

# Surrogate Key Generation through Multi-Agents

7-6536



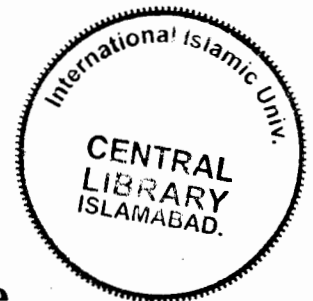
*Undertaken By:*

**Muhammad Nadeem Yasin**

**259-FAS/MSCS/F05**

*Supervised By:*

**Mr. Muhammad Imran Saeed**



**Department of Computer Science  
Faculty of Basic and Applied Sciences  
International Islamic University Islamabad  
2009**

A Thesis Submitted to the  
**Department of Computer Science**  
International Islamic University Islamabad  
as a partial fulfillment of requirements for the award of  
the degree of  
**MS in Computer Science**

**Dated:** ..28-01-2010

### **Final Approval**

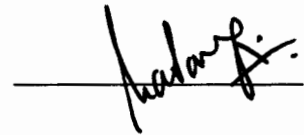
It is certified that we have examined the thesis titled “Surrogate Key Generation through Multi-Agents” submitted by Muhammad Nadeem Yasin, Registration No. 259-FAS/MSCS/F05, and found as per standard. In our judgment, this research project is sufficient to warrant it as acceptance by International Islamic University, Islamabad for the award of MS Degree in Computer Science.

### **Project Evaluation Committee**

**External Examiner:**

**Dr. Nazir Ahmad Sangi**

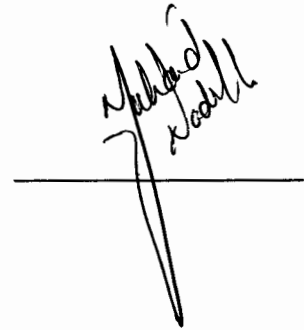
Chairman, Department of Computer Science  
Allama Iqbal Open University, Islamabad



**Internal Examiner:**

**Muhammad Nadeem**

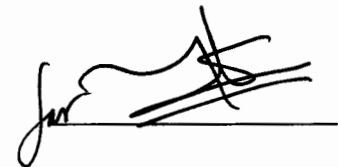
Assistant Professor  
Department of Computer Science  
Internatinal Islamic University, Islamabad



**Supervisor:**

**Muhammad Imran Saeed**

Assistant Professor  
Department of Computer Science  
International Islamic University, Islamabad.



**In the Name of ALLAH  
Who has all the names, and who does not need any name**

### **Declaration of Originality**

I hereby declare that this work, neither as a whole nor a part of it has been copied out from any source. It is further declared that I have developed the framework on the base of proposed model and the results with my personal efforts and under the sincere guidance of Muhammad Imran Saeed and Ainan Sadiq. If any part of this project is proved to be copied from any source or found to be reproduction of some other project, I shall stand by the consequences. No portion of the work presented in this dissertation has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Muhammad Nadeem Yasin  
(259-FAS/MSCS/F05)

## **Acknowledgement**

At very first, I bestow all praises, acclamation and appreciation to Almighty Allah, the most merciful and compassionate. The most Gracious and Beneficent, Whose bounteous blessings enable us to pursue and perceive higher ideals of life, All praises for His Holy Prophet Muhammad (SAW) Who enabled us to recognize our Lord and creator and brought to us a real source of knowledge from Allah, The Quran, Who is role model for us in every aspect of life. Secondly I must mention that it was mainly due to my family's moral and financial support during my entire academic career that enabled me to complete my work dedicatedly.

I am very thankful to my supervisor Muhammad Imran Saeed and friend Ainan Sadiq for their kind help and supervision. I shall remember their support for building my research capacity for research methodology and always guiding me to next bold step.

I would like to pay thanks to my class mates and faculty members of the university for their help and support, with special thanks to Mr Zeeshan Shafi Khan, Rashid Mehmood and Muhammad Shoaib.

Finally once again I would like to admit that I owe all my achievement to my parents who mean most to me, for their prayers which are more precious than any treasure on earth.

Muhammad Nadeem Yasin  
(259-FAS/MSCS/F05)

## **Project in Brief**

**Project Title:** Surrogate Key Generation through Multi-Agents

**Organization:** International Islamic University, Islamabad (IIUI).

**Undertaken by:** Muhammad Nadeem Yasin (259-FAS/MSCS/F05)

**Supervised by:** Muhammad Imran Saeed

**Starting Date:** August 2008

**Completion Date:** December 2009

**Tools Used:** Visual Studio Dot Net, 2008 (ASP.Net with C#)  
SQL Server 2005  
MS Access 2003  
SAGE (Agent Framework)

**Documentation Tools:** MS Word  
E-Draw

**Operating System:** Windows XP

**System Used:** Pentium IV (2.0 GHz, Dual Core)  
RAM 1 GB

## **ABSTRACT**

A central repository to store huge amount of data from multiple information based sources is known as data warehouse. The data in warehouse is transformed and formalized into single format for the analysis process. The primary reason for using data warehousing is to provide analytics result to businesses from data mining, OLAP, score carding and reporting. Sorting of the data warehouse records facilitate the searching and mining algorithms. The sorting can be performed either by using dimensional approach or through normalized approach. Since primary keys can not be applied on the data of heterogeneous databases so to uniquely identify a record from billions of heterogeneous records, data warehouse introduces the concept of surrogate keys. Researchers have proposed many algorithms to generate the surrogate keys but all of them have some lacking with respect to different parameters. Few techniques creates gaps among the records, few results in deadlock and hot spots, some of the techniques assign the same key to multiple records, some increases the searching and mining delay and few techniques generates the bigger keys hence increases the storage and processing load. We in this proposal introduced a new technique to generate the surrogate keys through software agents. The proposed technique will give better results in all the dimensions discussed above. It will avoid the deadlock, reduce the delay and size, eliminate the chances of duplication, and will support the searching and mining techniques. The most important advantage of the proposed approach is to trace the source system of the particular record as records come from heterogeneous environment.



## Table of Contents

<u>Contents</u>	<u>Page #</u>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Introduction to Databases .....	2
1.2 Overview of Data Warehouse .....	2
1.2.1 Data Warehouse Characteristics .....	3
1.2.2 Data Warehouse Architecture .....	4
1.2.3 Data Warehouse Benefits.....	5
1.2.4 Comparison of OLTP and Data Warehouse Systems.....	6
1.2.5 Extraction, Transformation and Load.....	6
1.2.6 Components of ETL.....	8
1.2.7 ETL Environment .....	9
1.2.8 ETL Issues .....	9
1.2.9 Common Operations on Data Loaded.....	10
1.2.10 Data Modeling .....	10
1.2.11 Data Warehousing Modeling.....	10
1.2.12 Dimensional Modeling.....	11
1.3 Surrogate Keys.....	12
1.3.1 Characteristics of Surrogate Keys.....	13
1.3.2 Reasons for using Surrogate Keys .....	14
1.3.3 Advantages of Surrogate Keys.....	15
1.3.4 Cost of Surrogate Keys .....	16
1.3.5 Surrogate Key Types.....	17
1.3.6 Surrogate Key Generation Stage.....	17
1.3.7 Surrogate Key Assignment Management .....	18
1.3.8 Visibility and Intelligence of Surrogate Keys.....	19
1.4 Problem Area .....	20
1.5 Objectives .....	20
1.6 Thesis Outline .....	20

<b>2 Literature Survey .....</b>	<b>22</b>
2.1 Related Research.....	23
2.1.1 Surrogate Key Generation Methods and Limitations .....	23
2.2 Gaps between Sequentially Structured Numbers.....	27
2.3 Software Agents.....	27
2.4 Summary .....	28
<b>3 Problem Analysis.....</b>	<b>29</b>
3.1 Introduction.....	30
3.2 Problem Scenarios .....	30
3.3 Focus of Research.....	31
3.4 Summary .....	32
<b>4 Proposed Solution.....</b>	<b>33</b>
4.1 Introduction.....	34
4.1.1 Agent Technology.....	34
4.1.2 Data Sharding.....	34
4.1.3 Data Loading.....	35
4.1.4 Increasing ETL Performance .....	36
4.1.5 Parallelism in ETL .....	37
4.2 Proposed Architecture.....	37
4.3 Methodology .....	38
4.4 Agent Architecture for Key Generation.....	39
4.5 Transformation Implementation .....	40
4.6 Fact Table Data Processing.....	41
4.7 Major Capabilities of Proposed Framework.....	42
4.8 Scope of Proposed Framework .....	42
4.9 Summary .....	42
<b>5 Implementation.....</b>	<b>44</b>
5.1 Introduction.....	45

---

5.1.1 Abstract Flow of Framework.....	45
5.1.2 Detailed Flow of Framework.....	46
5.2 Framework Implementation.....	47
5.2.1 Database Information.....	48
5.2.2 Screen Shots.....	49
5.2.2.1 Welcome Screen .....	49
5.2.2.2 Source & Destination Database Connections .....	49
5.2.2.3 ETL Manager .....	50
5.2.2.4 Agent Manger .....	52
5.2.2.5 Output Page (Fact Table).....	53
5.2.2.6 Output Page (Dimension Tables).....	54
5.2.2.7 Star Schema of Fact & Dimension Tables.....	55
5.2.2.8 Source Database with Table.....	56
5.2.2.9 Agent Manager Window in SAGE .....	57
5.2.2.10 Agent Actions Window in SAGE.....	58
5.2.2.11 Agent Creation Window in SAGE.....	59
5.2.3 Proposed Framework Flow .....	60
5.3 Summary.....	62
<b>6 Testing and Performance .....</b>	<b>63</b>
6.1 Test Scenarios .....	64
6.2 Testing and Results.....	64
6.3 Comparison of Keys Generation by Agents and Ordinary Methods .....	64
6.3.1 Source System Tracing .....	65
6.3.1.1 Example Source Tracing: Ordinary vs. Agents .....	65
6.3.2 Agent based Framework Performance.....	66
6.3.3 Database Volume vs. Response Time.....	67
6.3.4 Transactions Volume vs. Response Time.....	68
6.3.5 Concurrency Optimization.....	69
6.3.6 Reduced I/O Operations.....	72
6.3.7 Gaps between Sequentially Generated Numbers .....	73

---

6.3.7.1 Example: Gaps Generation .....	73
6.3.8 Replication .....	74
6.3.8.1 Example: Replication.....	74
6.4 Other Benefits .....	75
6.5 Summary .....	75
<b>7 Conclusion &amp; Outlook .....</b>	<b>77</b>
7.1 Conclusion .....	78
7.2 Contribution .....	78
7.3 Outlook .....	78
<b>8 Appendix</b>	
<b>9 References</b>	

## **List of Tables**

<b>Contents</b>	<b>Page #</b>
1.1 Comparison of OLTP and Data Warehouse System.....	6
1.2 Common Load Operations.....	10
1.3 Comparison of Surrogate Key Types.....	17
5.1 Database Information.....	48
6.1a Keys Generated by Ordinary Methods.....	65
6.1b Keys Generated by Multi Agents.....	65
6.2 Time Taken by each Method for ETL .....	66
6.3 Data Volume vs. Response Time.....	67
6.4 Transaction Volume vs. Response Time .....	68
6.5a Request per Sec. vs. User Load.....	70
6.5b Response Time vs. User Load .....	71
6.6 DBMS I/O Operations .....	72
6.7a Keys Generated by Ordinary Methods (with Gaps).....	73
6.7b Keys Generated by Multi Agents (without Gaps).....	74
6.8a Keys Generated by Ordinary Methods in Source .....	74
6.8b Keys Generated by Multi Agents in Replication .....	75

## List of Figures

<u>Contents</u>	<u>Page #</u>
1.1 Architecture of Data Warehouse.....	4
1.2 Detail of Warehouse Layer .....	5
1.3 ETL Process and Components.....	7
1.4 Hierarchy between Process Flow, Data Flow and Operations.....	8
1.5 The OLTP relational Database Model for Books .....	11
1.6 Surrogate Key Assignment .....	13
1.7 Data Warehousing Steps .....	15
1.8 Surrogate Key Generation.....	17
1.9 Sequence Generator in ETL.....	18
1.10 Dimension Cross Reference Table.....	19
1.11 Dimension Table Surrogate Key Management.....	19
3.1 Data Warehousing Steps .....	30
4.1 Data Sharding.....	35
4.2 ETL Data Loading .....	35
4.3 Data Loading with Staging Database.....	36
4.4 Proposed Architecture.....	38
4.5 Agent Architecture for Key Generation.....	39
4.6 Surrogate Key Assignment in Dimension.....	40
4.7 Transformation for Dimension Change .....	40
4.8 Fact Table Record Processing.....	41
5.1 Abstract Flow of Proposed Framework .....	45
5.2 Detail Flow of Proposed Framework.....	47
5.3 Welcome Screen .....	49
5.4 Connection Strings.....	50
5.5 ETL Manager .....	51
5.8 Agent Manage.....	52
5.9 Fact Table.....	53
5.10 Name Dimension Table .....	54
5.11 Address Dimension Table.....	54

5.12 Phone Dimension Table.....	55
5.13 Star Schema .....	55
5.14 Source Database.....	56
5.15 Source Table Data View .....	56
5.16 Agent Management in SAGE .....	57
5.17 Agent Actions in SAGE.....	58
5.18 Agent Creation in SAGE .....	59
5.19 Proposed Framework Flow Visualization.....	61
6.1 Time Taken by Each Method for ETL.....	67
6.2 Database Volume vs. Response Time.....	68
6.3 Transaction Growth vs. Response Time .....	69
6.4a Request per Sec. vs. User Load.....	70
6.4b Response Time vs. User Load .....	71
6.5 DBMS I/O Operations .....	72

# **CHAPTER 1**

## **INTRODUCTION**



## 1. Introduction to Databases

Database is a structured collection of logically related and shared data and its description, and is designed to get the desired information for multiple users in any organization. A database model is required to organize and structure the data and it also defines several operations to be operated on the collected and stored data. The most common and likely used database model relational model.

There are various database architectures including row-oriented and column-oriented architectures, depending upon nature of stored data, their use and end-user requirement. Different databases use single or a combination of architectures. Mostly row-oriented architecture is used in Online Transaction Processing system and column-oriented architecture is commonly used by data-warehouse systems.

Software named as Database Management System is used for storing, organizing and managing the data in a system. DBMS provides a gateway to users by which database can be defined, created and maintained and also the control to access the database. DBMS also involves with performance, recovery (from hardware failure), integrity and concurrency. DBMS is not dependent on the data model used but it can be categorized based on their supported database model. In Relational Database Management System, data is organized and managed by tables in which the relationships of related tables are represented by common values. Hence a relational model is implemented in RDBMS.

Tables are used in databases to organize the data having many rows and columns. A single database record is represented by a table row. Keys are used in database to keep the all records straight. A key is an attribute in database which is used to identify individual record, keeps the tables normalized, provide the index creation facility and/or sort the records in some manner. A key in a table having such properties is called Primary Key. One or more columns should be made as PK in each table for a good design. Foreign Keys are another type of keys used to cross-reference data between logically related tables.

### 1.2 Overview of Data Warehouse

IBM devised a solution for manipulating data stored in non-relational systems named as '*information warehouse*'. Unlike OLTP, data warehouse is relational database

which is not used for transaction processing but built for querying and analysis purposes and enables the knowledge workers like analyst, manager or executive for fast and better decision making [16]. Usually it consists of historical data extracted from one or more transactional systems. There is a separation between transaction workload and analysis workload in warehouse systems [62].

### 1.2.1 Data Warehouse Characteristics

A simple way of defining data warehousing is to describe the characteristics of a data warehouse as devised by Bill Inmon, who earned the title of ‘father of data warehousing’. The characteristics of DWH include [66]:

- Subject Oriented.
- Integrated.
- Time Variant.
- Nonvolatile.

By the definition of Bill Inmon, the data is **Subject-Oriented** as the WH is not structured around the major application areas such as stock control, inventory and invoicing but it covers the major subjects of the enterprise such as customers, products and sales. Simply the ability to define a DWH by subject matters, makes it subject oriented. The data is **Integrated** because it is coming in inconsistent from disparate sources and put into a consistent format so that a unified view of the data can be presented to the end-users. The data is **Time Variant** because as the historical data has to be maintained, so data in the WH is only valid over some time interval. The data is **Nonvolatile** because it should not change once entered. Data in WH is refreshed from operational systems on a regular basis rather than updating in real time. New or updated data is always supplementary integrated to the database, rather than replacing the existing one.

Instead of defining the DWH on the basis of characteristics of data, it can be defined according to processing associated with accessing the data from the source systems to the delivery of the data to the end-users. As defined by Ralph Kimball [40]:

“DWH is a system that extracts, cleans, transforms, and delivers source data into a dimensional data store and for decision making it also provides and implements querying and analysis mechanism”.

