbers also indicate the number of all value combinations, i.e. $\mathcal{N}$, exist in the dataset. On the other hand, from the graph we can see that using optimized algorithm to search on these datasets, the number of queries imposed on the dataset has been minimized to 101 - 110, which means the vast majority of tests are resolved by either min_list or max_list.

The graph to the right shows the comparisons in terms of CPU time (in seconds) of the runs. We can see that the optimized algorithm are 12 - 17 times faster than the exhaustive search. This is not surprising since the number of queries imposed on the dataset has been minimized and two much smaller data structures are used to resolve most of quires, thus the optimized algorithm can effectively reduces the run time.

6.6 Summary

To handle the outlier and relax the assumption that the data changes at source systems are only caused by random events, we have defined the progression pattern, which can be used to describe data change with temporal regularities. Inspired by abductive reasoning, the discovery of progression pattern is designed to be a process of forming an explanatory hypothesis from an observed irregularity of update event requiring explanation. The discovery process consists of three steps: event monitoring, hypothesis generation and hypothesis evaluation, in which the important inference principle is that the observed irregularities of update event are believed to be the consequent of the pattern unless the contrary is proved in the succeeding process. When generating candidate hypotheses, this principle helps to

- eliminate the irrelevant attributes at early stage.
- reduce the search space. To generate the candidate hypotheses, instead of searching against an entire historical dataset, the search space is limited to the pattern instance, which is only the small portion of the entire historical dataset.

In addition, to avoid highly redundant exploration, the upward and downward pruning rules have been integrated into the binary search algorithm, such that the search path can be divided then conquered.

The simulation has been designed to show the negative impacts caused by outlier data, which helps to understand the associations between the deviation from expected data change and the extra communication costs incurred. The results of the simulation support the claims that capturing the data change pattern is the necessary complement to the previously proposed group hash algorithm.

We have conducted the experiments to evaluate the efficiency of the optimized algorithm for the task of candidate P-variables generation. The experiment results have shown that the optimized algorithm can greatly reduce the CPU time of the runs

in each experiment, and more specifically, comparing the to the exhaustive search, the optimized algorithm is 12 - 17 times faster. The efficiency is achieved by reducing the queries imposed on the dataset and querying the much smaller data structure min list and max list whenever it is possible. One final comment is of interest here. The more candidate p-variables exist in a dataset, the more queries can be resolved by the max_list. On the other hand, the less candidate p-variables exist in a dataset, the more queries will be resolved by the min_list.

Chapter 7

Nested Group Hash Method

7.1 Motivation

In this research, the optimization goal is to reduce the volume of data transfer for the task of delta extraction at the expense of additional data processing. In Chapter 5, we have introduced a 2-staged group hash algorithm. A 2-staged group hash algorithm can greatly reduces the communication costs, but does not guarantee the communication costs being minimized. When using a 2-staged group hash algorithm, all the changed groups identified at stage 2 are resolved by conducting the individual comparisons. The opportunities for improvement could be that rather than conducting the individual comparisons too early, all the identified changed groups are broken down into smaller subsets for additional group testing, hereby referred to as *nest group hash method*. Essentially, this method is characterized by the condition that all elements to be examined in the next stage must be chosen from the previously identified changed groups. For practical purposes, we restrict that the break-down of a changed group happens only once and when breaking down a changed group, the changed group is always partitioned into two smaller subsets.

In this chapter, the nested group hash method for the task delta extraction is aiming to seek the possibility of continual improvement in terms of efficiency and reliability, where the efficiency is measured by volume of data transfer required to complete the task, and reliability means how stable the algorithm behaves in different settings. The

design approach of the nested group hash algorithm is fundamentally different from the approach used for 2-staged group hash algorithm in that firstly, the nest group hash algorithm is a multi-staged algorithm and secondly, the dependencies across the different stages need to taken into account since, the later tests according to the result of previous tests. Therefore, the cost model of the nest group hash algorithm to determine the optimized group plan needs to be refined.

7.2 Introduction

As shown in Figure 7.1, the nested group hash method for delta extraction is a multi-staged procedure.

Stage 1. As the 2-staged group method, all initial groups compare with the peer groups at remote site. If the identical groups can be established, then the groups are cleared and are exempt from subsequent data comparisons; otherwise, the procedure of delta identification moves to stage two.

Stage 2. It aims to resolve the changed groups that contain deletions. In stage 1, any group that contains deletion was not able to find its matched group. In this case, it is detected as a changed group. For a change group that is identified containing deletion, after eliminating the deletion, a new group is recomposed. Without further testing, we know this new group is qualified to find its matched group at the remote site. In this case, it is worth of testing whether these matched groups are identical.

Stage 3. At this stage, all changed groups detected from previous stage are partitioned into two subgroups A and B. The cardinality of all initial groups is $|\,G\,|$, two subgroups A and B are randomly selected from their initial group and have the same cardinality, denoted by $\psi = \frac{|G|}{2}$. Since all deletions have been resolved in the previous stage, the procedure only needs to deal with update. At this stage, it is to test the subgroup A.

Stage 4. Stage 4 is the final stage, in which all changed tuples will be resolved unambiguously. For all remaining changed groups, depending on testing result of their subgroup A

- If the identical group of the subgroup A can be found, then inference can be made that the subgroup B is a changed group, i.e. the subgroup B must contain at least one update. Consequently, all tuples in the subgroup B will be tested individually.

- Otherwise, all tuples in the subgroup A will be tested individually. Whether the

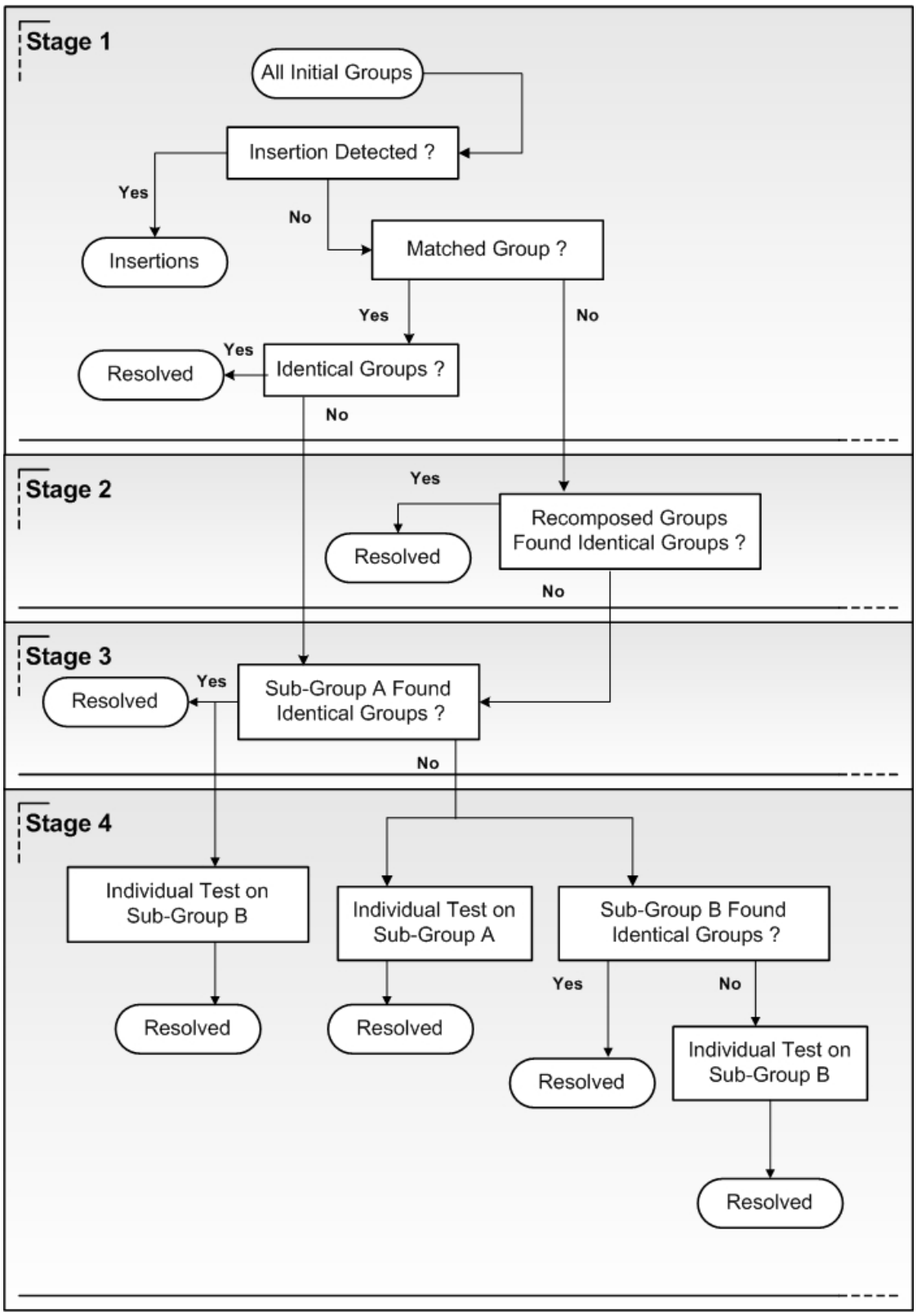

Figure 7.1: The Process of Nested Group Method

subgroup B contains the update is unknown yet, thus we choose to do the group test of subgroup B.