## 1.2.2 Data Warehouse Architecture

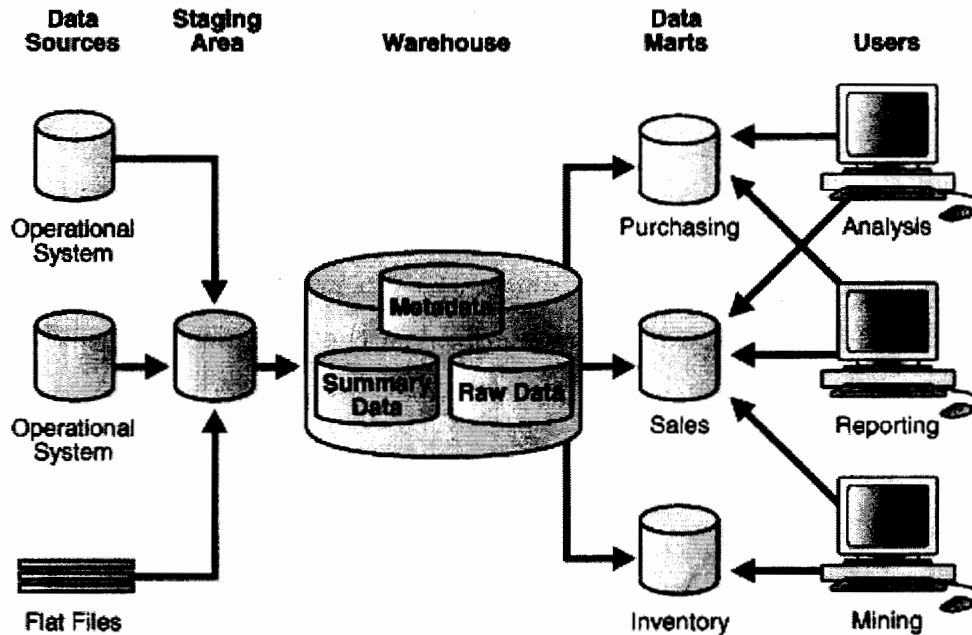


Fig 1.1: Architecture of a DWH [62]

Data Warehouse architecture can be divided into five layers, as **Data Source**, **Staging Area**, **Warehouse**, **Data Marts** and **Users**, based on the nature of job of each layer. Typically, data comes from one or more Operational Data Sources into WH or Operational Datastore (ODS) on a daily, weekly or monthly basis. Before adding data to DWH, it is normally processed into a **staging file** held in ODS. ODS is a repository of current, extracted, cleaned and integrated data ready for analysis purposes. The **Load Manager** present in WH Layer performs extraction and loading of data into the WH.

The **Warehouse Manager** performs the management operations on the data in the DWH. The operations include consistency assurance, transformation, merging and transferring of source data into DWH tables from temporary storage, indexes and views creation on base tables and archiving data. The **Query Manager** performs the management of user queries. All the detailed data in the database schema is stored in the

**Detail Data** of the Warehouse Layer, all the predefined lightly and highly summarized data generated by the warehouse manager is stored in **Lightly and Highly Summarized Data** area of Warehouse Layer. All the detailed and summarized data for the purpose of archiving is stored in **Archiving/Backup Data** area of the Warehouse Layer. All the meta-data definitions used by extraction and loading process, warehouse manager processes and query management processes in the warehouse are handled by **Meta Data** area.

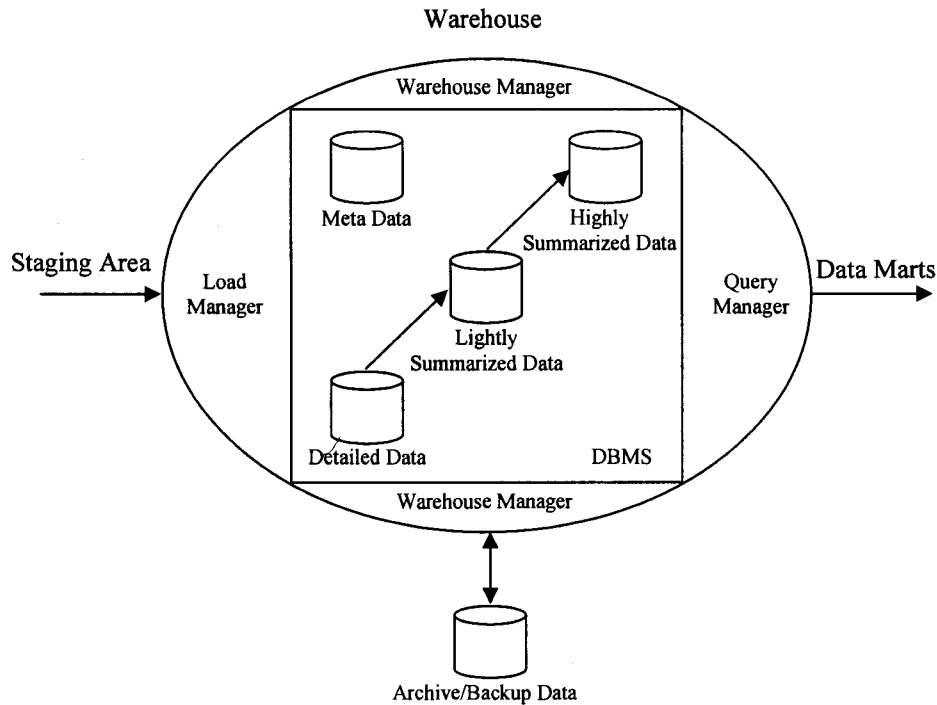


Fig 1.2: Detail of a Warehouse Layer [66]

### 1.2.3 Data Warehouse Benefits

A successful implementation of DW can bring various benefits to an organization as well as an individual including:

- DWH is very useful for trend reporting in decision support system applications.
- DWH is designed to perform well with aggregate queries running on large amounts usually terabytes of data.
- It decreases the workload on legacy systems by building and maintaining the complex queries.

- Reports based on data that is non uniform and scattered across variety of sources can efficiently be managed by WH Systems.
- DWH enable queries to cut across different segments e.g. Production data can be compared against Sale data even if they are different in structure and stored in different databases.
- Data warehousing provides the capability to store, manage and analyze large amounts of historical data.
- Potential high returns on investment, Competitive advantages to an organization can be achieved by the successful implementation of a DWH.
- By creating an integrated database of consistent, subject oriented, historical data, the productivity of corporate decisions makers can be improved.

#### 1.2.4 Comparison of OLTP and DWH Systems

OLTP and DWH systems are built for different purposes and have different requirements and characteristics. The comparison between these systems can be made by a table as [62][66]:

OLTP Systems	DWH Systems
Only support predefined operations	Accommodate ad hoc queries
Stores current data	Stores historical data
Holds detailed data	Holds detailed, lightly and highly summarized data
Updation through individual data modification statements	Updation through bulk data modification techniques by ETL.
Nature of data is dynamic	Mostly data is static
Use normalized schemas	Use denormalized schemas
Transaction Driven	Analysis Driven
Application Oriented	Subject Oriented
Useful for day-to-decisions	Useful for strategic decisions

**Table 1.1: Comparison of OLTP and DWH Systems**

#### 1.2.5 Extraction, Transformation and Load (ETL)

ETL consists of all the back-end processes required to make data ready for the end users to make business decisions. ETL includes collection of data from different

source systems, validation of it so that accuracy can be maintained, cleaning and transforming according to business rules to make it consistent, and eventually loading it in the WH where the users can query it [16][45][59][62].

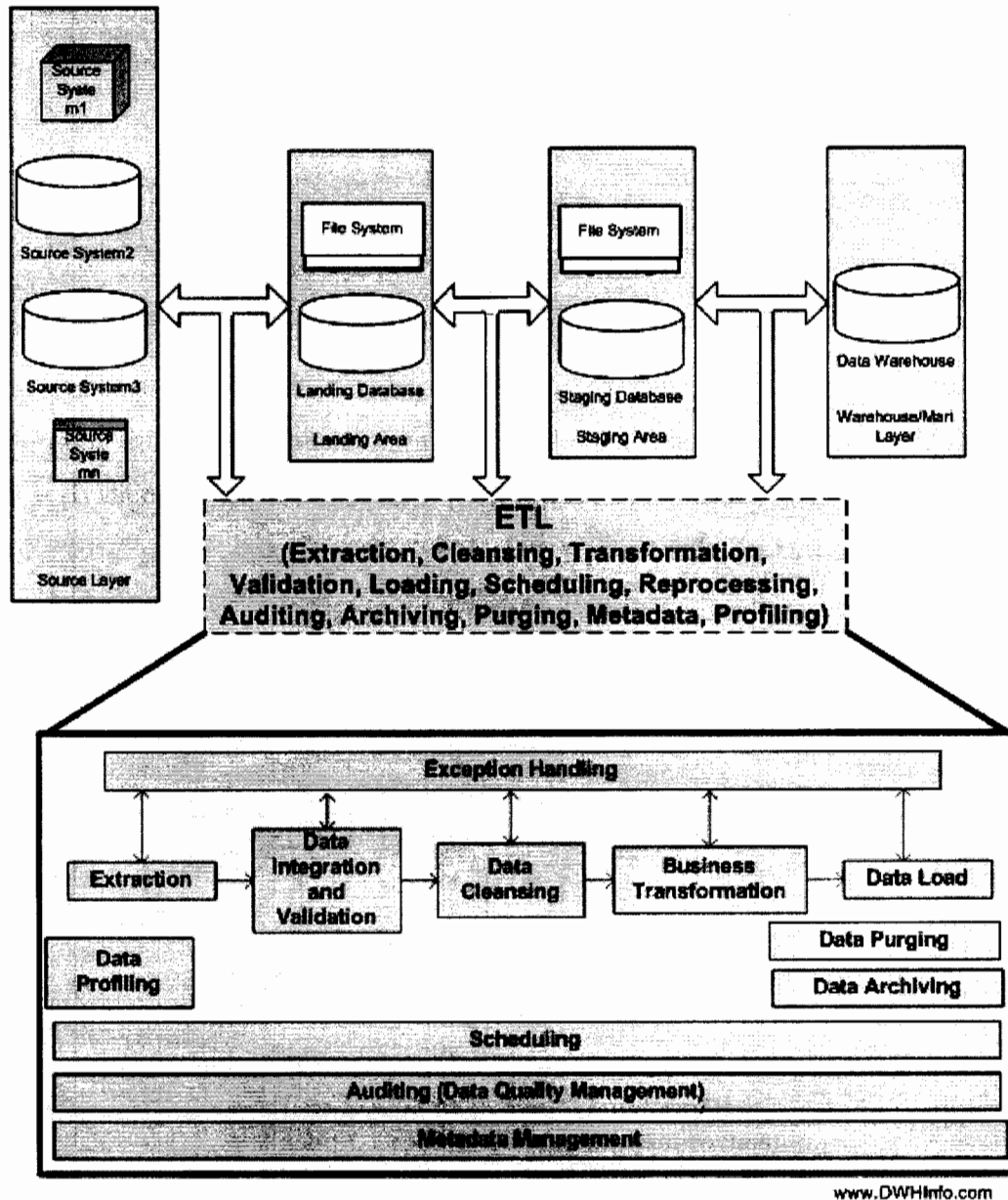


Fig 1.3: ETL Process and Components [45]

According to above figure, the ETL process starts from the source layer and ends on WH layer after passing through different stages. So, ETL can also be defined simply as "collection of all the processes involved in preparing and loading the data for users" [45].

The set of data in WH is called **Dataset** which can be persistent or runtime. Persistent datasets can be in the form of database or files data and Runtime datasets are the outcome of particular process. Each ETL process contains **Operations** which are usually parameterized and operate on the datasets. Operations can be grouped to make a **Data Flow** which performs a set of operations to provide functionality. Similarly a group of data flows is called **Process Flow** which provides a requirement of one subject area.

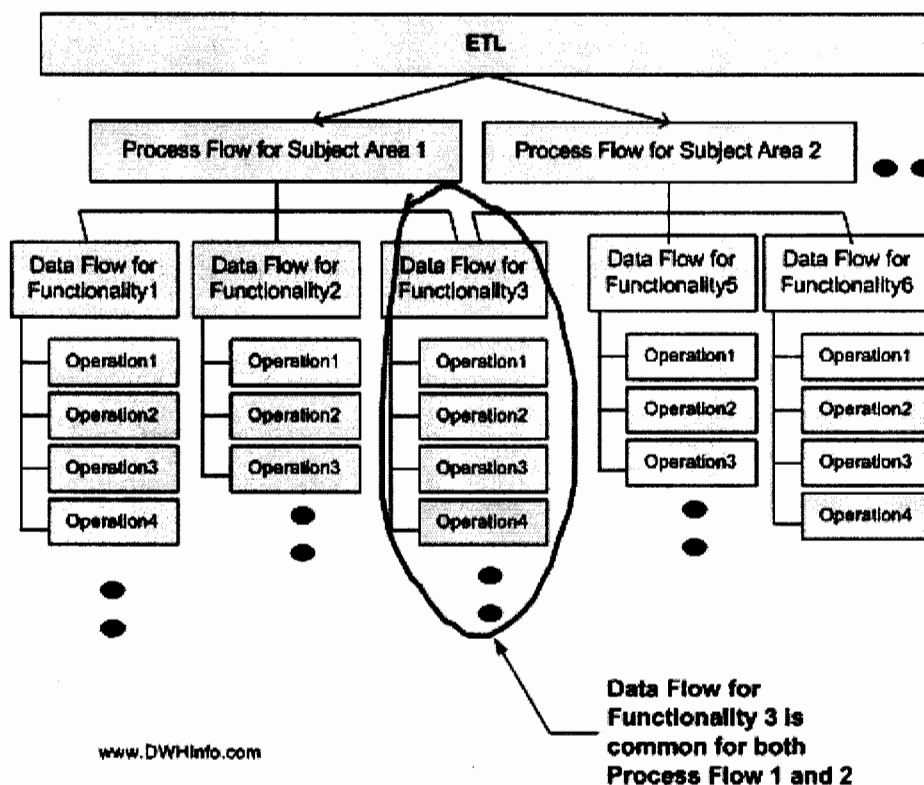


Fig 1.4: Hierarchy between process flow, data flow and operations [45]

### 1.2.6 Components of ETL

ETL consist of following components [45].

- Data Profiling.
- Data Extraction.
- Data Validation and Integration.
- Data Cleansing.
- Data Transformation.

- Exception Handling.
- Surrogate Key Generation.
- Data Load.
- ETL Scheduling.
- Data Archiving and Backup.
- Data Purging.
- ETL Auditing and Data Quality.
- ETL Metadata.

### 1.2.7 ETL Environment

The ETL environment consists of three architectures as data, application and technology and a group of people including developers, support and management [59].

Data and its quality, models to represent data, structure and business rules are included into the **Data Architecture**. Hardware like computers, networks and its all elements and Software like operating system and data management systems are included in **Technology Architecture**. **Application Architecture** includes ETL programs.

### 1.2.8 ETL Issues

We can categorize the ETL issues according to ETL Environment Architecture as [59]:

- Data Architecture Issues.
- Technology Architecture Issues.
- Application Architecture Issues.
- People Issues.

Data Architecture issues includes the dissimilarity of target and source system data structures, meta data, dependencies in the data, poor quality of source system data and complexity of source data. Technology Architecture issues includes less interpretability between platforms, improper scheduling, disk space and volume and frequency of loads.

Application Architecture Issues includes imprecise logging process, inappropriate error notifications and managing of start over and start from points in a



failure scenario. People Issues includes support, in-house expertise and management's comfort level with technology.

### 1.2.9 Common Operations on Data Loaded

There are many operations which are needed on the loaded data as described in following table [59]:

Surrogate Key Generation	Identity Column, Key Generator
Code Translation	If-then Logic, Lookup
Splitting Data	Bulk of Write Statements
Merging Data	Joining from Multiple Tables
Logging Error and Progress	Scheduling
Loading Code Tables	Set of Scripts

Table 1.2: Common Load Operations

### 1.2.10 Data Modeling

For building the complete DWH system, the first step is designing the data model for it. The primary objective of the data model is to confine the structure of data and its contents, and also the relationships between the data. Due to the granularity of the OLTP's relational database model, it does not cater the DWH requirements [31]. As DWH often uses column-oriented data store architecture so the model used for the data warehouse environment is dimensional model.

### 1.2.11 Data Warehouse Modeling

There is a sequence of steps in approaching data warehouse database design, beginning with the end-user prospects [29].

- **Subject Areas:** Establish the business subject areas.
- **Granularity:** Granularity is the level of detail required to satisfy the user queries. The recommendation is to include all historical data down to the lowest level of granularity. This ensures that any possible requirements in future for detailed analysis can always be met.
- **Building Dimensions:** Dimensions contain static information. Dimensions describe facts through storing static details about fact table transactions.

Dimensions are built before facts because fact tables contain foreign key references to dimension tables.

- **Building Facts:** Facts are transactional records. As facts are dependent on dimensions, so fact tables are created after all dimensions are finalized.

### 1.2.12 Dimensional Modeling

For data warehouse systems, neither normalized nor de-normalized relational database model is suitable. A quite different model is needed for data warehouse which is called a **dimensional database model**. A dimensional database model contains **dimensions** and **facts** and a standard framework called **star schema** is used to store the data. Dimension describes facts and fact table contains historical transactions, such as all orders placed by the customers for the last ten years. Dimensional model can be best described by the example.

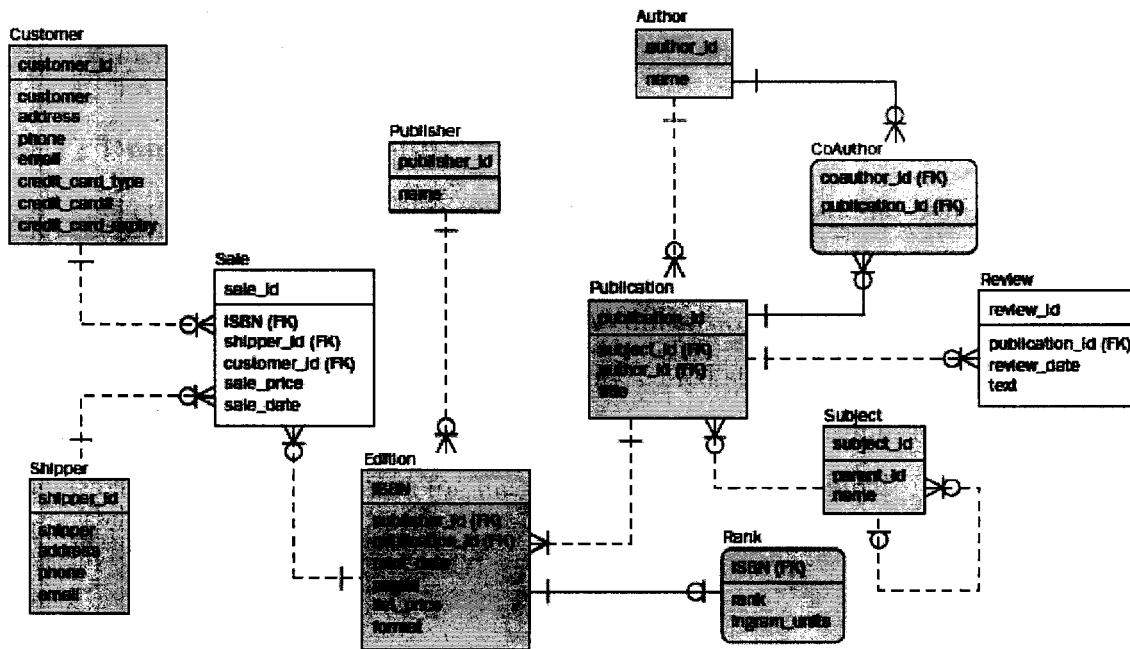


Fig 1.5: The OLTP relational database model for books [29].

Above figure shows a relational database model for static and dynamic book data. In figure the gray shaded tables are static whereas others are dynamic (in state of constant change) data tables. Dimensions are represented by static data tables and Facts

are represented by dynamic data tables. Star Schema contains centralized fact table that is joined to denormalized dimension tables.

The main characteristics of fact tables include normalized table, multipart primary key, numeric values, large number of rows, granularity level is same and not considered to be updated. The types of FT are as follows:

- Factless FT.
- Transaction FT.
- Periodic Snapshot FT.
- Accumulating Snapshot FT.

The main characteristics of dimension tables include denormalized table, a meaningless primary key, usually textual values and attribute values can change over a period of time [33]. The types of DT are as follows:

- Degenerate Dimension.
- Conformed Dimension.
- Slowly Changing Dimension.
  - Type 1 SCD.
  - Type 2 SCD.
  - Type 3 SCD.

### 1.3 Surrogate Keys

There are many dimensional modeling concepts and techniques that are critical to implement dimensional models. The key concepts are: surrogate keys and slowly changing dimensions.

Surrogate keys are keys that are created and maintained within the data warehouse instead of the taking natural keys of source systems. SK contains not any information and independent of data itself. These are used in place of heavy meaningful composite keys of the source systems due to performance reasons [9]. **Performance** and **Semantic Homogeneity** are the basic reasons for this replacement [16]. Surrogate keys are named by many other aliases, such as non-natural keys, system generated keys, dummy keys, meaningless keys, database sequence numbers, arbitrary unique identifier , artificial keys, non-intelligent keys, entity identifiers, synthetic keys, technical keys and

integer keys. Dimension tables should always be built with a surrogate key assigned by the ETL process. It is recommended that SK are created and used as the PK of all the dimension tables [8][37]. The surrogate keys joins the dimension tables to the fact table [19]. Surrogate keys serve as an important means of identifying each entity or instance inside a dimension table [11]. SK are not defined in Logical Model [25]. SK provides better performance with respect to joins due to their numeric nature [13].

Surrogate Keys and Primary Keys can be differentiated on the basis of the type of the database whether **Temporal** or **Current** database [63]. SK can be used as primary key in current database because it stores the data that is currently valid and the relationship between PK and SK is one-to-one. However the SK can not be used as primary key in temporal database because the correspondence between PK and SK is many-to-one. Hence there is a need of an attribute other than SK to uniquely identify any object in temporal database.

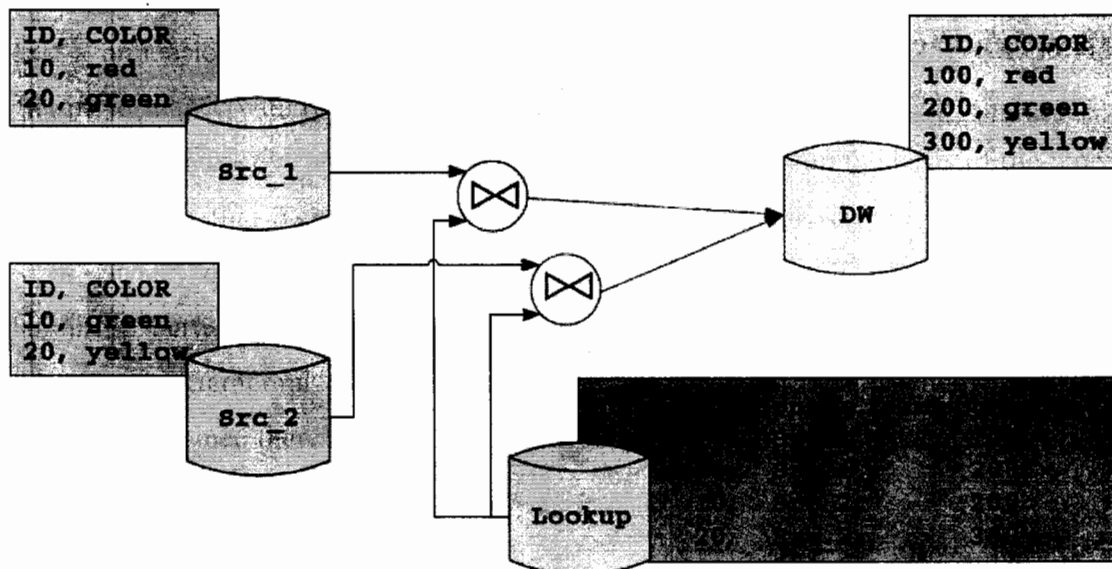


Fig 1.6: Surrogate Key Assignment [16]

### 1.3.1 Characteristics of Surrogate Keys

Surrogate keys has the following characteristics which make them perfect keys for the dimensions [30][63].

- Numeric in nature.
- System generated.

- System wide uniqueness.
- Unambiguous
- Stable i.e not changed.
- Invisible to business users.
- Single rather than Composite.
- No business meaning.

### 1.3.2 Reasons for Using Surrogate Keys

Primary reasons to use Surrogate Keys are as follows [12]:

- An entity can have different keys in different tables of different OLTPs and similarly a key is also used by different entity instances [38]. In other words reconciliation problems due to semantic homogeneity can be avoided [31].
- SK can preserve the order in which the rows are being inserted.
- Unlike integer keys, mostly string keys can cause overflow problems and also performance degradation.
- Surrogate keys are very useful to handle changes in dimension table. It maintains data warehouse information when dimensions are changed [8].
- Due to the uniform key structure it is very helpful in integration [30].
- Occasionally OLTP system keys may be reused due to absolute data or change in the source systems. However, the key may still be in use in historical data in the data warehouse, and the same key cannot be used to identify different entities [15].
- By the use of surrogate keys, performance of queries can be improved. The narrow integer surrogate keys mean a thinner fact table and due to thinner fact table, the performance can be increased.
- SK can be used to uniquely identify the billions of records of dimension tables.
- Surrogate keys also help to handle exceptional cases such as 'To Be Determined' or 'Not Applicable scenarios'.
- Due to the numeric nature of SK, they have better structure, more stable [24], give better control and have less storage.

- The join between fact and dimension tables can be simplified by using Surrogate Keys [26].
- The use of Surrogate Keys reduce the number of I/O operations in such a way that more rows of fact tables can be loaded into data pages of memory.
- Surrogate Keys enable database optimizations through bitmap indexing and star optimization.

Surrogate key generation is the one major step in getting the data from source system and delivering it to the target user and it takes a lot of time i.e 35% of overall data warehousing process.

STEP #	DESCRIPTION	TYPICAL SHARE OF TIME
1	Request a data package.	0%
2	Extract data from the source.	10% of the ETL process
3	Transformation rules/transfer to the PSA.	15% of the ETL process
4	Update rules.	40% of the ETL process
5	Surrogate key generation and database updates.	35% of the ETL process
6	Subsequent steps.	(varies)

Fig 1.7: Data Warehousing Steps [32]

### 1.3.3 Advantages of Surrogate Keys

Using surrogate keys in dimension tables provide some of the following benefits:

- Surrogate Keys replace the use of compound keys. Joins and maintenance of indexes for single column keys are very efficient [20].
- Surrogate keys have the same structure of data instead of different representation of same key in different OLTPs.
- Due to the compactness of SK, performance based on joins is very high.
- Surrogate Keys have storage and access independence [17].
- As their values and format of keys never change that is why surrogate keys as foreign keys are very stable [11].

- Surrogate Keys buffer the data warehouse environment from changes in the source system. In other words they protect the data warehouse system from changes in the source system. For example, a migration to a new software package in the source system will likely create a new set of keys [28].
- Surrogate keys allow the data warehouse system to integrate data from various source systems. Different source systems may keep data of the same products with different keys.
- Surrogate Keys are not tightly coupled with business therefore they are easy to maintain [7].
- Surrogate keys facilitate the user to add rows to dimensions that do not exist in the source system [24].
- Surrogate keys are used to keep track of changes in dimension attributes over time. Generally, it keeps the history of add new rows to the dimension table when an attribute changes. This means there will be various rows in the dimension table with the same source system key. The surrogate key provides the unique key for each row [40].
- With Surrogate Keys the Data Warehouse Load process can control the value assignment and ensure uniqueness because all keys have the same format.
- Integer surrogate keys are efficient keys and these are used to improve query and processing performance [41].
- The size of the fact table can be tiny due to the compactness of surrogate keys.
- The index on PK and FK is also compact due to compact nature of SK [30].
- Surrogate Keys are easy to handle for developers due to consistency of the keys.
- The use of SK also reduces the no of I/O operations [12].

The ability to track changes in dimension attributes over time is reason enough to implement surrogate keys.

### 1.3.4 Cost of Surrogate Keys

The most critical cost of using surrogate keys is that it places the burden on the ETL process [41]. Data normalization, query optimization, data disassociation, business process modeling [63] are also the cost factors. Other disadvantages of Surrogate keys

include extra space due to addition of a new column, hence extra index has to be built on SK column, difficulty in changing SK, data verification is not possible, as cluster index can give fast searching mechanism but cluster index on SK is useless and getting next value [9][21].

### 1.3.5 Surrogate Key Types

There are three types of surrogate keys as Concatenated Surrogate Keys, Semisurrogate Keys and Surrogate Keys [25]. Concatenated SK is multi-column compressed key with strings data type. Semisurrogate Key substitutes a portion of a multi-column concatenated key with a system generated numeric data. Finally Surrogate Keys are totally system generated numeric keys.

Factor	Normal	Concatenated	Semisurrogate	Surrogate
Integrity	Yes	Yes	Yes	Yes
Integration	None	None	None	None
Readability	Best	Ok	Compromise	None
Accessibility	Difficult	Easy	Moderate	Easy
Performance	Not Good	Better	Better	Best
Security	Almost None	Almost None	Better	Best

Table 1.3: Comparison of Surrogate Key Types [25]

### 1.3.6 Surrogate Key Generation Stage

Surrogate Keys are generated in the data-staging. Also the maps to the operational systems, current loads of operational data, and any atomic data not currently used in the data marts is stored in the Data Staging. Most of the heavy lifting performed by the ETL tools occurs here as well [30].

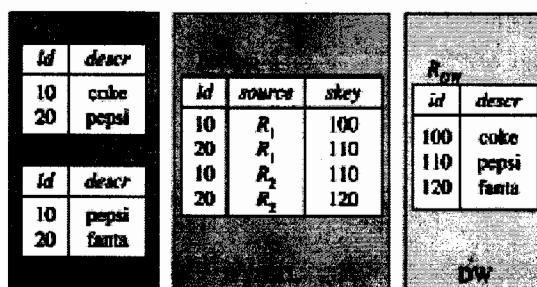


Fig 1.8: Surrogate Key Generation [18]



According to Vincent McBurney, there are two way to generate Surrogate Keys in the data load of ETL process [13].

- Surrogate Keys can be generated in the ETL job.
- Surrogate Keys can be generated in the target database.

SKs are made foreign keys in the fact table and hence the referential integrity in DWH can be ensured. The speed of data loading process can be increased by disabling redundant FK reference checking by the system. A row in the reference table is located with the help of natural key in order to assign a surrogate FK to incoming data and selecting the primary SK value. If no row is selected so we can either reject the incoming row or create a new row in the reference table from the natural key of the incoming data [30]. To generate SK for dimension tables, a Sequence Generator is used in ETL [45].

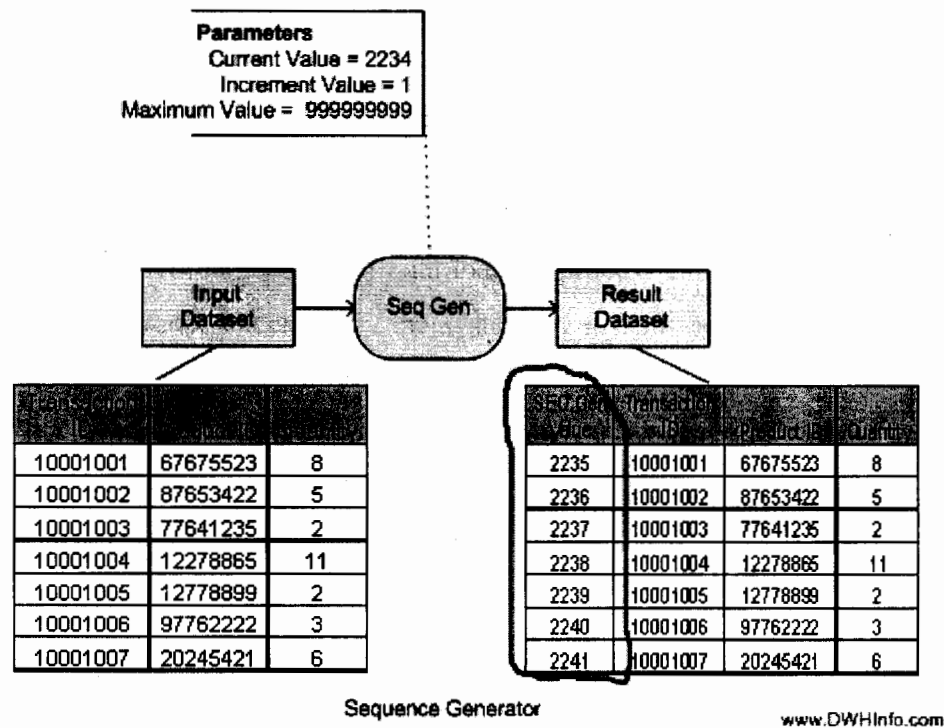


Fig 1.9: Sequence Generator in ETL [45]

### 1.3.7 Surrogate Key Assignment Management

As surrogate keys are generated for the dimension tables during the load process, a master cross-reference table for each dimension must also be maintained. This table

keeps the track that a particular SK is assigned against which business key along with time information.

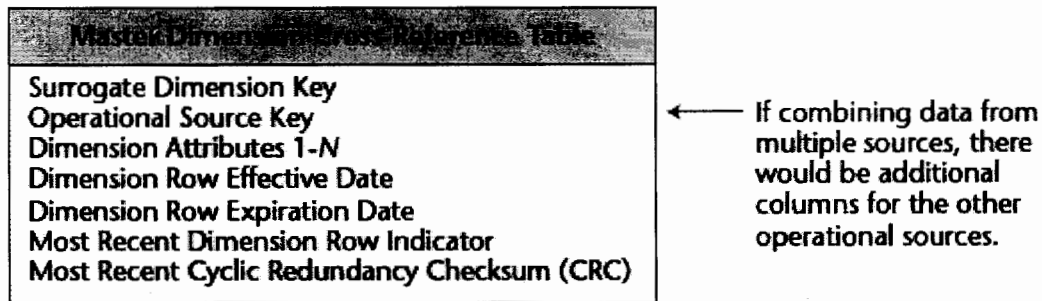


Fig 1.10: Dimension Cross Reference Table [40]

When a new dimension row has to be entered in the warehouse, then it is simple than managing the change in the existing row of the dimension table. In order to manage the change in dimension, we can follow the procedure as:

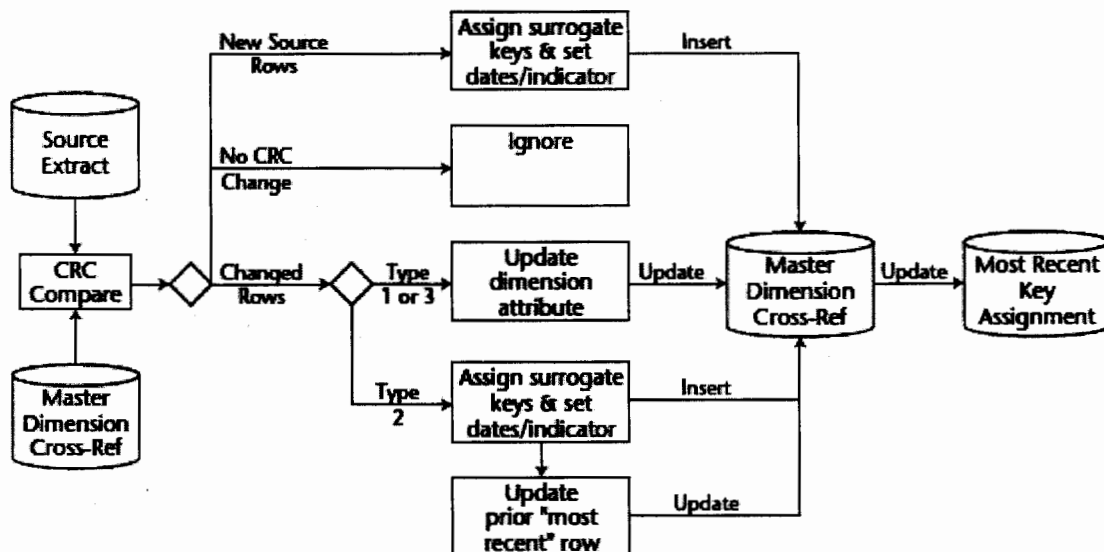


Fig 1.11: Dimension Table Surrogate Key Management [40]

### 1.3.8 Visibility and Intelligence of Surrogate Keys

Many data warehouse experts suggest that business code should be avoided in joining the dimension table with fact table. Hence meaningless numeric surrogate key should use in gluing between dimension and fact tables [40]. Also the intelligence should be avoided because it creates the dependency on business data. Similarly according to some experts, the surrogate keys should be hidden from the user based on several

reasons. First, if SK are visible, then user can understand it as business attribute. Second, the user may want to change its value according to system standard [24].