1. in the case that the subgroup B can find its identical group, then procedure is terminated, which means the update only exists in the subgroup A.
2. otherwise, the procedure will test all individual tuples in the subgroup B and then the procedure is terminated. It is the worst scenario, it happens only the updates exist in both subgroups A and B.

7.3 Cost Model for Group Plan

The objective of group plan is to find an optimal one such that the data transfer can be minimized. To evaluate the group plan, it needs to build a cost model, which is used to estimate the expected number of data comparisons required at each stage and to clarify the costs associated with the different types of data comparisons. We focus on the model where the likelihood of any given tuple to be changed is available prior to the design of the procedure of delta extraction.

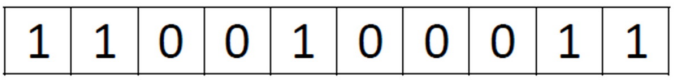

Figure 7.2: Bitstring of Subgroup

The nested group algorithm is a multi-staged procedure, thus it makes sense to use a designated identifier of a group, represented by *Gid*, to avoid repeatedly transferring the key values of individual tuples across different stages of the procedure. In addition, at later stage, the initial group will be partitioned into subgroups. The bitmap *btp* is used to indicate the positions of individual tuples in the initial group. The binary values $\{1, 0\}$ indicate the presence and the absence of tuples from the initial group. Figure 7.2 shows an example of such bitmap. In the example, a 10-bit bitmap is used, which indicates the 1st, 2nd, 5th, 9th and 10th tuples are included in the subgroup where the order is sorted by the key value of tuples. By using the bitmap, it avoids to repeatedly transmitting the key values of individual tuples in a group.

Stage 1. Let $\mathcal{S}_i$ be a snapshot to be synchronized, G is a group of tuples, $|\,G\,|$ be the cardinality of initial group in the stage one and N represents the number of all initial groups and it is

$$N = \frac{|\,\mathcal{S}_i\,|}{|\,G\,|} \tag{7.1}$$

A data unit is composed for each individual group, which contains $\langle Gid, G\kappa, h(G)\rangle$, where Gid is the identifier of a group, $G\kappa$ is the set of all keys in a group and $h(G)$ is

the hash value of a group. The costs incurred at stage one are

$$C_1 = \sum_{i=1}^{N}(sz(k) \times |G| + sz(Gid) + sz(h)) \tag{7.2}$$

Also note that in the multi-staged procedure, only the costs incurred at stage 1, i.e. C_1, are predetermined, while all the costs incurred in the later stage are the expected values based on the probability theory. The number of changed groups that are expected to be further tested after the stage one is represented by N_{cg}, which is obtained from

$$N_{cg} = N \times (1 - P(\equiv)) \tag{7.3}$$

where $P(\equiv)$ was defined by equation (5.14) on page 53 representing the probability of being identical groups.

Stage 2. Stage two aims to resolve the changed groups that contain deletions, in which any changed group that contains deleted tuples will be re-composed by eliminating deleted tuples. Among all initial groups, the number of groups that are expected to contain at least one deleted tuple is represented by N_d, which is obtained from

$$N_d = N \times (1 - P(\ominus)) \tag{7.4}$$

where $P(\ominus)$ is defined by equation (5.12) on page 53 representing the probability that there is *no* deletion in a group. The costs incurred when testing these re-composed groups include Gid and the new hash value of each group. The costs incurred at stage 2 are as follows

$$C_2 = \sum_{i=1}^{N_d}(sz(Gid) + sz(h)) \tag{7.5}$$

Stage 3. When moving to stage three, firstly, it needs to estimate how many changed groups can be resolved from the stage two. When the deletion is eliminated from a changed group, we know the matched groups can be found, but we do not know whether the matched groups are identical groups or not. Among all N_d groups, the expected number of groups that can be resolved after testing the matched groups is represented by N_{rd}. Considering that the update and deletion could possibly exist in

a single group simultaneously, N_{rd} is obtained from

$$N_{rd} = N_d - N \times (1 - P(\ominus)) \times (1 - P(\cong)) \tag{7.6}$$

From equation (7.6), it is straightforward to derive the expected number of changed groups remaining to be tested as

$$N_u = N_{cg} - N_{rd} \tag{7.7}$$

N_u denotes the expected number of changed groups that are caused only by update. At stage three, it is to test the subgroup A of all remaining changed groups and the costs at this stage is

$$C_3 = \sum_{i=1}^{N_u} (sz(Gid) + sz(bmp) + sz(h)) \tag{7.8}$$

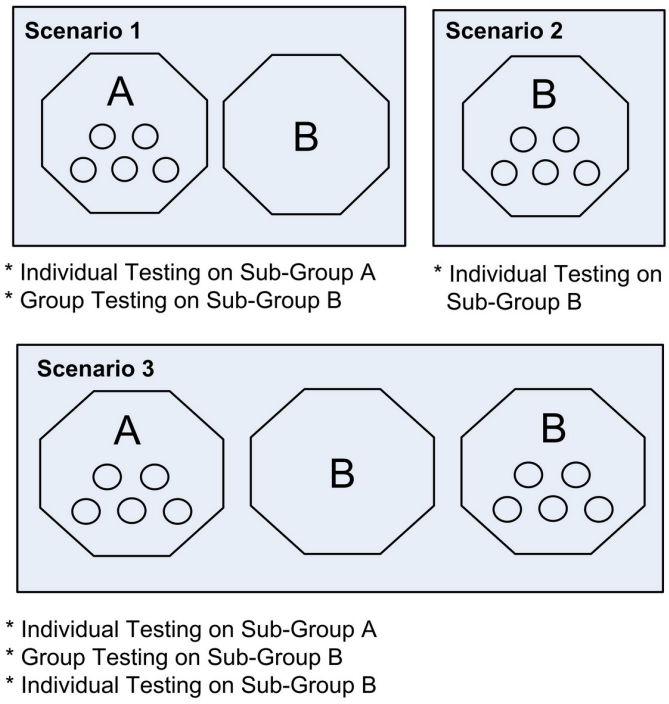

Figure 7.3: Scenarios after Testing Subgroup A

Stage 4. Initially, the subsequent executions are based on the testing results of the subgroups gathered from the stage 3. For each single group, Figure 7.3 shows

all possible subsequent scenarios that could happen after the stage 3. In analyzing the expected behaviors of subsequent executions, it needs to analyze the conditional distribution of changed tuples, and then to estimate how the changed tuples are possibly allocated. Once the allocation of changed tuples is well understood, it is possible to anticipate what scenarios that may occur, and consequently, it is possible to figure out the associated costs.

We use a single group to explain the analysis process. In what follows, the subscript g is used to indicate the costs only associated with a single group. Scenario 1 shown in Figure 7.3 will occur for the situation where the update only exists in subgroup A. The associated costs include

- individual tests of subgroup A and
- a group test for the subgroup B.

The associated costs of scenario 1 are formulated as follows

$$C_g(s1) = (sz(k) + sz(h)) \times \psi) + (sz(Gid) + sz(bmp) + sz(h)) \tag{7.9}$$

Scenario 2 will occur for the situation where the update only exists in subgroup B. The associated costs include

- individual tests of subgroup B.

The associated costs of the second scenario are formulated as follows

$$C_g(s2) = sz(k) + sz(h)) \times \psi \tag{7.10}$$

Scenario 3 will occur for the situation where the updates exist in both subgroups A and B. The associated costs include

- individual tests on subgroup A.
- group test on subgroup B.

- individual tests on subgroup B.

The associated costs of the third scenario are formulated as follows

$$C_g(s3) = (sz(k) + sz(h)) \times |G| + (sz(Gid) + sz(bmp) + sz(h)) \tag{7.11}$$

Let P be the probability of tuple being updated. Given that an initial group identified as a changed group, the conditional probability that the number of updated tuples exists in this group, represented by μ, equals to j is

$$P\{\mu = j \mid \mu \geq 1\} = \frac{\binom{|G|}{j} P^j (1-P)^{|G|-j}}{1-(1-P)^{|G|}} \qquad (j = 1, 2, ..., |G|) \tag{7.12}$$

Parameters	1 Update	2 Update	≥ 3 Update
1: $P = 5\%$, $\|G\| = 6$	87.6%	11.5%	0.8%
2: $P = 3\%$, $\|G\| = 8$	89.7%	9.7%	0.6%

Table 7.1: Allocation Rate of Possible Updated Tuples in an Initial Group

According to equation (7.12), Table 7.1 shows two examples about the allocation rate of updated tuples in a given changed group. For example, when $P = 3\%$ and $|G| = 8$, it has $P(\mu = 1) = 89.7\%$, $P(\mu = 2) = 9.7\%$ and $P(\mu \geq 3) = 0.6\%$ respectively.