## 1.4 Problem Area

There are different methods to generate the surrogate keys each having some limitations. According to figure 1.7, 35% of time is consumed by SK generation in ETL process. So in order to accelerate the ETL process there is still a need of SK generation method having characteristics like reducing gaps in the records, solve the issue of concurrent access and deadlock, small in size to overcome storage overhead, ability to trace the source system of particular record and supports in replication.

## 1.5 Objectives

Our objective is to provide a framework to remove the maximum issues involved in existing methods of generating the surrogate keys. The core objectives of the framework are:

- To eliminate the bottleneck created due to the deadlock in processing huge amount of data (TB of data) in warehouse.
- To optimize the cost and performance by reducing the size of the surrogate key in fact table.
- To provide the surrogate key generation in an intensive concurrent environment.
- Avoiding hot spots created by concurrent transactions.
- To minimizing the number of I/O operations.
- To keep track of a particular record present in DWH in such a way that the user can tell the belonging of record to its source system.
- To minimize the load on DBMS in managing almost terabytes of data.

## 1.6 Thesis Outline

The rest of this thesis is organized as follows. In chapter 2 we have highlighted the previous work and their limitations in this domain. In chapter 3 we have reviewed the requirement analysis of our proposed framework. In chapter 4 we have discussed the designed model of our proposed approach. In chapter 5 we have discussed the

implementation issues. In chapter 6 we have evaluated our framework by showing the results taken from the software designed on the base of our proposed framework. In the last chapter we have described the conclusion and future work in this domain.

## **CHAPTER 2**

# **LITERATURE SURVEY**

Data Warehouse is a process for building an application architecture having a knowledge-base, a decision support systems and an environment which supports decision makers. DW is becoming a focus of the database industry because it is essential element of corporate strategizing and decision support. A lot of research has been done on the technical aspects and issues of data warehouse rather than focusing on its interaction with the organization.

Currently the main focus of DWH research is on back end processes, modeling of data, management of metadata, query processing, scalability, recoverability and front end process. Back end process includes extraction, cleansing, transformation and loading of data into WH. Front end processes consist of data analysis.

There are also a number of 3<sup>rd</sup> party tools which are developed to accelerate and optimize the overall data warehousing process.

## 2.1 Related Research

So far as our topic “Surrogate Key Generation through Multi-Agents” is concerned, we could not find any relevant material based on our approach. As the process of SK generation is done in loading phase of data warehousing, a lot of work has been done in data loading related issues along with SK generation. We have gone through many electronic articles and papers which show some work on SK generation methods with pros and cons and the requirement under which particular method is applicable. Following is the survey of the literature we did in our research.

### 2.1.1 Surrogate Generation Methods and Limitations

Joe Celko and Markus Zywitza have described the basic method to generate SK by using a **counter** [9]. This counter is maintained by DB engine and usually starts with one. A function is used to increment the value of counter locally by one [5]. In Sybase SQL the function NUMBER(\*) is the example of this method.

The basic problem with this method is that it can not be useful in multi client environment and also for clusters or an environment where data comes from heterogeneous sources. The second problem with this method is the overflow of number range. This method is also not useful under the circumstances where data from different

databases of organization have to merge. There can be clash of counter value that is an object represented in one database can be different in other database. So in order to gain consistency a lot of work has to be done.

Another method to generate SK is to use the Oracle “**Sequences**” [2][4]. It automatically generates a unique number which can be used as key of any kind of table. This is customized in such a way that user can set the incremented value between two numbers and also the limit of numbers. The main limitation of this technique is that it is only supported in Oracle.

An automatically generated value with “**IDENTITY**” keyword in SQL Server, Sybase can also be used for SK generation [10], as described by Gary Meyer. It is also customized as user can set the initial value and increment value. The value of identity column is sequentially unique numbers and generated on saving of record. A table can have one identity column [11]. The main advantages of using this method includes the keys are small, high performance on joining and indexing, debugging is easy and key updating is not allowed.

There are also some drawbacks of using this technique. The main drawback is that it is problematic in replication. Second, generated values can only be accessed after insertion. Other includes the gaps between generated numbers, uniqueness in a single table, not portable, creates insertion hot spot in concurrent environment, must declare as primary key to avoid duplication [49].

According to Scott W. Ambler the simplest solution to surrogate primary key generation is the “**Max Plus One**” [7]. In this strategy a column with numeric data and starting from one is used. On insertion of any new record, the maximum value on this column by using Max function is calculated and then one is added.

The main disadvantage of this approach is that it affects the performance when the size of the table is large as mostly in warehouse. Also it gives uniqueness within single table. It can be problematic after deletion of last row because its value can be reused for next record [34]. It causes serious problem in concurrent environment when concurrent transaction read the same value for the next record [42]. So this algorithm is best suitable for low intensity applications. The solution given by Mayer is “**Enhanced Max Plus One**” algorithm in which the deficiency of Max Plus One algorithm is

improved by including the keyword “**hold-lock**” in the query [10]. However, this algorithm also causes deadlocks in a highly concurrent environment.

Another approach to generate surrogate primary key is the “**Composite Key**” devised by Meyer [10]. In this concept two approaches are applicable including one column with multi part and other is combination of more than one columns. In first approach, the first part is the object id which controls the inserts into the table and the second part is datetime which ensures uniqueness. The main disadvantage of this approach is the size of the key and the other is, no mechanism to check and control the gaps generated between keys. This method also degrades the performance of DWH Process [30].

Joe Celko devised an approach to build a “**Separate Table**” that holds the next value to be generated. The table has single column usually integer type for that storing next value. Each time to generate SK a SP is invoked to get next value and then update that table [9]. The main advantage of this approach is the high portability [4].

This approach causes gaps between numbers when operated between concurrent transactions and also slows their processing by causing Hot-Block problem [29].

Another technique described by Roy Hann is use of a **Pseudo Random Number Generator** [15]. The values generated by this technique are the integers in random order not random number. The repetition of numbers does not appear for billions of values. A next seed table is used get the key value for the generator. This key value is hidden. The main advantage of this technique is that the keys are distributed uniformly. The other advantage of this technique is the security in sense that next value can not be predicted and it is also useful in concurrent environment.

The disadvantages of random surrogate keys include cyclicity in a sense it generates duplicates, size and complexity [9].

Surrogate keys can also be generated and inserted in the DW by **DB Triggers**. The main disadvantage of triggers is the severe bottleneck created in ETL [33].

According to Chuck Ballard and Daniel M. Farrell, **GUIDs** can also be used as surrogate keys [8][14]. GUIDs are Microsoft’s standard. The technique is hashing the current datetime with the MAC address of network card to produce the key value. In the absence of network card, the software id is used. The main advantage of this technique is



the uniquely generation of SKs over each table. It also allows the integration of data comes from different sources without any clash. It is helpful in replication. GUIDs increase the performance by reducing the joins [14]. It is highly portable [55] and hot spot can be avoided.

Some authors suggest to avoid the usage of GUIDs in DWH due to several reasons. The main reason is storage [37]. GUIDs takes a large amount of space then integers. The other reason is that as GUIDs are four times larger than integer key so indexes on GUID columns are slower integer keys. The process of debugging GUIDs is hard task and these are updateable [49].

UUID is another method to generate the surrogate key, described by Markus Zywitza and Scott W. Ambler [5][7]. Open Software Foundation create the algorithm for this approach. UUID is 128 bit value. This value is generated by a hash of the datetime of the currently used system and network card id. In the absence of card, an equal software representation is used. There are two representations in UUIDs, **hex** and **compact** [5]. In hex representation the id is converted and represented by hex values and string is made which is readable. In compact representation the id is converted into array of byte and then represented into string.

The disadvantage of this method is same as GUIDs. It includes large size, slower indexing and hard debugging.

According to Joseph Sack Surrogate Keys can be generated and managed by using *ROWGUIDCOL* property columns which provides the uniqueness at very high level [11]. This is very useful in heterogeneous environment. The main limitation of this technique is that a table can have only one rowguidcol column.

The integrity of data of column can be attained by the use of constrains in SQL Server as devised by Joseph Sack [11] and also can be used to generate SKs. It includes **Unique Constraint** and **Default Constraint**. Unique Constraint can be used to generate distinct values on non key columns and Default Constraint can be used on column where data type is not known. The major drawback of these techniques is the duplication in concurrent transaction environment when clock is set to backward.

Ralph Kimball and Joe Caserta have described another technique to generate the SK for DWH by concatenation of system's **natural key with datetime stamp** [33]. It also describes the insertion time of particular record.

There are various drawbacks of this technique. First is the dependency on source system. Secondly it is only supported in Homogeneous Environment. The major drawback is the less control, huge size and degraded query performance.

Scott W. Ambler describes another technique named High/Low strategy [7]. According to this strategy, any key is logically consists of two parts i.e. High and Low. High value comes from the source and Low is n digit value that is assigned by the application itself. The procedure is as on obtaining high value every time, the low value is set to zero. If the high value is 101 and  $n=3$  then the key will be 101000,101001, 101002 and so on. If the high get the value 201 and  $n= 3$  then the sequence will be 201000, 201001 and so on.

An approach which is useful in concurrent environment is the **Recycled Series** described by Gary Meyer [10]. In this approach every process in concurrent transaction environment has conceptually next key table with specific range. For example one process has its range from 101 to 1000 and other would have 1001 to 2000. This approach is most effective in insert intensive scenarios where cache hit ratio and concurrency can affect the performance.

## 2.2 Gaps between Sequentially Structured Numbers

There are four main reasons due to which gaps are generated between sequentially structured numbers according to DBMS support [48].

- Abortion of DML statements.
- Rolling back of any transaction.
- Seed value is reset.
- Rows deletion.

## 2.3 Software Agents

Software Agent is a stand alone computer based program works for predefined tasks in dynamic situation on behalf of user or any entity. It can work without any control

for an extended period [36][61]. In a Multi Agent System, there are more than one software agents and they can interact with each other in different scenarios including competitive, cooperative or autonomous. They can produce results for entities that initiated them and these entities can also terminate their instance. They can have a user interface [65]. A software agent can run manually by user in foreground or automatically in background. An agent call other agent, they are portable and can be replicated, serve for specific task. Other characteristics include persistence, reactive, social/collaborate and flexible. The application area of software agents includes data presentation, event notification, pattern recognition, data collection, data sorting and filtering, optimization and planning.

Agent Technology is used rapidly in data warehousing with growing size of DWH.

## 2.4 Summary

There are many techniques to generate a PK for a table in different DBMS and hence using for surrogate key generation for the dimension tables in DWH. Each technique has its own implication with pros and cons. Hence surrogate keys generation and management is still a problems for developers of DWH. Software agents are usually used for reporting purpose in DWH. By considering the application of software agents we are going to generate the surrogate keys through agents with various performance benefits.

# CHAPTER 3

## PROBLEM ANALYSIS

TH-6536

In this chapter we have analyzed the problems and limitations with existing methods to generate the SK with respect to their success. We have also discussed the major factors which are basis of our purposed work

### 3.1 Introduction

DWH contains huge volume of data this is stored on historical basis for analysis purposes for decision makers. The creation and maintenance of DWH is also a cumbersome job. It takes too time and resources of the software including DBMS as well as hardware resources including memory and processor. Collecting data from different source systems either homogeneous or heterogeneous environment, integrating them with some transformation and then loading them into the DWH tables is very complex, time taking, heavy and critical process. The whole process is called ETL and its various components are extraction, cleansing, transformation, profiling, SK generation, loading and other sub components which play supportive role with major components. We are concern with SK generation which is the major step in ETL process. It takes 35% of time of overall DW process [32].

STEP #	DESCRIPTION	TYPICAL SHARE OF TIME
1	Request a data package.	0%
2	Extract data from the source.	10% of the ETL process
3	Transformation rules/transfer to the PSA.	15% of the ETL process
4	Update rules.	40% of the ETL process
5	Surrogate key generation and database updates.	35% of the ETL process
6	Subsequent steps.	(varies)

Fig 3.1: Data Warehousing Steps [32]

### 3.2 Problem Scenarios

After conducting an extensive literature survey we conclude that all the current methods to generate the surrogate keys have some limitations. First of all few techniques generate gaps in the records listing, which results in confusions and de-organization of

data. Secondly the issue of concurrent access is not dealt efficiently in few of the techniques. Two concurrent records may obtain the same surrogate key. In few of the techniques the algorithms designed to handle the concurrency results in deadlock. The deadlock occurs due to the combination of shared lock and exclusive lock mechanism. So due to the chances of deadlock these techniques can only be efficient in low intensity environment. The size of the key is also one of the main issues that need to be dealt properly. In few techniques the size of the generated keys becomes bigger which indirectly increases the storage overhead and results in searching delays. Few techniques required the value of the next seed. If multiple transactions are allowed to access and retrieve the value of the next seed then multiple transactions can get same seed value and if only one transaction is allowed to get the value of the next seed then the delay arises.

As data come from different sources in DWH, there is no method available, to our best of knowledge, to keep the track of a record that it belongs to which system. The reason may be the transformation of data during ETL before loading of data into WH.

Methods which generate the SK automatically by DWH with DBMS supported type are considered best. The main problem with these types of method is the mismatch of values generated in replication and the originally in DWH.

The major cost of SK is that it places huge burden on the ETL system. First SK is assigned to each distinct dimension row. Secondly substitution is made on keys of transactional systems in fact table with SK from DT [41].

Till now, agents are used in DW system for reporting purposes including trend checking, event notification, pattern recognition etc. They can be used to maximize the performance of ETL process by sharing the load of processing and calculations.

### **3.3 Focus of Research**

As there are many components of ETL which are required to be considered for advancement but we have selected Surrogate Key Generation for our focus point. There are many techniques available but we are going to propose a framework to generate surrogate keys through multi agents. Now generation of unique number across the DT is the duty of agents rather than DBMS itself. Agents are invoked and terminated by the WH system. By using the advantages of multi agents, the limitations of prior method to

generate keys can be resolved and burden on ETL system can be reduced by dividing the task between agents. Tracing the source of the record present in DT is a major advantage of using this technique, which was not possible in prior methods (to the best of our knowledge).

### **3.4 Summary**

Present methods of SK generation have limitations including gaps between sequentially arranged numbers, concurrent access issues, deadlock occurrence, size of key, next seed problem and unknown ownership of records. To overcome these problems we have proposed a technique to generate SK by using multi agents and it will provide satisfactory performance optimization.

# **CHAPTER 4**

## **PROPOSED SOLUTION**



In this chapter we have given comprehensive discussion on our proposed framework designed for the purpose to bring into existence the surrogate key generation through multi agents concept.

## 4.1 Introduction

We start with discussing the base on which idea is inherited, SK generation stage, agent technology, proposed framework architecture in detail and consequences of solution.

### 4.1.1 Agent Technology

Software Agent is a stand alone computer based program works for predefined tasks in dynamic situation on behalf of user or any entity. It can work without any control for an extended period. In a Multi Agent System, there are more than one software agents and they can interact with each other in different scenarios including competitive, cooperative or autonomous. The major characteristics of agents include persistence, reactive, social/collaborate and flexible. The application area of software agents includes data presentation, event notification, pattern recognition, data collection, data sorting and filtering, optimization and planning.

Agent Technology is used rapidly in data warehousing with growing size of DWH. We have used multi agents for SK generation for the WH. They reside on separate layer and interact with WH in order to get datarow and then return the key for that row.

### 4.1.2 Data Sharding

Data warehousing is used very commonly for last few years due to size of application database and huge volume of transactions. Google engineers devised the term sharding. It is “*shared nothing*” partitioning approach in which huge volume of database is partitioned into smaller *shards* and then distributed over many servers for achieving scalability, throughput and performance increase [54].

The performance measures depend upon three components including disk I/O, memory and CPU. The basic database partitioning techniques are Master-Slave, Clustering Computing, Table Partitioning and Federated Tables. Each technique has its

own pros and cons but in a common each has dependency on shared resources and services. Database Sharding has mechanism for dividing database into smaller shards and distribute over independent servers having their own memory, disk and CPU. It provides faster, easy to manage and low cost solution for managing huge volume of database.