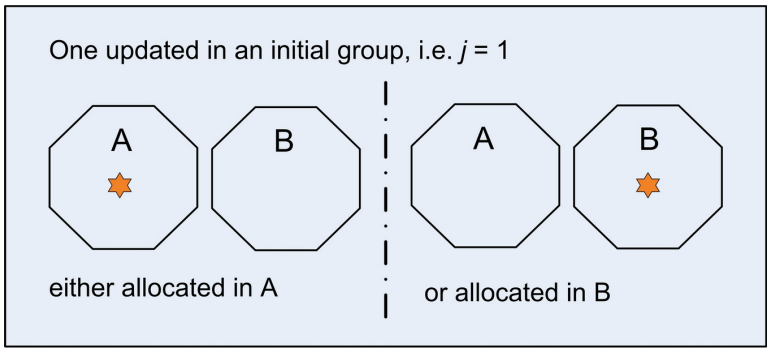

Figure 7.4: One Updated Tuple exists in an Initial Group

If only one updated tuple exists in the initial group, i.e. $j = 1$, as show in Figure 7.4, then this updated tuple is randomly allocated either in the subgroup A or B. In

either case, the probability is 0.5. In the case that $j = 1$ and it is in the subgroup A, then the first scenario as shown in Figure 7.3 will be applied, i.e. the costs of $C_g(s1)$) will be incurred. If it is in the subgroup B, then the second scenario will be applied, i.e. the costs of $C_g(s2)$ will be incurred. From above, let $C_g(j = 1)$ represent the costs required when exactly one updated tuple exists in an initial group, then it has

$$C_g(j = 1) = (C_g(s1)) + C_g(s2)) \times 0.5 \tag{7.13}$$

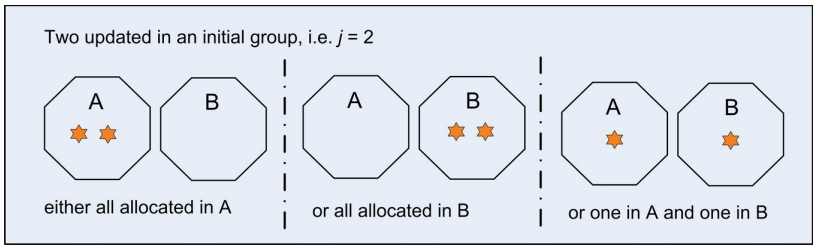

Figure 7.5: Two Updated Tuples exist in an Initial Group

If two updated tuples exist in the initial group, i.e. $j = 2$, as show in Figure 7.5, then there are three possible allocations of these two tuples. Firstly, two updated tuples are allocated into a single subgroup, i.e. either all in A or all in B. Its probability, represented by $P(j = 2, A\,or\,B)$, can be obtained from

$$P(j = 2, A\,or\,B) = \frac{\binom{|G|-2}{\psi-2}}{\binom{|G|}{\psi}} \times 2 = \frac{|G| - 2}{2\,(|G| - 1)} \tag{7.14}$$

If it is in the subgroup A, then the first scenario is applied, i.e. the costs of $C_g(s1)$ will be incurred. If it is in the subgroup B, then the second scenario will be applied, i.e. the costs of $C_g(s2)$ will be incurred. These two tuples being randomly allocated either in subgroup A or B are equivalent events, i.e. having the equivalent probability of $P(j = 2, A\,or\,B)/2$. The costs are

$$C_g(j = 2, A\,or\,B) = (C_g(s1) + C_g(s2)) \times \frac{P(j = 2, A\,or\,B)}{2} \tag{7.15}$$

Secondly, these two updated tuples are allocated into two subgroups separately, i.e. one is in A and the other is B. Its probability, represented by $P(j = 2, A\,and\,B)$, can

be obtained from

$$P(j=2, A\,and\,B) = \frac{\binom{2}{1}\binom{|G|-2}{\psi-1}}{\binom{|G|}{\psi}} = 1 - \frac{|G|-2}{2\,(|G|-1)} \tag{7.16}$$

In this case, only the third scenario can be applied, i.e. the costs of $C_g(s3)$ will be incurred. By considering its probability, the costs can be obtained from

$$C_g(j=2, A\,and\,B) = C_g(s3) \times P(j=2, A\,and\,B) \tag{7.17}$$

By combining equations (7.15) and (7.17), the expected costs required when exactly two updated tuples exist in a changed group are as follows

$$C_g(j=2) = C_g(j=2, A\,or\,B) + C_g(j=2, A\,and\,B) \tag{7.18}$$

For the case that three or more updated tuples exist in a changed group simultaneously, i.e. $j \geq 3$, as shown in Figure 7.6, its probability would be small for small P. To avoid complex analytic calculations, we approximated the costs of $j \geq 3$ by using the worst scenario, i.e. $C_g(s3)$ as follows

$$C_g(j \geq 3) \approx C_g(s3) \tag{7.19}$$

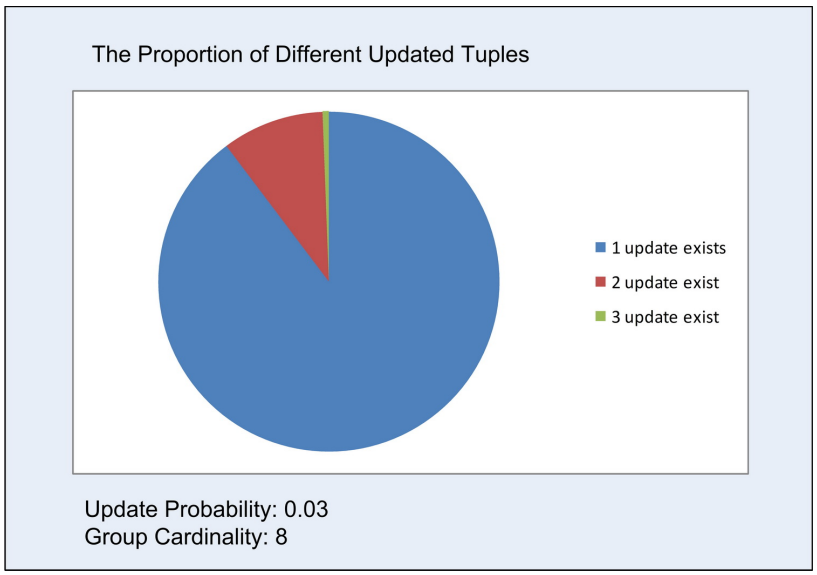

Figure 7.6: The Proportion of Different Updated Tuples Exist in a Changed Group

By considering the various scenarios and the value of $P(\mu = j)$, the expected costs that are required to identify all updated tuples in a single changed group at stage 4 are generalized as follows

$$\begin{aligned} C_g = {} & C_g(j = 1) \times P(\mu = 1) + \\ & C_g(j = 2) \times P(\mu = 2) + \\ & C_g(j \geq 3) \times P(\mu \geq 3) \end{aligned} \tag{7.20}$$

From above, the total expected costs at the stage 4 are approximated as

$$C_4 = \sum_{i=1}^{N_u} C_g \tag{7.21}$$

Finally, the expected costs for the task of delta identification when using the nested procedure are

$$C = C_1 + C_2 + C_3 + C_4 \tag{7.22}$$

7.4 Experiments

Parameters	Value
1: No. of Total Tuples	100,000
2: Size of Snapshot	38 MB
3: Length of Tuple	400 Bytes
4: Length of Key	7 Bytes
5: Length of Hash	16 Bytes
6: Length of Gid	4 Bytes
7: Random Function	dbms_random.random

Table 7.2: Nested Group Parameters

The experiments presented in this section aims

- to verify the method of the cost analysis for the nested group algorithm.
- to evaluate the efficiency of the nested group algorithm.
- to compare the performance between nested group algorithm and the 2-staged group algorithm that is introduced in Chapter 5.

The experiments use a test dataset of 100,000 tuples and the snapshot size is 38 MB. The changed dataset is generated by the Oracle function *dbms_random*. As shown in the Table 7.2, for the volume of data, individual tuples are specified by using the unit of Byte, while for better readability, the results of experiment are reported by using the unit of MB.

7.4.1 Allocation of Changed Tuples in Stage 4

In this chapter, we present an analytic approach to estimate the total costs of nested group algorithm. The analysis is based on the probability theory and the majority of analysis focus on the final stage, i.e. stage 4. Essentially, it has to have a good estimation about the possible allocation of changed tuples after the testing of the

subgroup A, which aims to estimate that given all identified changed groups, how many groups have

- *1A* : exactly one updated tuple and it is in the subgroup A;
- *1B* : exactly one updated tuple and it is in the subgroup B;
- *2A* : exactly two updated tuples and all in the subgroup A;
- *2B* : exactly two updated tuples and all in the subgroup B;
- *2AB* : exactly two updated tuples and one is in A and the other is in B;
- ≥ 3 : 3 or more changed tuples.

	Change Pct	No.Cg	Allocation of Changed Tuples								Costs
Exp	**1**	947	448	448	896	11	11	27	50	2	0.138
Actl		948	445	453	898	14	9	25	48	2	0.138
Exp	**3**	2552	1073	1073	2146	83	83	199	365	40	0.395
Actl		2552	1094	1051	2145	68	94	205	367	40	0.395
Exp	**5**	3831	1422	1422	2844	187	187	449	823	163	0.629
Actl		3829	1398	1433	2831	179	179	483	841	157	0.628
Exp	**7**	4846	1576	1576	3151	296	296	712	1305	390	0.841
Actl		4848	1586	1572	3158	325	288	686	1299	391	0.830

Table 7.3: The Comparison of Expected and Actual Result at Stage 4

This experiment is designed to verify the the estimation method for the allocation of changed tuples. The estimation method was specified by equations (7.12), (7.14) and (7.16). Table 7.3 shows the results from both our estimation and the result collected from the actual executions. The experiments have been conducted at the data change rates of 1%, 3%, 5% and 7% respectively and are executed by using the setting of group cardinality *12*. Notice that using the group cardinality of *12* is just for consistent view of results, and it is not necessary to be the optimum group based on the cost estimation. The column *No. Cg* represents the number of changed groups at stage 4 need to be resolved. Let *T1* and *T2* denote the total numbers of changed groups that contain

exactly one and exactly two updated tuples respectively, then the column *Allocation of Changed Tuples* in the table is read following the order of *1A 1B T1 2A 2B 2AB T2 and* $\geq$*3*.

The results in Table 7.3 demonstrate that the proposed estimation method generates a good approximation about the possible allocation of changed tuples at the stage 4. Although some differences exists in some categories, but the estimated numbers of *T1, T2* and $\geq$*3* are consistent with the results collected from actual executions. Consequently, in all cases, the estimated overall costs in the stage 4 are highly consistent with the costs incurred in the actual executions.

7.4.2 Efficiency Evaluation

Change Pct	Optimum Group	Expected Costs	Actual Costs	Usage Pct
1	14	0.981	0.981	2.58
3	8	1.239	1.239	3.26
5	6	1.431	1.429	3.76
7	6	1.593	1.593	4.19

Table 7.4: Efficiency Evaluation

Table 7.4 demonstrates the results of the expected costs and the actual costs. Same as before, the comparisons are conducted at data change rates of 1%, 3%, 5% and 7% respectively. With each change rate, the executions use the optimum groups that are decided by the cost estimation model. The column of *optimum group* in the table indicates when the cardinality of initial group is set to the optimum group, the overall costs of nested group algorithm are optimized. From the results, we can see that in all test cases, the expected costs that are generated by the estimation model and the actual costs that are incurred during the actual execution are consistent. The column of *Usage Pct* represents out of the the overall size of the snapshot, what the percentage is required for the task of delta identification. For example, in the case that the change rate is 5%, when the cardinality of initial group is set to 8, the delta identification

requires $38 \times 0.0376 = 1.43$ MB of the data volume. The the size of differential file can be obtained from the value of the column *Change Pct*, i.e. $38 \times 0.05 = 1.9$ MB. By considering to transport the differential file, the costs required for the task of delta extraction are $38 \times (0.05 + 0.0376) = 3.3$ MB that is 8.76% of overall the size of snapshot. Comparing to the complete fresh method, the saving ratio of 91.23% has been achieved.

7.4.3 The Comparison with 2-Staged Group Strategy

	Change Pct	Optimum Group	Exp Costs	Avg. Costs	S.D. σ_μ	Avg Qry μ_q
2Stg	**1**	10	1.065	1.068	0.003	109559
Nest		14	0.981	0.981	0.001	116740
2Stg	**3**	5	1.359	1.358	0.002	114122
Nest		8	1.239	1.238	0.001	128031
2Stg	**5**	4	1.552	1.552	0.002	118541
Nest		6	1.431	1.430	0.002	134613
2Stg	**7**	4	1.698	1.698	0.002	125204
Nest		6	1.593	1.593	0.002	147011

Table 7.5: The Comparison with 2-Staged Grouping Strategy

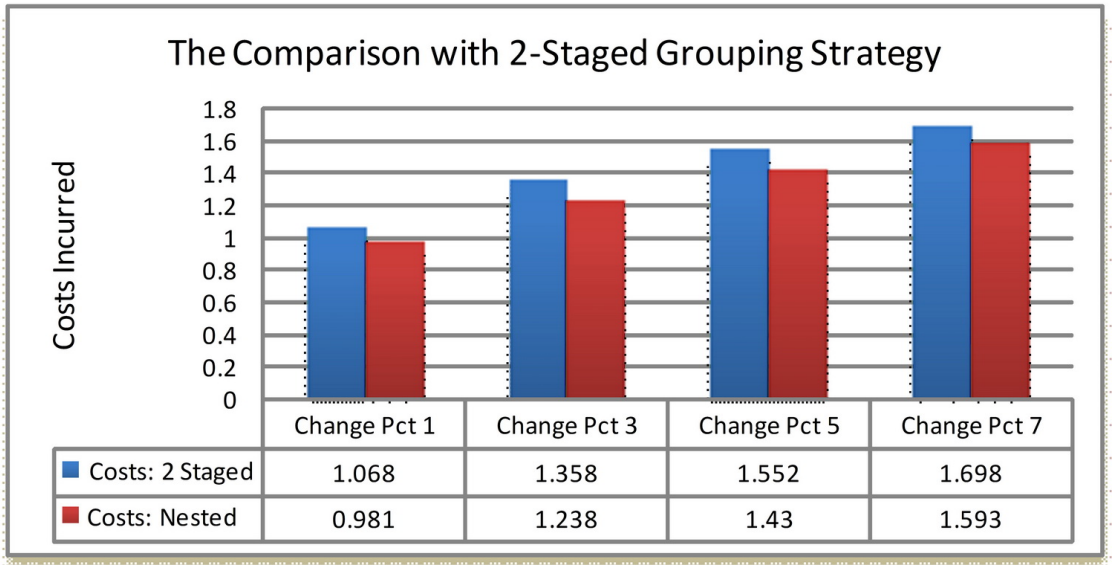

Figure 7.7: Comparison of Communication Costs

The experiment aims to make the comparison between 2-staged algorithm (abbreviated 2stg in the table) and the nested algorithm in term of the optimum group, expected costs, average costs and the stability. The stability is measured by the standard deviation between expected costs and the costs collected from actual executions. Table 7.5 lists the results of the data change rates of 1%, 3%, 5% and 7%. For each data change rate, the reported results are collected from 60 independent executions of both the 2-staged algorithm and the nested algorithm. To maintain the independence and the randomness, a test data set of 100,000 tuples is re-created at every individual execution.

Firstly, when processing the test data with the same change rate, the optimum group chosen by the nested algorithm are generally larger than the ones chosen by 2-staged algorithm. Secondly, the results of both estimation model and the executions are shown to be consistent, which is proving that both estimation models generate reliable results. Thirdly, as shown in column *Avg. Costs*, in the conducted 60 executions, for all different data change rates, comparing to the 2-staged algorithm, the nested group algorithm always achieves better cost saving. The comparisons are also shown in Figure 7.7. At the data change rates of 1%, 3%, 5% and 7%, comparing to the 2-staged algorithm, the nested algorithm further reduced 8.1%, 8.8% 7.8% and 6.2% of communication costs for the task of delta identification.

Also in Table 7.5, the column of σ_μ is the standard deviation, which is used to quantify the amount of variation between the expected costs and actual costs collected from each individual executions. Let x_i represent the costs collected from the ith execution and let C represent the expected costs, for the conducted 60 executions, i.e. $N = 60$, σ_μ is obtained from

$$\sigma_\mu = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - C)^2} \tag{7.23}$$

If σ_μ is close to 0 indicates that the actual costs of individual executions tend to be very close to the the expected costs, while a higher value of σ_μ indicates that the actual

costs are spread out over a wider range of values. In the table, we can see that both algorithms at all change rates only have very small amount of variation, which also explains why the estimation and actual costs are consistent.