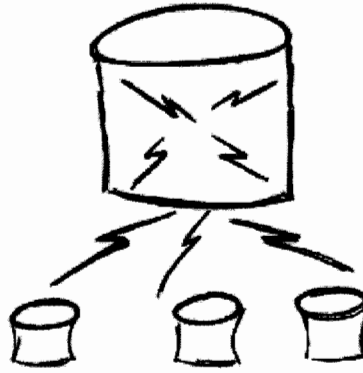


Fig 4.1: Data Sharding [54]

We have inherited the idea of database sharding with little change to our proposed solution. Instead of dividing whole database, its all related tasks and distributing it on other server, we distribute the task of surrogate key generation to multiple agents interacting with database application. The burden of generation and management of SK is now responsibility of agents. Agents are invoked and terminated by DB application itself. During the time to key generation DBMS can perform other task without depending on the return of generated keys.

### 4.1.3 Data Loading

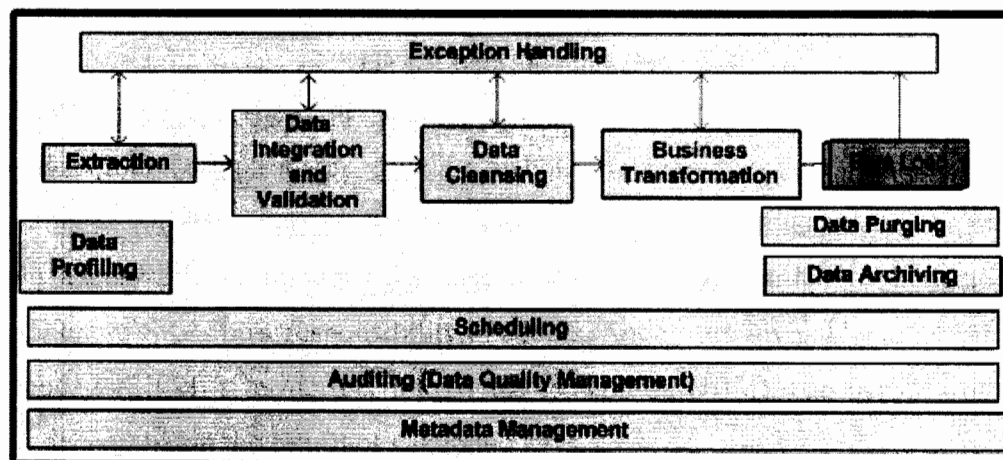
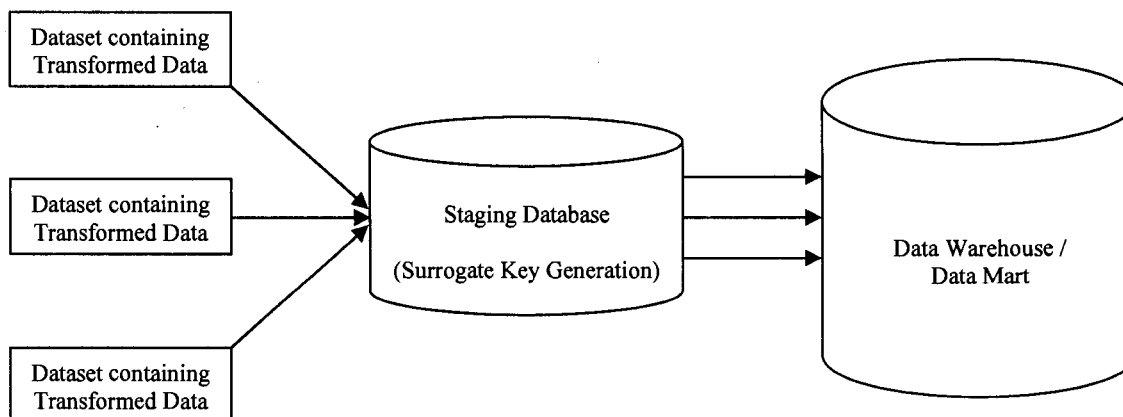


Fig 4.2: ETL, Data Loading [45]

Data Load is the process of loading the cleaned and transformed data to warehouse/mart where users can access it. The loading process is completed in two steps if DWH architecture has a staging database as:



**Fig 4.3: Data Loading with Staging Database**

- Loading transformed data to staging database.
- Loading data in staging database to DWH/data mart.

Insert, Update and Upsert operations are performed during loading the data into DWH. These operations can be performed by one by one row at a time but can create bottleneck. So a bulk load utility can be used. There are two types of Data Load including history load and incremental load [45].

In staging area agents are invoked to generate SK for DWH. Agents being held on separate layer, they can interact with DW system through this staging area.

#### 4.1.4 Increasing ETL Performance

In common practice, the data loading of ETL process is the slowest phase due to index creation, integrity maintenance and concurrent environment. So bulk loading operations can be used to accelerate ETL process but it can also create bottleneck during access to database. The ETL performance can be increased by partitioning table in small size and hence indexes, disabling validation, integrity checking and triggers during load process on target DB, generate ids in ETL instead of database, dropping index before load process and creating after load process in DWH, using bulk loading in parallel and minimizing dependency of jobs to each other [64].

Most authors suggest that ETL processing should be done outside the DB in order to get higher performance. For example, by using *distinct* we can remove duplicates from the table but it is slow in DB so we can perform it outside the DB. We have proposed the solution based on it with a change that key generation is done outside the DB and return back to it.

#### 4.1.5 Parallelism in ETL

The performance of ETL process can be improved by implementing parallelisms. It can be achieved in three ways by splitting a large table into small chunks for parallel accessing, by pipelining multiple components on same dataset and by pipelining several components on different dataset [63].

We in our proposed solution provide parallelism by dividing a data row from selected dataset into chunks according to subject area. Then agents automatically generate keys according to their subject based chunks and then loaded into the DW.

## 4.2 Proposed Architecture

We have proposed the said technique named **Generating Surrogate Keys through Multi Agent** after comparing all the prior methods. By using this technique cost, bottleneck, deadlock, hot spot and complexity can be reduced and performance can be maximized.

As data modeling for WH is dimension modeling and hence dimensions and fact tables are maintained within DWH. Surrogate keys are also called primary keys of DW. They are the part of the dimension table and the replacement of the keys coming from the source system, either from homogeneous or heterogeneous environment and also referenced into the fact table as foreign key. According to proposed architecture there is MAS layer in contact with staging area (staging database). Multiple agents reside on MAS layer. In staging area, cleaned and transformed data is stored for temporary basis and data is going to load into the different dimensions. As the numbers of dimensions to be made in DWH are known, so one software agent is reserved for each dimension in MAS layer to generate key for it. Agent will guarantee the generation of unique key

against each source system key. Each agent will add a new surrogate column to the data collected at staging database on staging area of ETL layer.

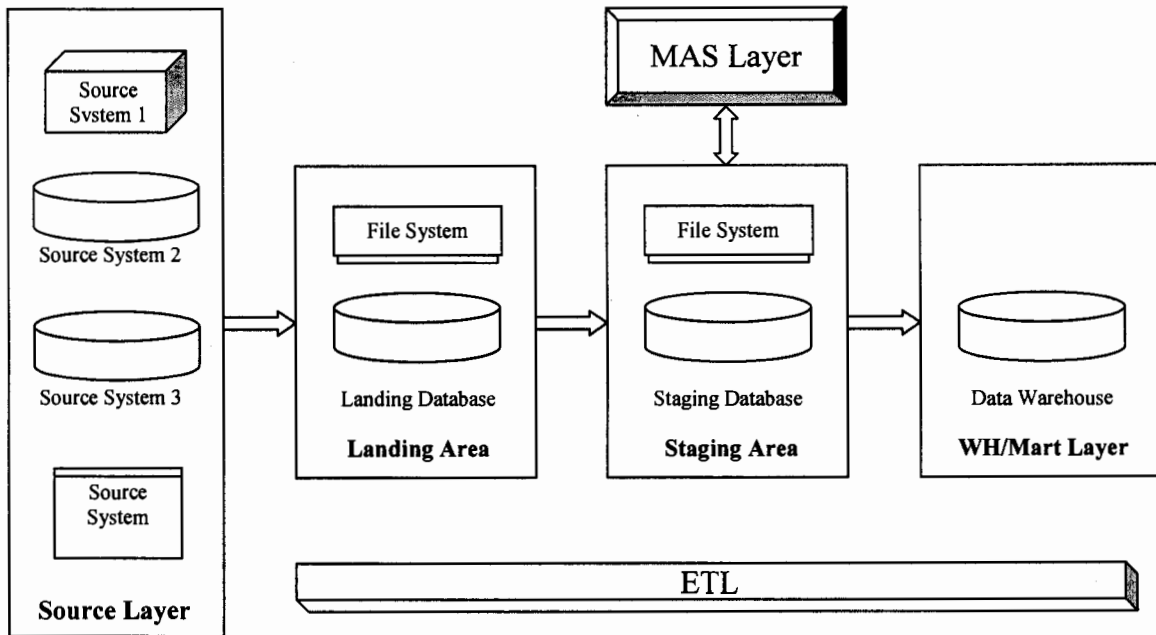


Fig 4.4: Proposed Architecture

### 4.3 Methodology

Information about each agent and its concerning is stored at central point in MAS layer. Information about agent includes agent id, unique id and start value where as it's concerning information includes source system id, database name, schema name, table name and the primary key field(s).

Surrogate Key is made by concatenating the unique id of particular agent with start value. Hence a unique number is generated. For example if an agent unique id is 100 and next value is 1 then the key will be 1001 and it will be the desired key for the dimension table. After generating the number it will return this number to the calling module in data staging area and then it immediately increments the start value by one for next generation of key.

Agent id differentiates each agent with other present in MAS layer. Unique id is the prefix used for keys which shows that this key is generated by which agent and hence

its dimension and source can be easily found. Start value is the seed value which will increment sequentially by agent.

#### 4.4 Agent Architecture for Key Generation

An agent is present against each dimension table which is responsible for the generation of the surrogate key against each record for that particular dimension. Being an autonomous system, agents are not part of the DBMS but can communicate with it. When the data is ready to be loaded into the dimension of the data warehouse, a request of surrogate key value is made against each record of the cleaned data resides in data staging area. Agents are constantly listening to the request for the next key value. After the request is made by ETL module, the particular agent according to the contents of data is invoked and key is returned.

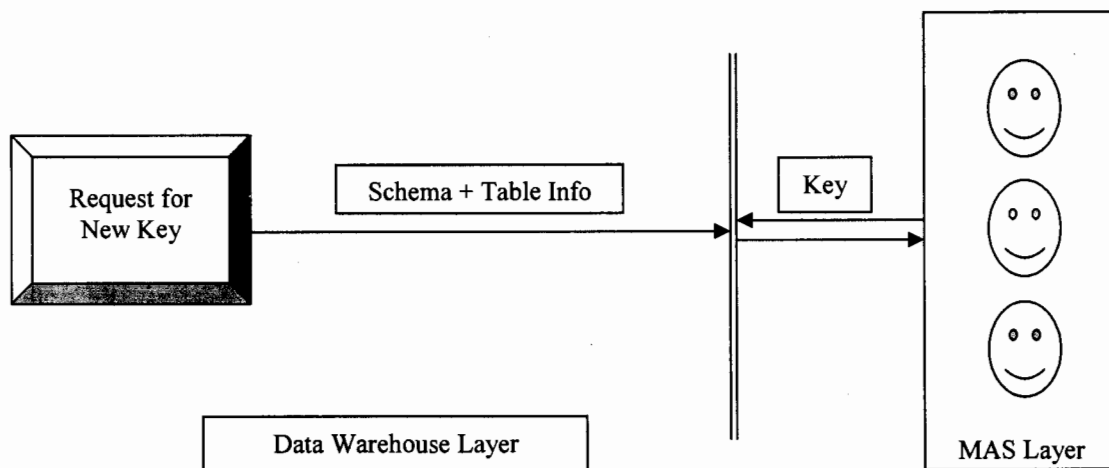


Fig 4.5: Agent Architecture for Key Generation

A new surrogate key must be assigned to every newly inserted dimensional record. In data warehouse two different SK can have same value but it does not make confusion for DWH because both belongs to separate dimensions [33]. But in our proposed technique this situation can not occur because each agent amends its id to the next number to be obtained. Hence, for two or more agents, generating same key for their dimensions is totally out of question.

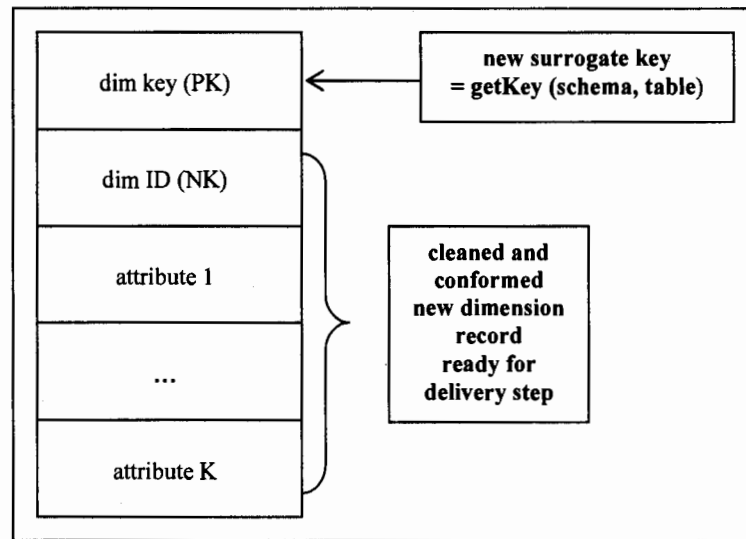


Fig 4.6: Surrogate Key assignment in dimension

## 4.5 Transformation Implementation

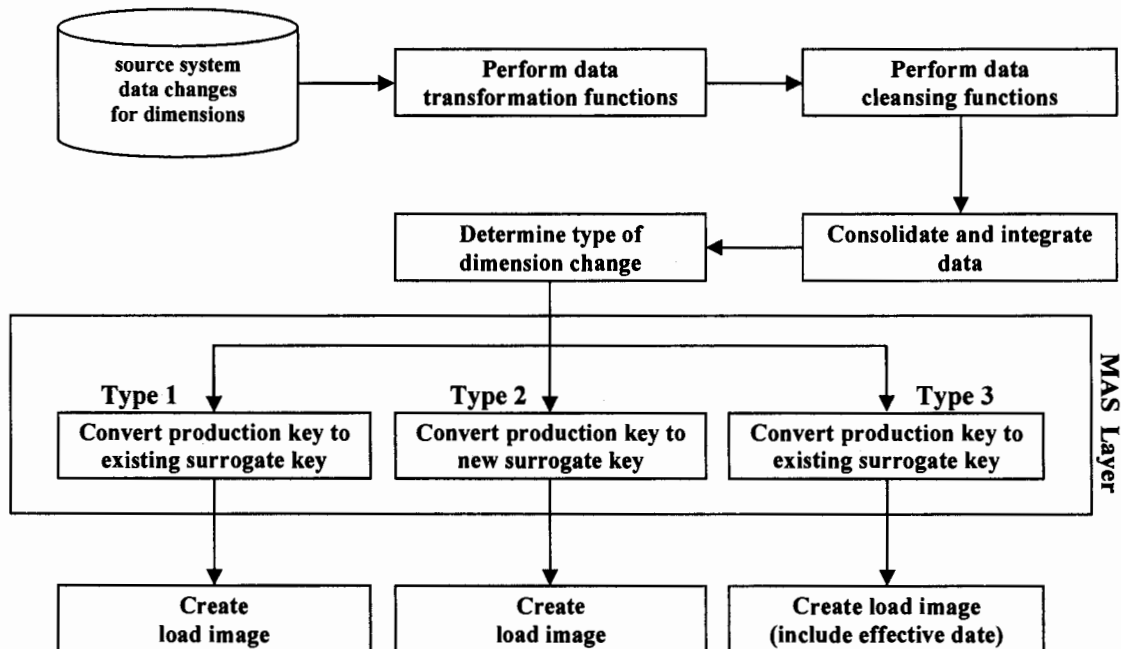


Fig 4.7: Transformation for dimension change

Instead of adding new rows into dimension tables, there is also a possibility that the data of dimensions may change. There are three options to handle change in dimension and these are called Type 1, Type 2 and Type3 changes. In Type 1, new values are

overwritten with old values by losing preserve history. In this type value of SKs do not change. In Type 2, as history is preserved so new values are assigned new surrogate keys and save as new dimension record. In type 3, current values are pushed to old value and new values are added in dimensions as current values. In this type value of SKs also do not change. Our proposed system also caters these types of changes.

## 4.6 Fact Table Data Processing

As primary key of dimensions are surrogate keys, so these are referenced in fact table as foreign keys. Hence prior to processing the fact tables, dimension table must be populated with new records. So in order to maintain referential integrity in DWH, this order must be followed but converse is not true except in deletion. As natural keys of the dimension are replaced with SK, similarly in order to process the FT, natural keys of dimensions are bring into fact table and then replaced with SK with the help of look up table. Our proposed architecture follows the same sequence.

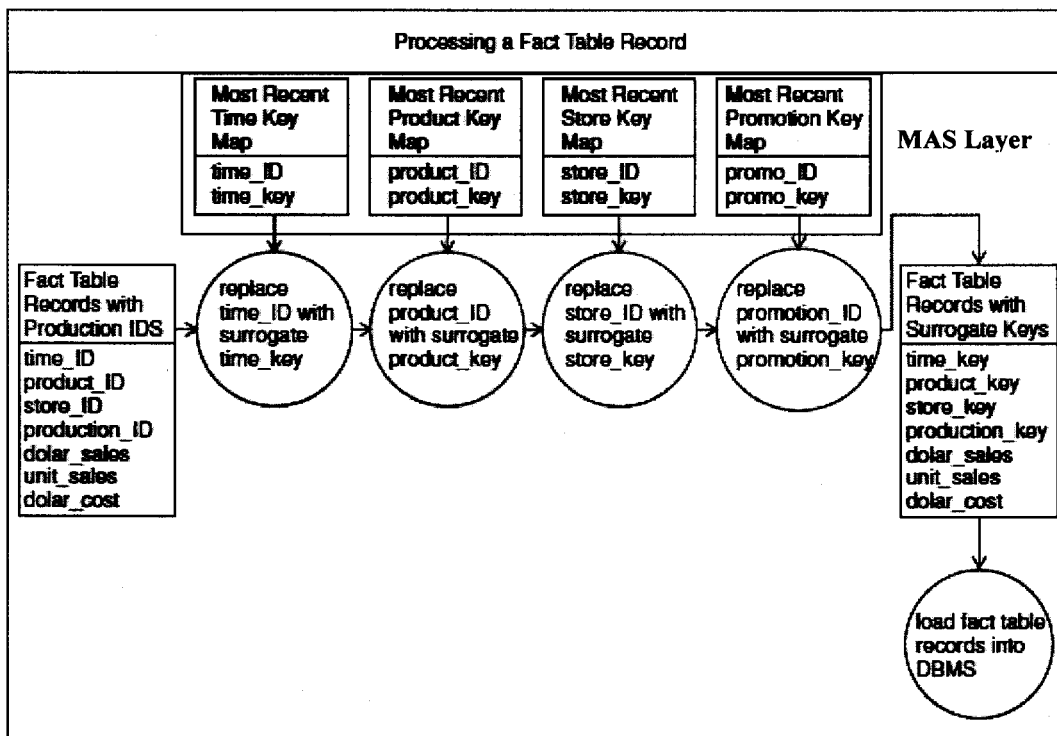


Fig 4.8: Fact table record processing [33]



## 4.7 Major Capabilities of Proposed Framework

The main purpose of our proposed framework is to remove the maximum issues involved in existing SK generation methods. Our framework will be distinguished from other methods in the following ways:

- It will provide the mechanism to trace the source of a particular record present in DWH.
- The resultant keys will be generated without gaps.
- It is also portable and support the replication and will give effective and accurate results.
- Cost and performance will be optimized due to reduced size of the surrogate keys in fact table.
- The bottleneck created due to the deadlock in processing huge amount of data (TB of data) in warehouse will be eliminated.
- It will support the surrogate key generation process in an intensive concurrent environment.
- Hot spots created by concurrent transactions will be eliminated.
- Number of I/O operations will be minimized..
- The load on ETL layer and DBMS in managing almost terabytes of data and keys will be reduced.

## 4.8 Scope of the Proposed Framework

Our research is focused on generation of surrogate keys with an affective way other than ordinary methods which will support DW building process as well as end user. Our proposed framework is capable of generating the unique gapless sequentially numeric surrogate keys.

## 4.9 Summary

We have introduced a new method to generate surrogate keys for dimension using multiple agents. All agents are grouped in multi agent system (MAS) layer which can interact with data staging area in ETL phase. The only purpose of agent is to generate surrogate keys and maintain referential integrity between dimensions and fact tables.

When data is going to load into dimension, a request is made for its WH key. This request is intercepted by particular agent present in MAS layer and returns the key. The proposed technique can also handle all three types of changes in data present in dimensions and fact.

# **CHAPTER 5**

## **IMPLEMENTATION**

In this chapter we have discussed the implementation detail of our proposed framework and its working in DWH environment. Other detail includes the description of development environment, demo application, flow of framework, output and visualization of key generation method.

## 5.1 Introduction

The approach is that an agent is present against each dimension table which is responsible for the generation of the surrogate key against each record for that particular dimension. There may more than one agent against one dimension in case of data present in staging area belongs to two or more sources. Being autonomous system, agents are not part of the DBMS but can communicate with it and each other also.

### 5.1.1 Abstract Flow of Framework

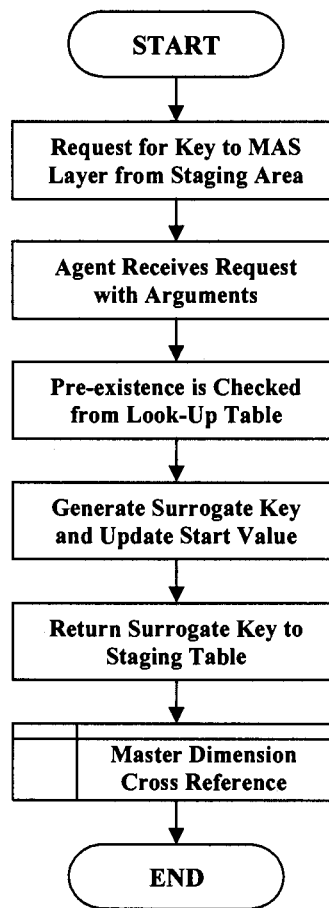


Fig 5.1: Abstract Flow of Proposed Framework

Figure 5.1 is the abstract level of flow of our agent based framework. When data is ready to be loaded into DWH at staging area after transformation, a request for key is initiated to MAS layer and data row is sent as argument against which key has to be generated. At MAS layer, this data is divided into chunks based on subjects/dimension and particular agent concerned with particular dimension generates key after checking either it is new or pre-existing row. If the data is new to DWH then key is returned to data staging area and agent increments the starting value to generate next key. Then it is loaded into dimension tables in DWH.

### 5.1.2 Detailed Flow of Framework

Figure 5.2 is the detailed level of flow of our agent based framework with some extension with figure 5.1. Pre-existence checking is the most critical part of the whole process. If Data coming from source system do not already exist in WH so process is simple inserting it into dimensions after key generation. But there is also a situation where data in dimension get change. There are three types of changes categorized as Type 1, Type 2 and Type 3. If change is checked in MAS Layer, then there are two scenarios.

In Type 1 or Type 3, new SK is not generated and designate scenario 1 whereas in Type 2, new SK is generated and designated as scenario 2. In Type 1, new values are overwritten with old values by losing preserve history. In this type value of SKs do not change. In Type 2, as history is preserved so new values are assigned new surrogate keys and save as new dimension record. In type 3, current values are pushed to old value and new values are added in dimensions as current values. In this type value of SKs also do not change.

In scenario 1, as new key is not generated so simply changed fields are updated by neglecting the history constraint and this change is synchronized with dimension table in DWH.

In scenario 2, new key is generated against change values by maintaining the history constraint. There are two tasks which have to be performed in scenario 2. First, generate new SK for the changed values and insert into dimension. Second, prior most recent values are updated into the dimension.

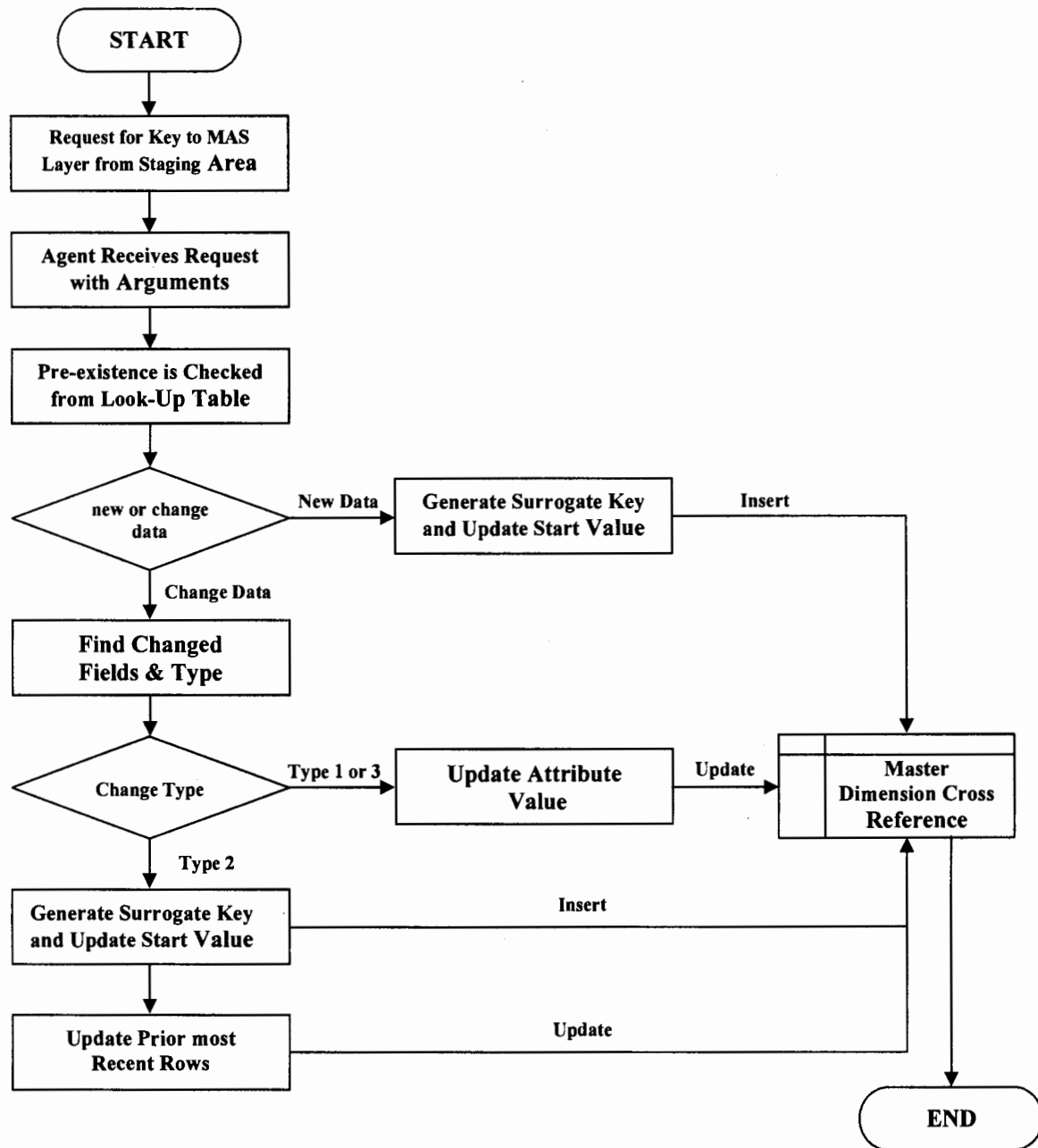


Fig 5.2: Detailed Flow of proposed framework

## 5.2 Framework Implementation

We have developed an application based on framework in Visual Studio .Net 2008 with SQL Server 2005 as the DBMS. Our Demo source data is the PTCL Telephone Directory which is created in MS Access 2003 Database. Our application is basically web

based and agent framework used for this project is "SAGE" which is open source created in JAVA. SAGE has its own GUI for creating and managing agents. Instead of running separate application for SAGE, we have inherited the Agent creation and managing which includes killing, suspending, resuming and sending messages process in our application and made our own module i.e. it will execute the "exe" files of each agent.

This application contains the coding of Complete **ETL Process** i.e. Extraction of data from source systems, Cleaning of data, Transformation of data and Loading of data into data warehouse along with surrogate keys generated by software agents.

### 5.2.1 Database Information:

The information about source system DBMS and data warehouse DBMS is as follows:

Source Database File Name	DWHProject2008.mdb (MS Access 2003)
Total No. of Tables	1
Table Name	51
Total No. of Records	253077
Destination Database Name	ETLDestination
Total No. of Tables	5 (1+3+1)
	1 – Table to store Cleaned Data
	3 – Dimension Tables
	1 – Fact Table
Table Name	tblPTCLcleanData (stores cleaned data)
Dimension Tables	DimtblName, DimtblAddress, DimtblPhone
Fact Table	tblFact
Selected No. of Records for Data Warehouse	1251
"conn-type=07"	
No. of cleaned records saved in destination	1251
"tblPTCLCleanData"	
No. of Distinct records for Name Dimension	1154
No. of Distinct records for Address Dimension	324
No. of Distinct records for Phone Dimension	1251

**Table 5.1: Database Information**

## 5.2.2 Screenshots

Now we will present the screenshots of each module involve in data warehousing with our proposed framework with detail description.

### 5.2.2.1 Welcome Screen

This is the 1<sup>st</sup> screen of our application. This page is loaded after checking the availability of source and destination databases and their connection with our application. Agents are not loaded here. The page has link button to navigate to the next page.



Fig 5.3: Welcome Screen

### 5.2.2.2 Source & Destination Database Connections

This page shows the connection strings of source and destination databases. We can get the system/server name and database name from connection string. As this is web based application, so have stored the source system and database name and also destination system and database name into separate session variables. As session



variables are global variables and have application wide visibility, we do not need to send this information as parameter to agent in key generation process. The page has also a link button to navigate to the next page.

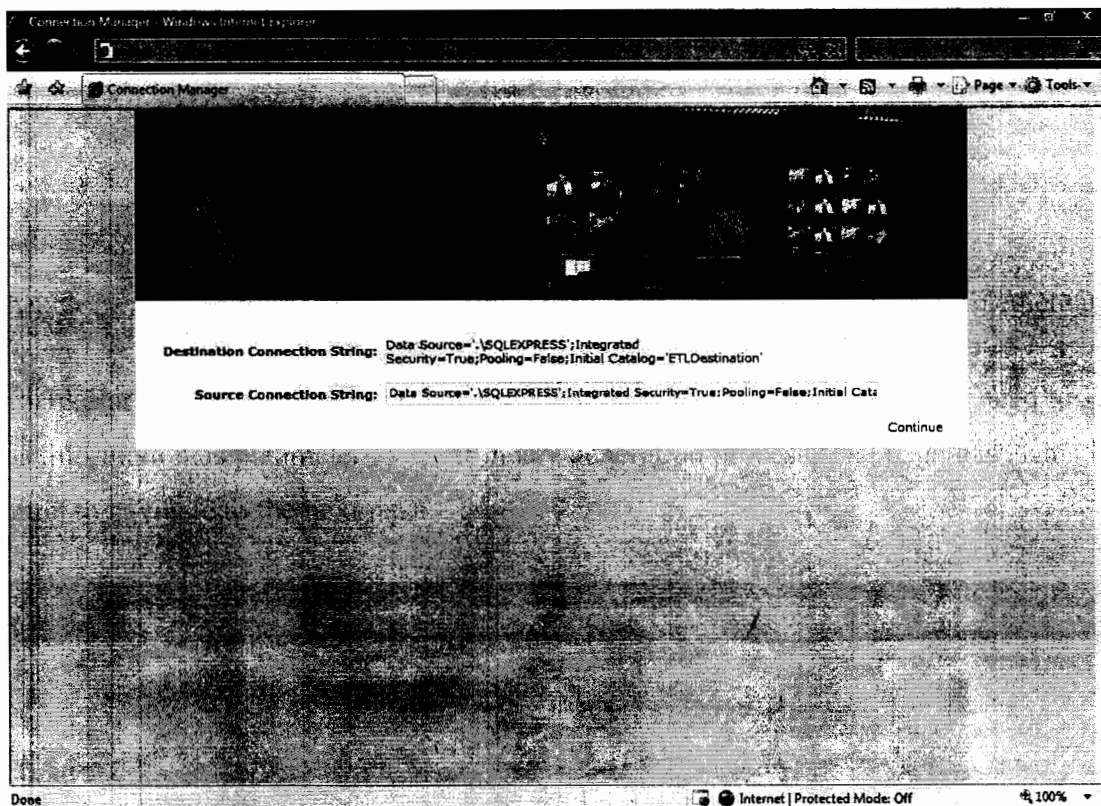


Fig 5.4: Connection Strings

### 5.2.2.3 ETL Manager

This is the main window of our application. Our proposed framework is incorporated in this module. The page has 3 command buttons. "**Manage Agents**" link button is used to redirect to Agent Manager Page where one can create and manage (edit, delete) agents. "**Start**" button is used to start the ETL process including Agent loading process. "**Stop**" button is used to stop the agents which are loaded into the memory. There three check boxes showing the ETL progress i.e. extraction, transformation and loading respectively.

As we have made ETL module in our application, so we have divide ETL manager module into three sub modules. Each sub module is represented by check box

on the screen. Initially start button is enabled and stop button is disabled and agents are not invoked. When we press start button, it gets disabled and stop button gets enabled. Also agents are invoked and activate in MAS Layer and load into memory. Also ETL process starts by executing the load sub module.