Finally, also in Table 7.5, the column of *Avg. Query*, denoted by μ_q, indicates the average number of queries to the source systems, which is required to complete the task of delta identification. Let $cnt(q)_i$ represent the number of queries required by ith execution, for the conducted 60 executions, i.e. $N = 60$, μ_q is obtained from

$$\mu_q = \frac{1}{N} \sum_{i=1}^{N} cnt(q)_i \tag{7.24}$$

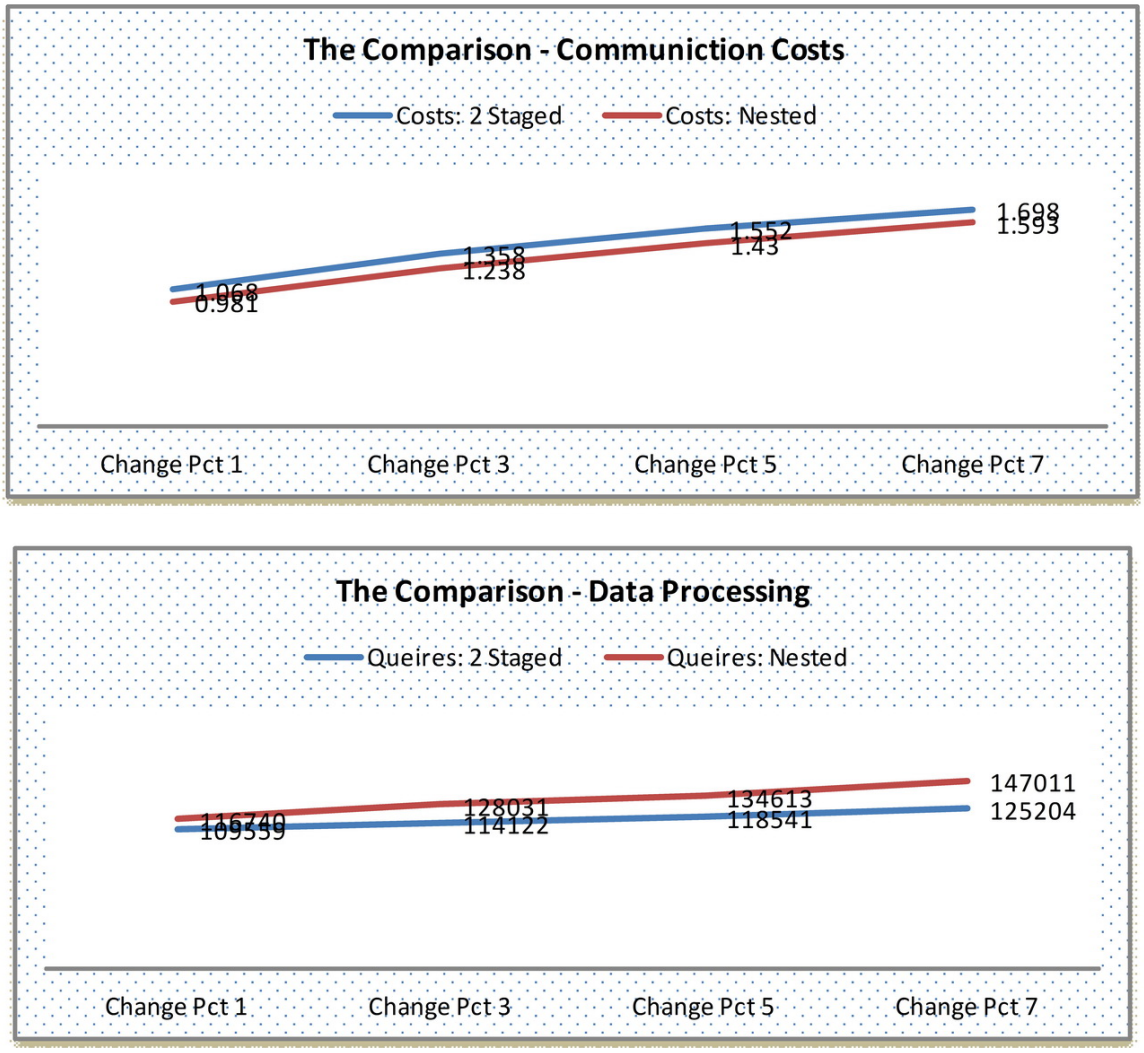

Figure 7.8: Cost Saving at the Expense of Additional Data Processing

Comparing to the 2-staged algorithm, the results of the column *Avg Qry* show that, the further cost saving achieved by the nested algorithm is at the expense of some additional data processing, i.e. need to impose the increased number of queries to the source systems. According to two columns *Avg. Query* and *Avg. Query* in Table 7.5, Figure 7.8 contains two graphs. The graph on the top shows the comparison between the 2-staged algorithm and nested algorithm in terms of communication costs incurred for the task of delta identification, while the graph at the bottom shows the comparison in terms of data processing. The two graphs illustrate the relationship between the goal, i.e. reducing communication costs, and the side effect, i.e. additional data processing. The results show that, at the data change rates of $1\%, 3\%, 5\%$ and 7%, for the task of delta identification, the $8.1\%, 8.8\%, 7.8\%$ and 6.2% further reduction on communication costs are achieved by the nested algorithm are at the expense of incurring addtional 6.6%, 12.2%, 13.5% and 17.4% of data queries to the sources systems respectively.

7.5 Summary

In this chapter, we presented a nested group algorithm for the task of delta identification and used an analytical approach for building its associated cost model. When developing the cost model, comparing to 2-staged group hash algorithm, the nested group algorithm needs to take it into account the conditional distribution of changed tuples within the detected change groups. More specifically, given a detected changed group, how many changed tuples it contains and of which what are the probabilities. This question was answered by equation (7.12). In addition, to accurately estimating the incurred costs, it also needs to consider all the scenarios that for how these changed tuples are possible allocated after the changed groups are partitioned into two subgroups.

We have conducted the experiments to verify the accuracy and efficiency of the nested group algorithm. The experiment results have shown that the proposed cost model is accurate and is consistent with the results collected from actual executions. The efficiency of the nested group algorithm is also satisfactory, in that the experiment results showed that for the data change rates of 1%, 3%, 5% and 7%, only 2.58% 3.26% 3.76% and 4.19% of total data volume are required for the task of delta identification. When comparing to the 2-staged group algorithm, the results showed that the further reduction on communication costs has been achieved by the nested procedure, which is at the expense of some additional data processing.

We are not claiming the nested procedure is the most efficient algorithm for the task of delta identification. Actually, we believe that if the changed groups are further split, it is possible to further reduce the costs, although it may not be significant. However, when facing various pricing models [7, 35, 85] for the emerging trend of the online provisioning of computing resources as services, this work provides the opportunities to balance the costs between the data processing and network traffic, such that a desired and cost effective method can be adopted.

Chapter 8

Summary and Future Work

8.1 Summary

In this thesis, we addressed the problem of delta extraction in a distributed environment, which is an area important to the problem of data integration. With a limited degree of collaboration, the data integration from the distributed and highly autonomous operational database systems has to overcome many organizational and technical problems. Therefore, the existing assumptions and techniques must be re-examined. To help understand the problem environment, in chapter 1, we have categorized the different collaboration levels. At the lowest level, a differential file is inferred by comparing a snapshot to its associated remote data source. Our objective has been set to reduce the communication costs over the computer networks when extracting a differential file.

In chapter 4, as the preliminaries, we briefly described the data warehouse environment, view decomposition and snapshot projection. We also presented the formal definition of a differential file, which is the representation of data changes and can be used to synchronize the data.

In chapter 5, we presented the 2-staged group strategies and algorithms for delta extraction. Firstly, the statistics about change probabilities of remote data need to be maintained at a data warehouse system. Maintaining the statistics certainly requires some extra efforts in the data warehouse system, but the collected statistics can be

used to construct into a simple model of the prior probabilities, which helps to capture the essential characteristic about remote data. The essential characteristic of data reflected by the statistical model in turn will create the possibilities of optimization. More specifically, we have developed the group strategies, such that the delta can be identified by examining the difference between groups of tuples, rather than testing individual elements. As the result, the volume of data transfer between data sources and a data warehouse system can be significantly reduced. Regarding to the nature of data changes, we have considered two broad scenarios: namely, uniformly distributed changed data, and the heterogeneous mixtures of changed data. In both cases, cost-driven group strategies have been developed. When dealing with uniformly distributed changed data, the idea is to find the group plan that requires the minimum volume of data transfer to complete the task of data extraction. On the other hand, for the heterogeneous mixtures of changed data, the idea is to find the group plan that could yields maximum savings.