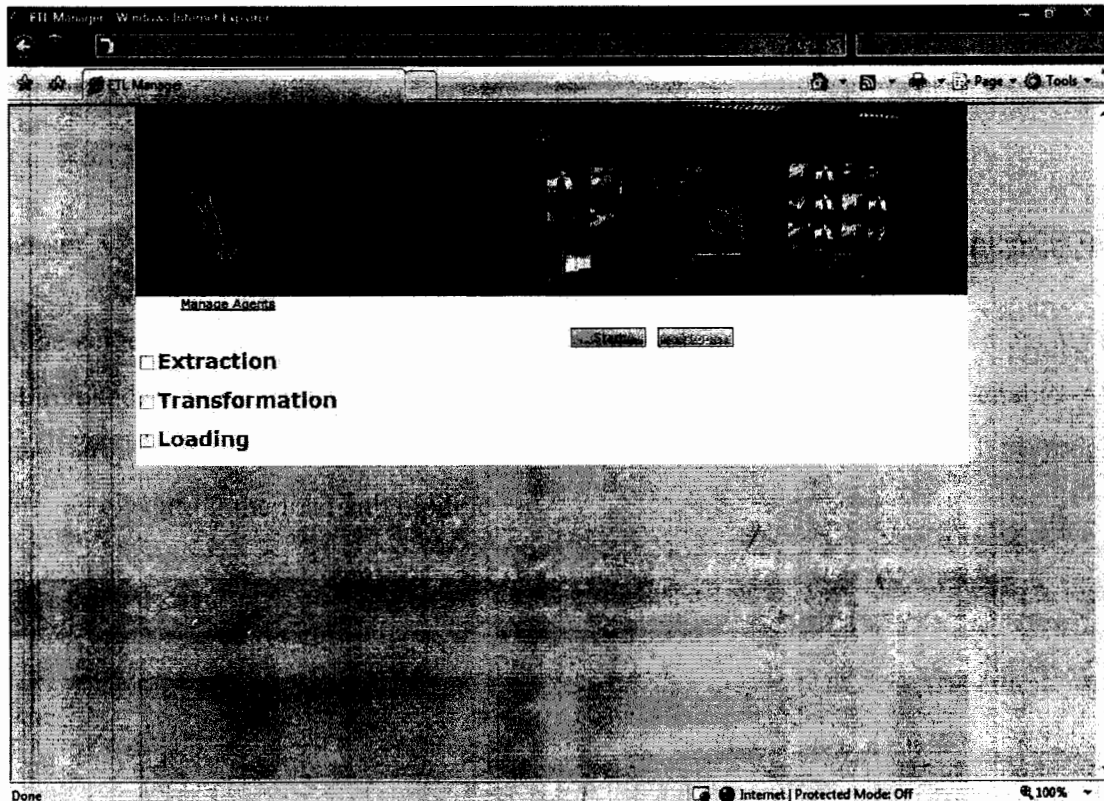


Fig 5.5: ETL Manager

When the extraction process is completed, then check box on the screen is marked with checked and next sub module i.e. transformation starts. We have created a temporary table to store extracted data from source system in our demo application. During transformation sub module, data is cleaned, inconsistencies are removed and transformations are made. In our demo application, source system table had one table with four columns (conn\_type, name, address, phone). After transformation, the temporary table in DWH has twelve columns (conn\_type, title, first name, middle name, last name, house no, street no, sector, city, isResident, phone no, description). The major operations involved in transformation sub module are string manipulation and use of regular expressions for patterns. After completion of transformation, the check box on screen is marked checked and next sub module starts i.e. loading. This is the major part

of our application where agents come into action and actual loading of data into the warehouse dimension take place. Surrogate Keys generation is done in this process.

As in transformation all the cleaned data is placed into temporary table, then in loading process it is moved into actual DWH dimensions. SKs are generated against each distinct row of this table. Each row is sent to MAS layer for key generation. This row id divided into subject area and an agent concerned with that subject generate key if its record does not already exist. Then it is loaded into its subjected dimension along with surrogate key. After the completion of load process, its checkbox is marked with checked and result page appears showing data of dimension(s) and fact tables.

Execution of ETL module can be interrupted and terminated by pressing stop button. On pressing stop button, it gets disable and agents loaded into the memory are killed and release the memory and start button gets enable and any change made to temporary table is rolled back.

#### 5.2.2.4 Agent Manager

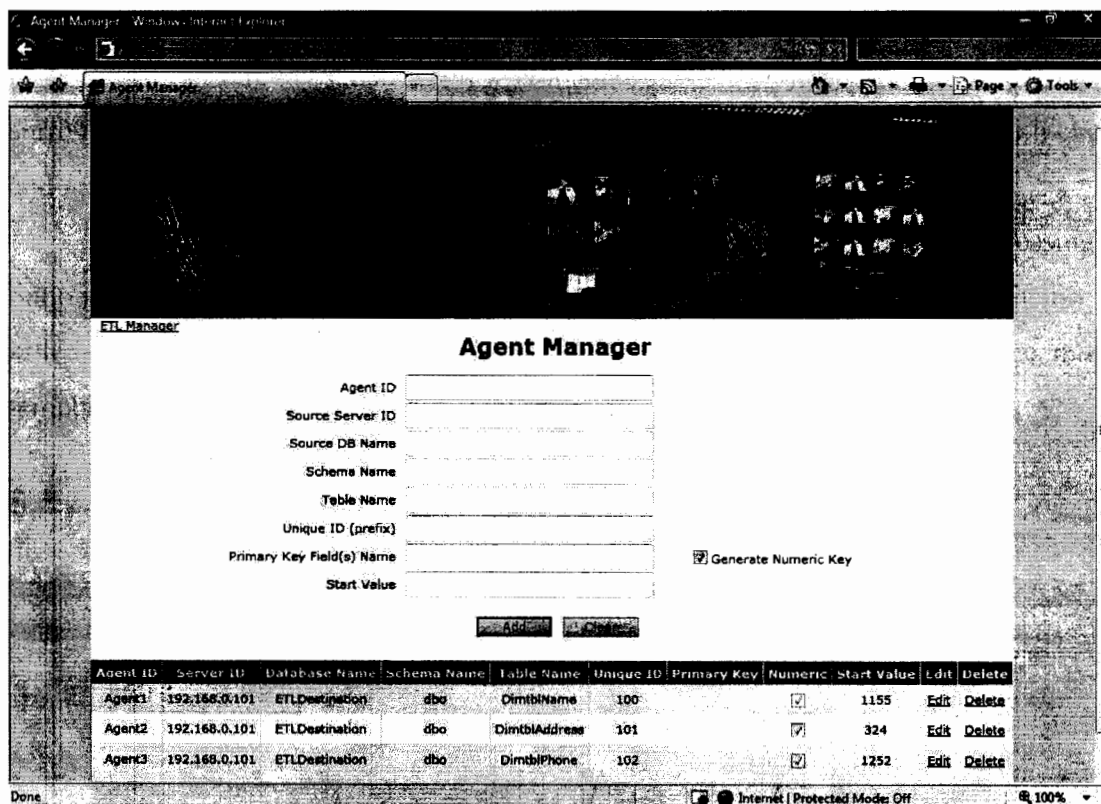
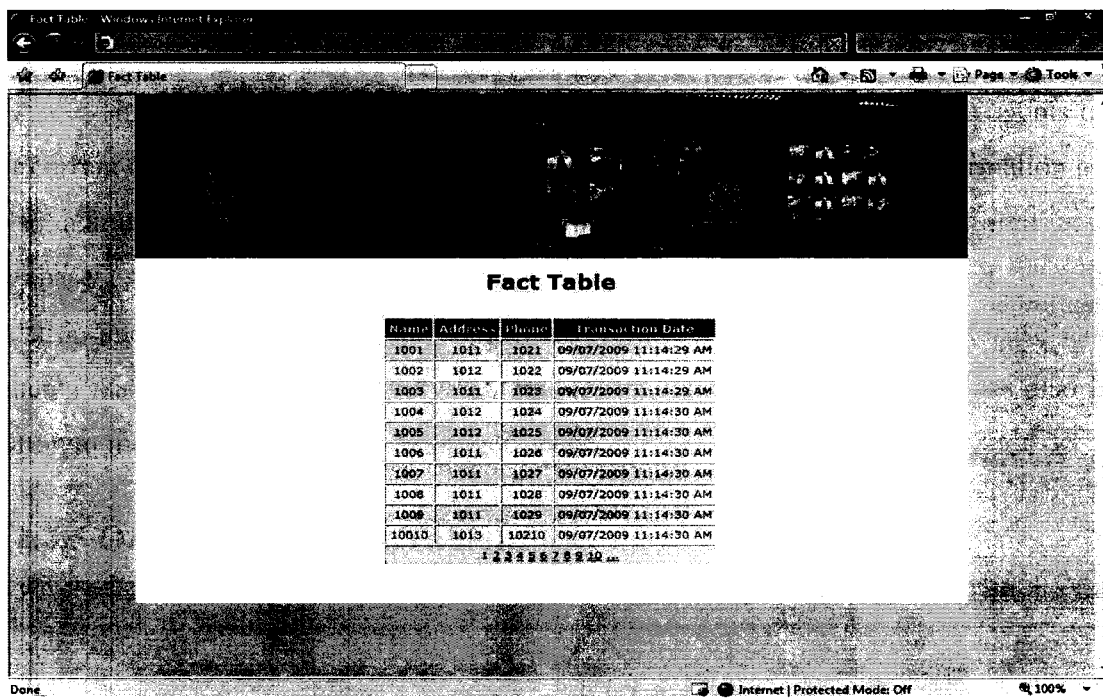


Fig 5.8: Agent Manager

This page has all the coding regarding Agents. One can view all the agents used for current database in this page. Administrator can also add new Agent information here. Also administrator can manage agents by updating their information stored in the database as well as delete an existing agent. Agent information is stored at central location and it includes agent id, unique id (prefix), source system id, database name, schema name, table name, primary key field(s) and start value. This page has also link button to move on to the "ETL Manager" page.

### 5.2.2.5 Output Page (Fact Table)



The screenshot shows a web browser window titled 'Fact Table'. The main content area displays a table with the following data:

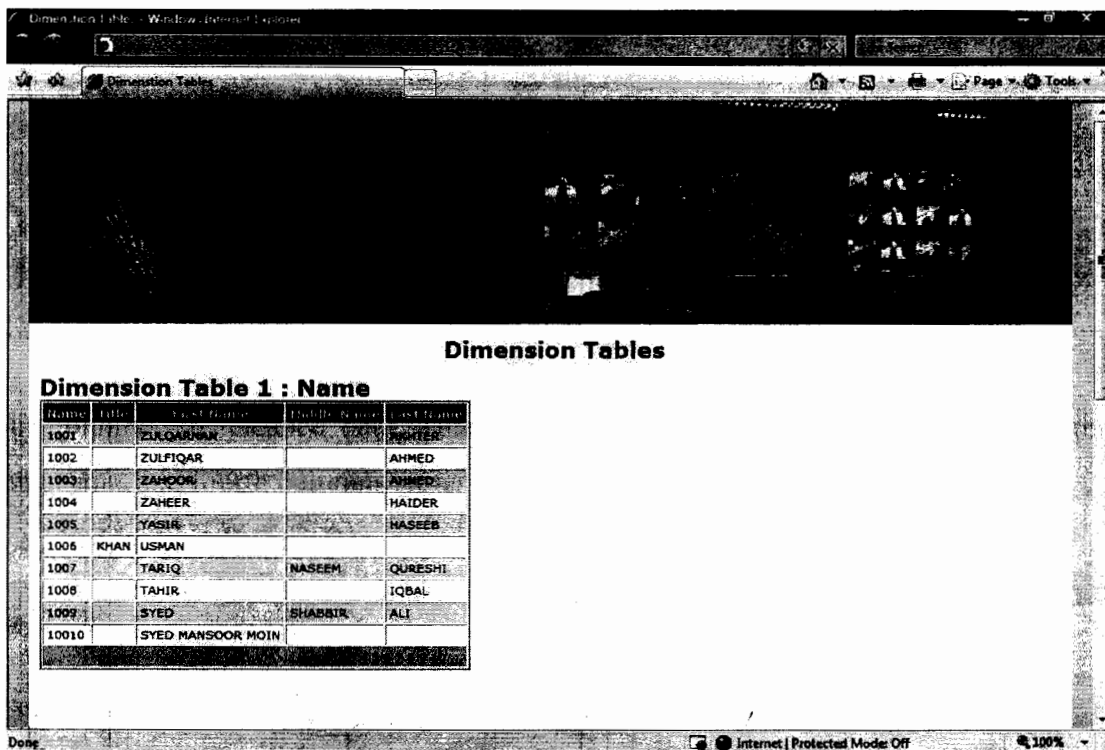
Name	Address	Phone	Transaction Date
1001	1011	1021	09/07/2009 11:14:29 AM
1002	1012	1022	09/07/2009 11:14:29 AM
1003	1011	1023	09/07/2009 11:14:29 AM
1004	1012	1024	09/07/2009 11:14:30 AM
1005	1012	1025	09/07/2009 11:14:30 AM
1006	1011	1026	09/07/2009 11:14:30 AM
1007	1011	1027	09/07/2009 11:14:30 AM
1008	1011	1028	09/07/2009 11:14:30 AM
1009	1011	1029	09/07/2009 11:14:30 AM
10010	1013	10210	09/07/2009 11:14:30 AM

Fig 5.9: Fact Table

This page is opened after the data is loaded into the data warehouse. The page contains the information of Fact Table. FT is the detail table of all the dimension tables. It has primary keys of all the dimension tables. In our example, the primary keys of Name Dimension, Address Dimension and Phone Dimension are present in it with transaction date. This page shows the success of ETL process being executed prior to it. Transactional table of an OLTP can be replaced by fact table in DW environment. In our demo application, the table contains four columns and 1251 rows selected for warehousing. After the ELT process in DWH, the fact table also contains 1251 rows with

four columns but cleaned data. Three out of four columns are referenced from three dimensions and fourth columns stores the transaction date and time.

### 5.2.2.6 Output Page (Dimension Tables)

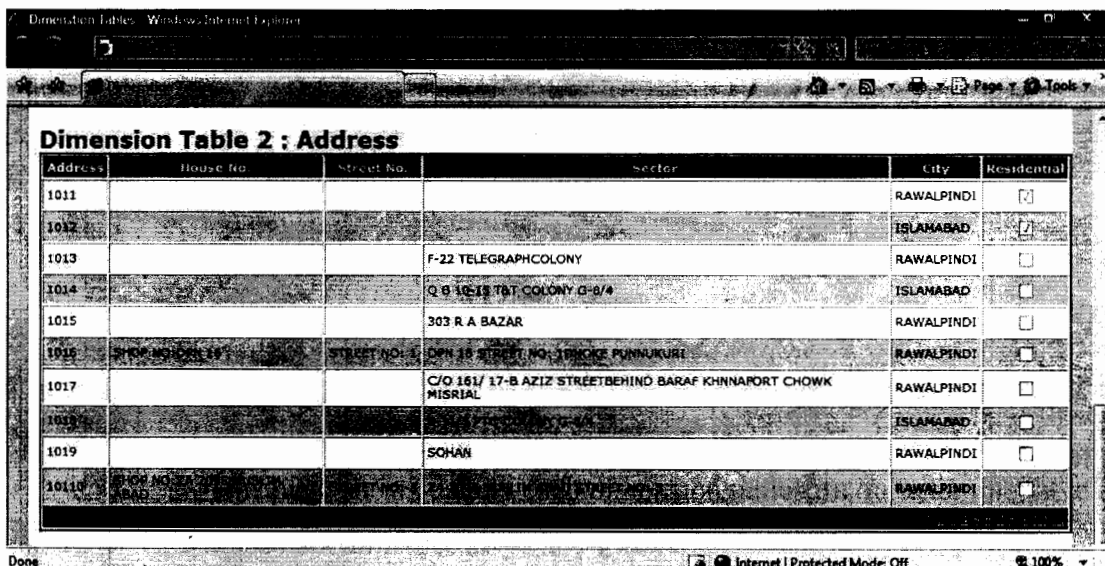


The screenshot shows a web browser window titled 'Dimension Tables'. The main content area displays a table titled 'Dimension Table 1 : Name'. The table has five columns: Name, Title, First Name, Middle Name, and Last Name. The data rows are as follows:

Name	Title	First Name	Middle Name	Last Name
1001		ZULQARNAN		HAIDER
1002		ZULFIQAR		AHMED
1003		ZANDOR		AHMED
1004		ZAHEER		HAIDER
1005		YASIR		HASEEB
1006	KHAN	USMAN		
1007		TARIQ	NASEEM	QURESHI
1008		TAHIR		IQBAL
1009		SYED	SHABBIR	ALI
10010		SYED MANSOOR MOIN		

Fig 5.10: Name Dimension Table

This page shows data in each dimension tables. In our example there are three dimension tables i.e. **Name, Address and Phone**.



The screenshot shows a web browser window titled 'Dimension Tables'. The main content area displays a table titled 'Dimension Table 2 : Address'. The table has six columns: Address, House No., Street No., Sector, City, and Residential. The data rows are as follows:

Address	House No.	Street No.	Sector	City	Residential
1011				RAWALPINDI	<input type="checkbox"/>
1012				ISLAMABAD	<input checked="" type="checkbox"/>
1013			F-22 TELEGRAPH COLONY	RAWALPINDI	<input type="checkbox"/>
1014			Q B 10-13 T&T COLONY G-8/4	ISLAMABAD	<input type="checkbox"/>
1015			303 R A BAZAR	RAWALPINDI	<input type="checkbox"/>
1016	SHOP NO. 11	STREET NO. 1	DPH 18 STREET NO. 18 CHOKI PUNNOKURI	RAWALPINDI	<input type="checkbox"/>
1017			C/O 161/ 17-B AZIZ STREET BEHIND BARAF KHINNA PORT CHOWK MISRIAL	RAWALPINDI	<input type="checkbox"/>
1018				ISLAMABAD	<input type="checkbox"/>
1019			SOHAN	RAWALPINDI	<input type="checkbox"/>
10110	TOP NO. 100			RAWALPINDI	<input type="checkbox"/>

Fig 5.11: Address Dimension Table

Phone ID	Phone No.	Description
1021	5523663	
1022	4442468	
1023	5517096	
1024	2293828	
1025	2851032	
1026	4421416	
1027	5585868	
1028	5960333	
1029	4455272	
10210	5581632	

Fig 5.12: Phone Dimension Table

### 5.2.2.7 Star Schema of Fact & Dimension(s) Tables

As there are two options for DWH modeling with star schema and snow flake schema. We have selected star schema for our demo application. It includes three dimension tables includes name, address and phone and single fact table.