In chapter 6, firstly, we conducted the simulations to show the negative impacts can be caused by outlier data. The results of the simulation support the claims that the capture of data change pattern is the necessary complement to the group hash algorithm. Accordingly, we have defined the progression pattern, which can be used to describe data change with temporal regularities. We also proposed a method for the progression pattern discovery, which is defined to be a process of forming an explanatory hypothesis from an observed irregularity of update event requiring explanation. In the pattern discovery process, the domain knowledge is visualized as three categories, namely, observations, instance knowledge and historical knowledge. Accordingly, the discovery process consists of three steps:

1. event monitoring, which is to detect the occurrence of interesting event. What have been monitored is considered as facts.

2. hypothesis generation, which is to discover the set of p-variables, i.e. progression

features, as explanatory hypothesis accounting for the observed facts, and then measure the temporal association between the progression features and effects. Also it is where instance knowledge is used, which links the observed facts to the historical knowledge.

3. hypothesis evaluation. The identified hypothesis is the plausible explanation for the observed facts. The strength of the hypothesis needs to be evaluated, which is to consider the degree of confirmation of hypothesis. The evaluation is conducted by referencing historical knowledge.

In chapter 7, we presented a nested group hash method for delta extraction aiming to seek the possibility of continual improvement. We have conducted the experiment to compare the performance with the 2 stage group algorithm. The results showed that in terms of reducing the volume of data transfer, improvement is possible, and has been achieved by the nested procedure. We believe that if continuing to split the changed groups, it is possible to reduce the costs further, although the costs may not be reduced significantly. To estimate the costs in the case of continuing to split the changed groups, it is just a matter of repeating the analysis upon

- the conditional distribution of changed tuples, to calculate the probability that given a detected changed group, the number of updated tuples in this group is equal to j (See also equation (7.12)).

- the allocations of changed tuples, to estimate given j changed tuples exist in a changed group, and how theses changed tuples are possibly allocated after the changed group is split (See also equations (7.14) and (7.16)).

This thesis has established a new approach for problem of delta extraction in a limited collaborative environment, which is motivated to facilitate the data integration in a highly autonomous environment. This research work is aiming to overcome the organizational and technical problems for the situation where the data integration

needs to operate in a distributed environment and involves highly autonomous source systems. The optimization goal is to minimize communication costs for the task of delta extraction as much as possible. We summarize our contributions as follows:

1. The first contribution of this thesis is that we developed a 2-stage group hash method for the task of delta extraction to be used in a data warehouse and also built a prototype. Extensive performance evaluations confirm that comparing to the data compression and other traditional methods, the 2-stage group hash method is a highly efficient solution to the problem of delta extraction, which consists of four essential steps

   (a) A data warehouse collects and maintains the statistics about remote data.

   (b) An estimation model uses the statistics to generate a list of group plans that might be associated with different costs.

   (c) The group plan selection decides the optimized group plan.

   (d) A data warehouse executes the process of delta extraction according to the optimized group plan.

   When designing 2-stage group hash method, based on the probability theory, we developed the estimation model that reveals the cause-effect relationship between a group cardinality and the communications incurred in the process of delta extraction. In addition, the extreme value theorem is used when deciding the optimized plan.

2. The second contribution of this thesis is that we developed a more complex nest group hash method to seek the possibility of continual improvement. A prototype has also been implemented and extensive performance evaluations confirm that comparing to the a 2-stage group hash method, a nested group method leads to an improvement in efficiency as well as reliability. The nested group hash method is a multi-staged method, which is characterized by the very natural constraint

that all elements to be examined in the next stage must be chosen from the previously identified changed groups. Same as the 2-stage method, a nested group hash method also consists of four essential steps. However, the nested analysis of conditional probability has been used to make the inference on the likelihood of what would occur in next stage given the estimation of the current stage. The statistical analysis also have been used to modeling the allocation of changed elements at different stages.

3. The third contribution of this thesis is that we have defined the progression pattern and proposed a method for the pattern discovery to handle the outlier and relax the assumption that the changes to remote data are only caused by random events. Firstly, we conducted simulation to show how the negative impacts are caused by outlier data. The results of simulation help to understand the associations between the data change deviation and the extra costs incurred. Secondly, for the purpose of detecting the occurrence of interesting events, we have used the technique *normal approximation to the binomial distribution* in the frequency domain of data evolution. Thirdly, the temporal association between progression features and effects may vary over a wide range, therefore it would be infeasible to exhaustively enumerate all the possible combinations one after another. To cope with this computationally expensive task, one of the inference principle we used to guide the discovery process is that the interesting update event observed from previous step is believed to be the consequent of the progression rule unless the contrary is proved in the succeeding process. In addition, to avoid highly redundant exploration for the task of candidate p-variables generation, the bi-directional pruning rules have been developed and included in the search algorithm.

Our focus in this research is to use data warehouse approach to integrate data for business intelligence environments. Typically, the problem domain is characterized by:

a very large and constantly growing historical datasets that are distributed over the wide area networks, and a process of data integration that is usually scheduled in a batch mode. A data warehouse benefits from using this approach when the higher level of collaborations are not easily available, the data acquisition can still be completed in an efficient way. The remote systems benefit from using this approach when they need to publish or share the data with other parties, but they are not willing or not able to do not provide the customized functions to support the view maintenance process. From a practical perspective, we have built the prototypes of proposed group hash methods in an Oracle environment. We believe it can also be easily implemented and commercialized in any database systems as a stand alone module.

Although this thesis is motivated to facilitate the data integration in a highly autonomous environment, we also believe it can be directly applied to the problem of massive data replication, duplication, synchronization and version management in distributed database systems.

8.2 Limitations and Future Work

8.2.1 Limitations

We stress that we are not going to claim that the proposed approach is the best solution to the problem of delta extraction. As pointed out in [56], all methods of detecting changes by the data comparison have the problem of scalability, which means as the volume of source data grows, we have to handle a growing amount of comparisons. We are saying that, however, this approach is complementary to some other approaches, because it creates a possibility to maintain a materialized view when the higher level of collaborations are not easily available. Although the extensive performance evaluations have confirmed the effectiveness of this proposed approach, it has several important limitations:

1. In the context of relational databases, comparing against the properties of the ACID, the group strategies satisfy the properties of atomicity, isolation and durability. However, due to the nature of batch processing, the group strategies only support weaker level of consistency.

2. In the context of distributed computer system, the design of group strategy prioritizes availability with a reduced level of data consistency. Accord the CAP theorem, the group strategy follows the pattern of AP and forfeit C.

3. The proposed method works best when the relational data in consecutive snapshots are very similar. As the ratio of changes is increased, although the method will also function correctly, the efficiency would be decreased.

4. As we demonstrated in Chapter 7, although further reducing the communication costs is possible, it is always at the expense of some additional data processing.

5. For the relational data with very small size of tuple, the efficiency would be

decreased, as the group hash value took up a considerable amount of space comparing to the original data size.

8.2.2 Future Work

In this subsection, we discuss some opportunities for future work in topics related to this thesis.

- The statistics about the data sources are collected and maintained every time the data synchronization is complete. The disadvantage of this approach is that the data synchronization has to be scheduled to run regularly, such that the statistics can be maintained in a consistent manner. As an enhancement, we can adopt sampling techniques to estimate change probabilities. With this enhancement, firstly, the data warehouse does not require to collect and maintain statistics. More importantly, it provides the flexibility that the synchronization could run in arbitrary frequencies as desired.

- Moving beyond PSJ view, the strategies and concepts developed in this thesis can be extended to the problem of delta extraction for other types of materialized views, for example, set query based views and aggregate views.

- It would also be interesting to improve the progression pattern discovery by adding the functionality of changed value prediction. The idea is that instead of excluding the identified pattern as the outlier, using the predicted values from the pattern to join in the group comparison. In this case, the nature of data comparison becomes to verify and confirm the prediction, rather than to detect the changes.

- Considering various pricing models [6, 15, 43] for the emerging trend of the cloud computing environment, as the future work, it would be helpful to design a mechanism that is used to balance the trade-off between data precessing that

is measured by CPU time and RAM usage and communication costs that is measured by the traffic allotment. According to the pricing model, a set of control parameters should be identified and defined, such that the process of delta extraction can be configurable in different settings according to the underlying pricing model.

- The proposed optimization techniques for the task of delta extraction are largely based on the relational model, where databases are organized as a collection of structured data with some clearly defined constraints. As the increasing need for handling big data, nowadays, the paradigm of data storage is shifting from relational databases to NoSQL like databases. Same as the design of NoSQL databases, the process of delta extraction also prioritizes availability with a reduced level of data consistency. However, the data structures managed by NoSQL databases include key-value, wide column, graph and document, which cannot be handled by the current process of delta extraction. It seems promising to extend our current optimization techniques to support NoSQL databases. This extension would be a great idea to solves the problems for big data management and address the challenges such as scalability, availability, high performance and particularly, large data transportation from big data perspective.

Bibliography

[1] Abadi, D. (2009) 'Data management in the cloud: limitations and opportunities', *IEEE Data Engineering Bulletin*, Vol. 32, No. 1, Pages 3-12.

[2] Agrawal, D., Abbadi, A. E., Singh, A. K. et al. (1997) 'Efficient view maintenance at data warehouses', In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Pages 417-427.

[3] Agrawal, D., Das, S. and Abbadi, AE. (2011) 'Big data and cloud computing: current state and future opportunities' In *Proceedings of the 14th International Conference on Extending Database Technology*, Pages 530-533.

[4] Agrawal, R., Imielinski, T. and Swami, A. (1993) 'Mining association rules between sets of items in large databases'. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Pages 207-216.

[5] Agrawal, R. and Srikant, R. (1995) 'Mining sequential patterns', In *Proceedings of 11th International Conference on Data Engineering (ICDE)*, Pages 3-14.

[6] Allen, JF. and Ferguson, G. (1994) 'Actions and events in interval temporal logic', *Journal of logic and computation*, Vol.4, No. 5, Pages 531-579.

[7] Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., and Ahmad, I. (2013) 'Cloud computing pricing models: a survey', *International Journal of Grid and Distributed Computing*, Vol. 6, No. 5, Pages 93-106.

[8] Armbrust, M. and Fox, A. (2010) 'A view of cloud computing', *Communications of the ACM*, Vol. 53, No.4, Pages 50-58.

[9] Bergamaschi, S., Castano, S. and Vincini, M. (1999) 'Semantic integration of semistructured and structured data sources', *SIGMOD Record*, Vol. 28, No. 1, Pages 54-59.

[10] Bernstein, P., Goodman, N. and Wong, E., et al. (1981) 'Query processing in a system for distributed databases (SDD-1)', *Journal of ACM Transactions on Database Systems (TODS)* Vol. 6 No.4, Pages 602-625.

[11] Bernstein, P. and Haas, L. (2008) 'Information integration in the enterprise', *Communications of the ACM* Vol. 51, No.9, Pages 72-79.

[12] Bettini, C., Wang, S. et al. (1998) 'Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences', *IEEE Trans. Knowledge and Data Engineering*, Vol. 10, No. 2, Pages 222-237

[13] Bhattacharya, I. and Getoor, L. (2004) 'Iterative record linkage for cleaning and integration', *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, Pages 11-18.

[14] Boonstra, A. and Vries, J De. (2005) 'Analyzing inter-organizational systems from a power and interest perspective', *International journal of information management*, Vol. 25, No. 6, Pages 485-501.

[15] Brewer, E. (2000) 'Towards robust distributed systems', In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, Page 7.

[16] Cali, A., Lembo, D., Rosati, R. (2003) 'Query rewriting and answering under constraints in data integration systems', In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Pages 16-21.

[17] Castano, S. and Antonellis, V. D. (1999) 'A discovery-based approach to database ontology design', *Journal Distributed and Parallel Databases*, Vol. 7, No. 1, Pages 67 - 98.

[18] Ceri, S. and Widom, J. (1991) 'Deriving production rules for incremental view maintenance', In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB)*, Pages 577-589.

[19] Chawathe, SS. and Garcia-Molina, H. (1997) 'Meaningful change detection in structured data', In *Proceedings of the 1997 ACM SIGMOD International conference on Management of Data*, Pages 26-37.

[20] Chen, H., Chiang R. et al. (2012) 'Business intelligence and analytics: from big data to big impact', *Journal of MIS Quarterly*, Vol. 36, No 4, Pages 1165-1188.

[21] Chen, S., Liu, B. and Rundensteiner, E. (2004) 'Multiversion Based View Maintenance over Distributed Data Sources', *ACM Transactions on Database Systems*, Vol. 29, No.4, Pages 675-709.

[22] Denning, P. J. (2005) 'The Science in Computer Science', *Communications of the ACM*, Vol. 48 No. 4, Pages 27-31.

[23] Deutsch, P. and Gailly, J. (1996) ZLIB compressed data format specification version 3.3.

[24] Do, HH. and Rahm, E. (2002) 'COMA: a system for flexible combination of schema matching approaches', In *Proceedings of of the 28th international conference on Very Large Data Bases (VLDB)*, Pages 610-621.

[25] Doan A., Domingos, P. and Halevy, A.Y. (2001) 'Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach', In *Proceedings the 2001 ACM SIGMOD international conference on Management of data (SIGMOD)*, Pages 509-520.

[26] Dorfman, R. (1943) 'The detection of defective members of large populations', *The Annals of Mathematical Statistics*, Vol. 28, No. 4, Pages 1033-1036.

[27] Du, D.Z., and Hwang, F.K. (2006) *Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing. World Scientific.*

[28] Easterbrook, S., Singer, J. el al. (2008) 'Selecting Empirical Methods for Software Engineering Research', *Guide to Advanced Empirical Software Engineering*, Pages 285-311.

[29] Foster, I., Zhao, Y., Raicu, I., Lu, S. (2008) 'Cloud Computing and Grid Computing 360-Degree Compared', In *Proceedings of IEEE Grid Computing Environments Workshop*, Pages 1-10.

[30] Galhardas, H., Florescu, D., Simon, E. et al. (2001) 'Declarative data cleaning: language, model, and algorithms', In*Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Pages 371-380.

[31] Ganti, V., Gehrke, J. and Ramakrishnan, R. (2001) 'Demon: mining and monitoring evolving data', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13 No.1, Pages 50-63.

[32] Giordano, A. (2011) *Data Integration Blueprint and Modeling: Techniques for a Scalable and Sustainable Architecture*, IBM Press.

[33] Gilbert, S., Lynch, N. (2002) 'Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services', *ACM SIGACT News*, Vol. 33, No. 2, Pages 51-59.

[34] Goethals, F. (2008) 'Important issues for evaluating inter-organizational data integration configurations', *Electronic Journal Information Systems Evaluation*, Pages 185-196.

[35] Gohad, A., Narendra, NC. and Ramachandran, P. (2013) 'Cloud Pricing Models: A Survey and Position Paper', In *Proceedings of Cloud Computing in Emerging Markets*, Pages 1 - 8.

[36] Goodhue, D., Michael D. et al. (1992) 'The Impact of Data Integration on the Costs and Benefits of Information Systems', *MIS Quarterly*, Vol. 16, No. 3, Pages 293-311

[37] Gray, J. and Patterson, D. A. (2003) 'conversation with Jim Gray', *ACM Queue*, Vol.1, No. 4, Pages 8-17.

[38] Greco, G., Greco, S., Zumpano, E. (2001) 'A logic programming approach to the integration, repairing and querying of inconsistent databases', In *Proceedings of the 17th International Conference on Logic Programming*, Pages 348-364.

[39] Grolinger, K. and Higashino, WA. (2013) 'Data management in cloud environments: NoSQL and NewSQL data stores', *Journal of Cloud Computing: Advances, Systems and Applications*, Vol.2, No. 1.

[40] Gupta, A., Mumick, I. and Subrahmanian, V. (1993) 'Maintaining views incrementally', In *ACM SIGMOD Record*, Pages 157-166.

[41] Gupta, A., and Mumick, I.S. (1995) 'Maintenance of Materialized Views: Problems, Techniques and Application', In *Bulletin of Technical Committee on Data Engineering*, Pages 145-157.

[42] Haake, A. (1992) 'CoVer: A Contextual Version Server for Hypertext Applications', *Proceedings of the ACM conference on Hypertext*, Pages 43-52.

[43] Halevy. A., Rajaraman, A. and Ordille, J. (2006) 'Data integration: the teenage years', In *Proceedings of the 32nd international conference on Very large data bases (VLDB)*, Pages 9-16.

[44] Han, J., Dong, G. and Yin, Y. (1999) 'Efficient mining of partial periodic patterns in time series database', In *Proceedings of 15th International Conference on Data Engineering (ICDE)*, Pages 106-115.

[45] B. He and K. Chang. (2003) 'Statistical Schema Matching across Web Query Interfaces', In *Proceedings the 2003 ACM SIGMOD international conference on Management of data (SIGMOD)*, Pages 217-228.

[46] Helmut, K. Tessarek, C. (2014) 'Documentation for db2-hash-routines'. [online] http://mod-authibmdb2.sourceforge.net. Accessed 23 Feb 2015.

[47] Hull, R. and Zhou, G. (1996) 'A framework for supporting data integration using the materialized and virtual approaches', In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, Pages 481-492.

[48] Inmon, WH., (2005) 'Building the data warehouse', John wiley & Sons Inc., New York, NY, USA.

[49] Kawaguchi, A., Lieuwen, D. F., Mumick, I.S. and Ross. K.A. (1998) 'Implementing incremental view maintenance in nested data models', In *Proceedings of the 6th International Workshop on Database Programming Languages*, Pages 202-221.

[50] Kimball, R. and Ross. M. (2002) *The data warehouse toolkit: the complete guide to dimensional modeling*, John Wiley & Sons Inc., New York, NY, USA.

[51] Kirk, T., Levy, A. Y., Sagiv, Y. et al. (1995) 'The Information Manifold', In *Proceedings of the AAAI 1995 Spring Symp on Information Gathering from Heterogeneous, Distributed Environments*, Pages 85-91.

[52] Kolaitis, P. (2005) 'Schema mappings, data exchange, and metadata management', In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*, Pages 61-75.

[53] Koschel, A. and Lockemann, P. (1998) *Distributed events in active database systems - Letting the genie out of the bottle. Data & Knowledge Engineering*, Vol. 32, No.1-2, Pages 11-28.

[54] Kossmann, D. (2000) 'The state of the art in distributed query processing', *Journal of ACM Computing Surveys* , Vol. 32, No.4, Pages 422-469.

[55] Kuncheva, L. (2004) 'Classifier ensembles for changing environments'. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 3077: Pages 1-15.

[56] Labio, W.J. and Garcia-Molina, H.(1996) 'Efficient snapshot differential algorithms for data warehousing', In *Proceedings of VLDB Conference*, Pages 63-74.

[57] Lee, K., Choy, Y. and Cho, Sung-Bae. (2004) An Efficient Algorithm to Compute Differences between Structured Documents, *Journal IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 8, Pages 965-979.

[58] Leedy, P. D., and Ormrod, J. E. (2005) *Practical research planning and design.* Pearson Education Inc, New Jersey.

[59] Lenzerini, M. (2002) 'Data Integration: A Theoretical Perspective' In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Pages 233 - 246.

[60] Little, R.J.A. (1993) 'Statistical Analysis of Masked Data', *Journal of Official Statistics*, Vol.9, No.2, 1993. pp. 407426

[61] Lu, Z., Liu, H.J., and Hyland, P. (2012) 'Delta extraction optimization for view maintenance in a limited collaborative environment', In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications*, Pages 575 - 582.

[62] Lu, Z., Liu, H.J. and Yan, J. (2015) 'Delta extraction in a limited collaborative environment', *International Journal of Intelligent Information and Database Systems*, Vol. 9, No. 1, Pages 54 - 78.

[63] Lu, Z., Yan, J. and Wang, X.Q. (2015) 'Using Grouping Strategy and Pattern Discovery for Delta Extraction in a Limited Collaborative Environment', *International Journal of Business Intelligence and Data Mining*, Vol 10, No.4, Pages 378-405.

[64] MacHanavajjhala, A., Kifer, A. el al. (2008). 'Privacy: Theory meets Practice on the Map'. In *Proceedings of 'IEEE 24th International Conference on Data Engineering*, Pages 277286.

[65] Madhavan, J and Halevy, A. (2003) 'Composing mappings among data sources', *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, Pages 572-583.

[66] Magnani, L. (2001) Abduction, Reason, and Science: Processes of Discovery and Explanation, Kluwer Academic Plenum Publishers, New York.

[67] Mannila, H. and Toivonen, H. (1996) 'Discovering Generalized Episodes Using Minimal Occurrences', In *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Pages 146-151.

[68] Marian, A., Abiteboul, S., Cobena, G. et al. 'Change-centric management of versions in an XML warehouse.', In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, Pages 581-590.

[69] Ma, S. and Hellerstein. J. (2001) 'Mining partially periodic event patterns with unknown periods', In *Proceedings of 17th International Conference on Data Engineering (ICDE)*, Pages 205-214.

[70] Moro, G. and Sartori, C. (2001) 'Incremental maintenance of multi-source views', In *Proceedings of the 12th Australasian conference on Database technologies* (ADC), Pages 13-20.

[71] Motro, A. (1987) 'Superviews: Virtual integration of multiple databases', *Software Engineering, IEEE Transactions*, vol. 13, No. 7.

[72] Newell, A. and Simon, H. A. (1976) 'Computer Science as Empirical Inquiry: Symbols and Search'.*Communications of the ACM* Vol. 19, No. 3, Pages 113-126.

[73] Oracle Corporation. (2014) 'PL/SQL Packages and Types Reference', Release 12.1.

[74] Osterbye K., 'Structural and Cognitive Problems in Providing Version Control for Hypertext' *Proceedings of the ACM conference on Hypertext*, Pages 33-42.

[75] Ozden, B., Ramaswamy, S. and Silberschatz, A. (1998) 'Cyclic association rules', In *Proceedings of Data Engineering (ICDE)*, Pages 412-421.

[76] Ozsu, MT. and Valduriez, P. *Principles of distributed database systems*, Springer, New York, USA, 2011.

[77] Peirce, C. (1931-1958) 'Collected Papers of Charles Sanders Peirce', *Cambridge Press*, Vol.2, Pages 272-607.

[78] Rabl, T., Sadoghi, M. and Jacobsen, H. (2012) 'Solving big data challenges for enterprise application performance management', In *Proceedings of the 38th Conference on Very Large Databases (VLDB)*. Vol. 5, No. 12, Pages 1724-1735.

[79] Rahn, E. and Bernstein, P. A. (2001) 'A survey of approaches to automatic schema matching', *The VLDB Journal*, Vol. 10, No. 4, Pages 334-350.

[80] Rajaraman, A. (2001) 'Constructing Virtual Databases on the World-Wide Web'. PhD thesis, Stanford University.

[81] Ram, P. and Do, L. (2000) 'Extracting delta for incremental data warehouse maintenance', In *proceedings of the 16th International Conference on Data Engineering*, Pages 220 - 229.

[82] Rocha, R. L. A., Cardoso, L. F., Sourza, J. M. (2003) 'Performance tests in data warehousing ETLM process for detection of changes in data origin', In *Proceedings of the 5th Data Warehousing and Knowledge Discovery - Dawak Conference.*

[83] Schneier, B. *Applied Cryptography*, *Wiley*, 1996.

[84] Severance, DG. and Lohman, GM. (1976) 'Differential files: their application to the maintenance of large databases', *Journal of ACM Transactions on Database Systems (TODS)* , Vol. 1, No. 3, Pages 256-267.

[85] Samimi, P., and Patel, A. (2011) 'Review of pricing models for grid and cloud computing', In *Proceedings of Computers and Informatics, IEEE Symposium*, Pages 634-639.

[86] Sakr, S., Liu, A., Alomari, M. 'A Survey of Large Scale Data Management Approaches in Cloud Environments', *Communications Surveys & Tutorials, IEEE* , Vol. 13, No. 3, Pages 311 - 336.

[87] US secure hash algorithm 1 [online] URL. http://www.faqs.org/rfcs/rfc3174.html. 22, Feb 2013

[88] Stocker, K. and Kossmann, D. (2001) 'Integrating semijoin reducers into state-of-the-art query processors', In *Proceedings of the 17th International Conference on Data Engineering*, Pages 575-584.

[89] Thi, T., Helfert, M. et al. (2011) 'Discovering business rules from business process models', In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, Pages 259-265.

[90] Tridgell, A. (2000) Efficient Algorithms for Sorting and Synchronization, PhD thesis, The Australian National University.

[91] Tu, X, Litvak, E and Pagano, M (1994) On the informativeness and accuracy of pooled testing in estimating prevalence of a rare disease: application to HIV screening, *Oxford Journals of Science & Mathematics, Biometrika* Vol 82, No 2, Pages 287-297.

[92] Wang, Y., DeWitt, D. and Cai, J. (2003) 'X-Diff: an effective change detection algorithm for XML documents', In *Proceedings of 19th International Conference on Data Engineering*, Pages 519 - 530.

[93] Widom, J. (1995) 'Research problems in data warehousing', In *Proceedings of the 4th International Conference on Information and Knowledge Management*, Pages 25-30.

[94] Widom, J. (1996) 'Integrating heterogeneous databases: Lazy or eager?', *ACM Computing Surveys*, Vol. 28, No. 4.

[95] Widom, J. and Ceri, S. (1995) Active Database Systems: Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, 1996.

[96] Wong, E. and Youssefi, K. (1976) 'Decomposition - a strategy for query processing', *ACM Transactions on Database Systems*, Vol. 1, No.3.

[97] Yan, LL., Renee J. Miller, Laura M. Haas. el al. (2001) 'Data Driven Understanding and Refinement of Schema Mappings'. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Pages 485-496.

[98] Zhuge, Y., Garcia-Molina, H., Hammer, J. and Widom, J. (1995) 'View maintenance in a warehousing environment', In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Pages 316-327.

[99] Zhuge, Y., Hector Garcia-Molina, H. and Wiener, J. (1998) 'Consistency algorithms for multi-Source warehouse view maintenance', *Journal Distributed and Parallel Databases - Special issue on parallel and distributed information systems*, Vol. 6, No. 1, Pages 7-40.