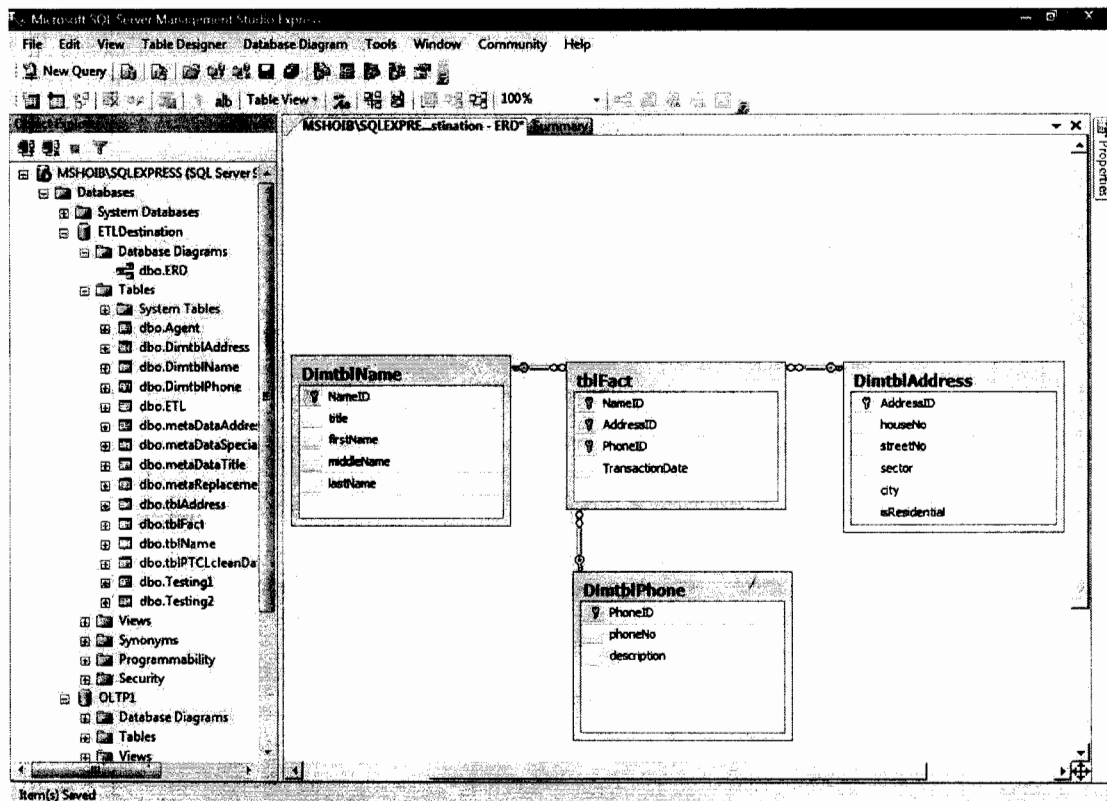


Fig 5.13: Star Schema

5.2.2.8 Source Database with Table

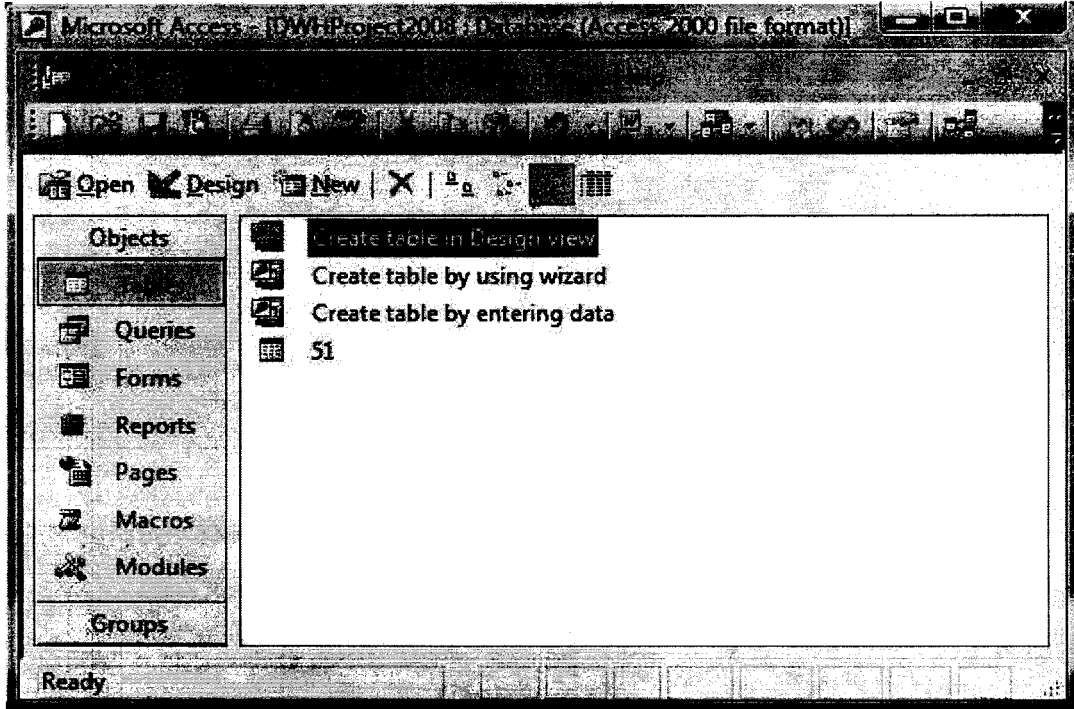


Fig 5.14: Source Database

The screenshot shows a data table in Microsoft Access. The table has four columns: 'conn\_type', 'name', 'phone', and 'address'. The data is as follows:

conn_type	name	phone	address
01	Mehnaz Hamid Mrs ADE IBA I.C. ITR PTCL P	2870000	G.M OFFICE PTCL HOUSEF-5 ISLAMABAD
01	UCH POWER LTD	2870001	OFFICE 107 1ST FLOOREVACUEE TRUST COMPLEX F-5/1ISLAMABAD
31	EMBASSY OF USA	2870002	US EMBASSY ENC RAMNA 5G-5 ISLAMABAD
31	EMBASSY OF USA	2870003	U.S EMBASSY ENCLAVE RAMNA 5G-5 ISLAMABAD
31	EMBASSY OF USA	2870004	U.S EMBASSY ENC RAMNA 5G-5 ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870005	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870006	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870007	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870008	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870009	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	NORTEL NETWORKS ASIA LTD	2870010	G/FLOOR EVACUEE TRUST BLDGF-5/1ISLAMABAD
01	MUHAMMAD AZEEM UL HAQ MINHAS	2870011	ROOM 26,27 FEDERAL LODGE-IIG-5 ISLAMABAD
01	DATA BASE SYSTEM	2870012	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOORBLK 74-W YASEEN PLAZA/B/A-F-7 IS
01	DATA BASE SYSTEM	2870013	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOORBLK 74-2 YASEEN PLAZA B/A-F-7 IS
01	DATA BASE SYSTEM	2870014	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOOR74-W YASEEN PLAZA B/A-F-7 ISLA
01	DATA BASE SYSTEM	2870015	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOORBLK 74-W YASEEN PLAZA/B/AREA I
01	DATA BASE SYSTEM	2870016	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOORBLK 74-W YASIN PLAZA B/AREA-F-7
01	DATA BASE SYSTEM	2870017	DIR AZHAR AMIN CH.HALF PORTION 1ST FLOORBLK 74-W YASEEN PLAZA/B/A-F-7 IS
01	MANSOOR HASSAN KHAN	2870018	H/NO 4 ST 50 F-7/4ISLAMABADISLAMABAD
01	MANSOOR HASSAN KHAN	2870019	H/NO 4 ST 50 F-7/4ISLAMABADISLAMABAD
01	MANSOOR HASSAN KHAN	2870020	H/NO 4 ST 50 F-7/4ISLAMABADISLAMABAD
01	HAMIDA NIZAM UD DIN	2870021	H/NO 10 ST 61 F-6/3ISLAMABAD
01	CHAUDHARY MUHAMMAD SHAFI	2870022	HOUSE NO 330 BG 6/2ISLAMABAD
01	HAFIZ NAZIR AHMED MIRZA	2870023	H/NO 2/4-B ST 18 G-7/2ISLAMABAD
01	SAKEENA BIBI W/O YAQOOB MASHI	2870025	H/NO 812-A ST 167 G-7/3-1ISLAMABAD
01	MAJ NADEEM SHAFIQ	2870026	H/NO12 ST 19 F-6/2ISLAMABAD
01	SEEMA BIBI	2870027	H/NO 195-A ST 40 G-6/1-3ISLAMABAD
01	PAKISTAN TOBACCO CO LTD	2870028	H/NO 210 ST 12 E-7ISLAMABAD
31	AMERICAN EMBASSY	2870029	CHANCERY BUILDINGDIPLOMATIC ENCLAVEISLAMABAD
01	MUHAMMAD BASHIR KHETRAN	2870030	FLAT 24-C BLK 72 F/SUITF-5/1 ISLAMABAD
31	EMBASSY OF MARACCO	2870031	H/NO 6 GOMAL RD E-7ISLAMABAD
06	FATEH UL AZAM	2870032	H/NO 26 ST 19 F-6/2ISLAMABAD
51	DIR TL-1	2870033	ROOM 108 PTCL HOUSEISLAMABAD
51	SYSTEM ANALYST	2870034	ROOM 313 REGIONAL OFFICE/ISI AMARAN

The status bar at the bottom shows 'Records: 14 of 51' and 'Datashheet View'.

Fig 5.15: Source Table Data View

Figure 5.15 shows the data view of source table. The data in the table is not clean and is inconsistent. There are spelling mistakes, missing words or fields. For example Islamabad is entered differently in different as ISB, I.S.B, I.S and Islamabad. Similarly for house no, different styles are present including HNO, H.No, H/No, House #, H #, H-#. Same is the case with name field. The main purpose of transformation module in ETL process is to remove such kind of inconsistencies so that same record is not saved again and again, as in DWH historical data has to be save ranging upto Tera Bytes (TB) of data.

### 5.2.2.9 Agent Manager Window in SAGE

Multi agent systems provide execution environment for intelligent software (agents). There are different standard governing bodies. We especially focus on Foundation for Intelligent Physical Agent (FIPA) standard which we follow throughout our thesis. In multi agents system single task is divided among multiple agents on the same or different platforms. The proposed architecture is based on the existing FIPA compliant multi agent system framework, namely SAGE-Lite [68].

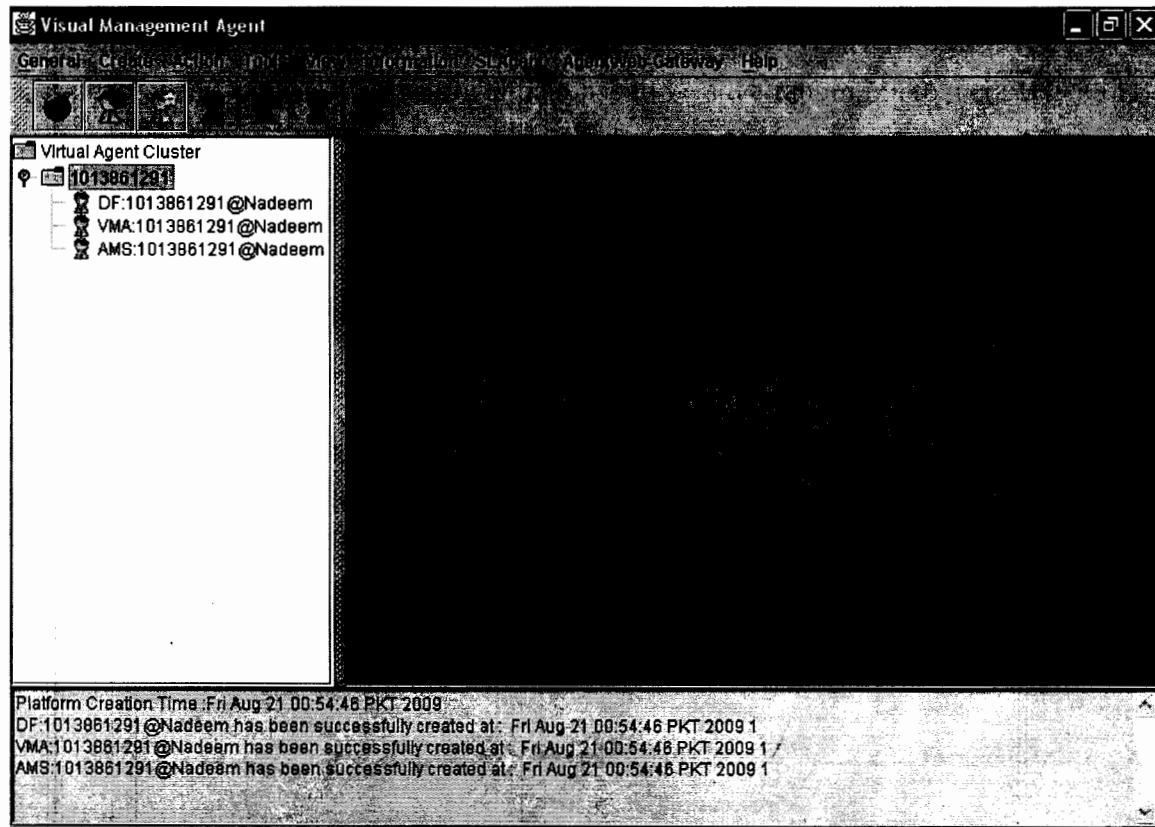


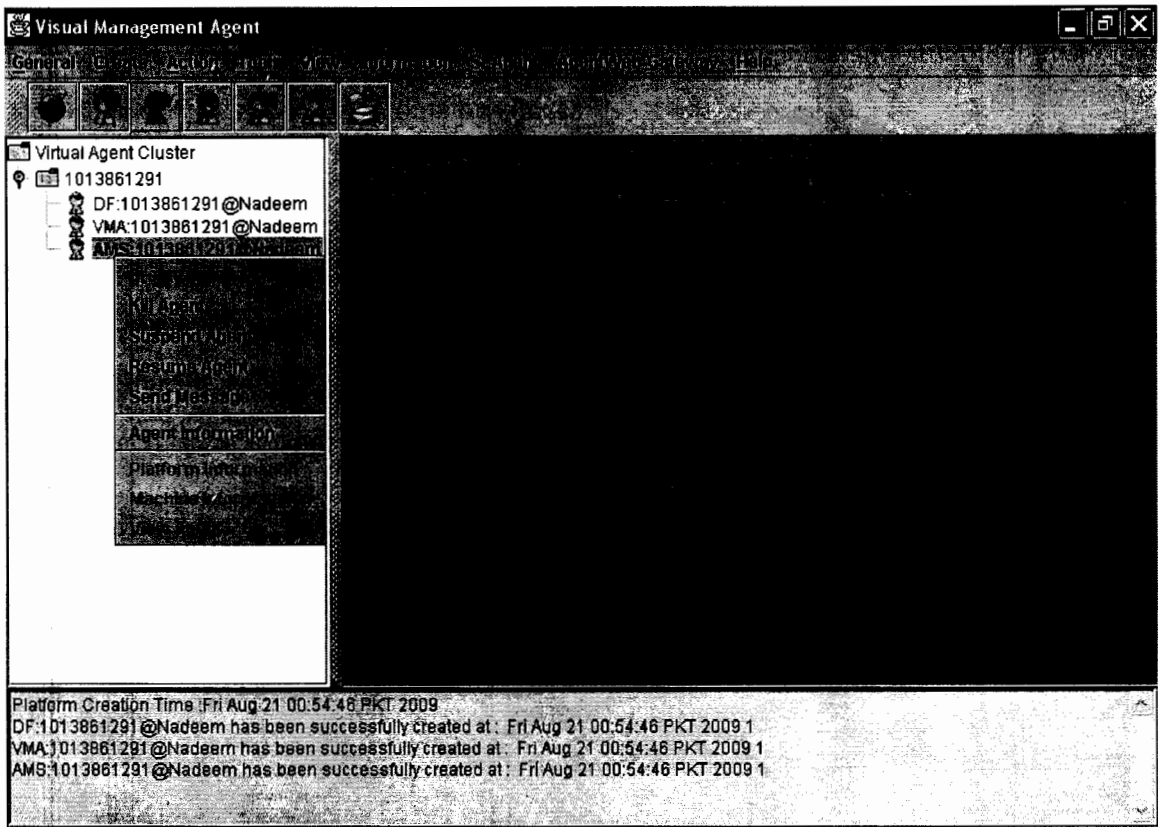
Fig 5.16: Agent Management in SAGE



SAGE-Lite is a lightweight, open source context aware multi agents system, which senses the capabilities of any system and reacts accordingly. In the proposed technique generating keys are published as services of the agents and any other agent can access the services according to its privileges.

As we are getting the task of generating the SKs from multi agents, so we have inherited the working of SAGE framework with modification in our demo application. The above snapshot is taken from the SAGE framework and shows the Agent Manager Window. This window enlists all the agents created so far and also the date and time of creation of each agent. The window also contains a toolbar having different commands to control each agent loaded at time including **Pause**, **Resume**, **Stop** and **Create New**.

### 5.2.2.10 Agent Actions in SAGE



**Fig 5.17: Agent Actions in SAGE**

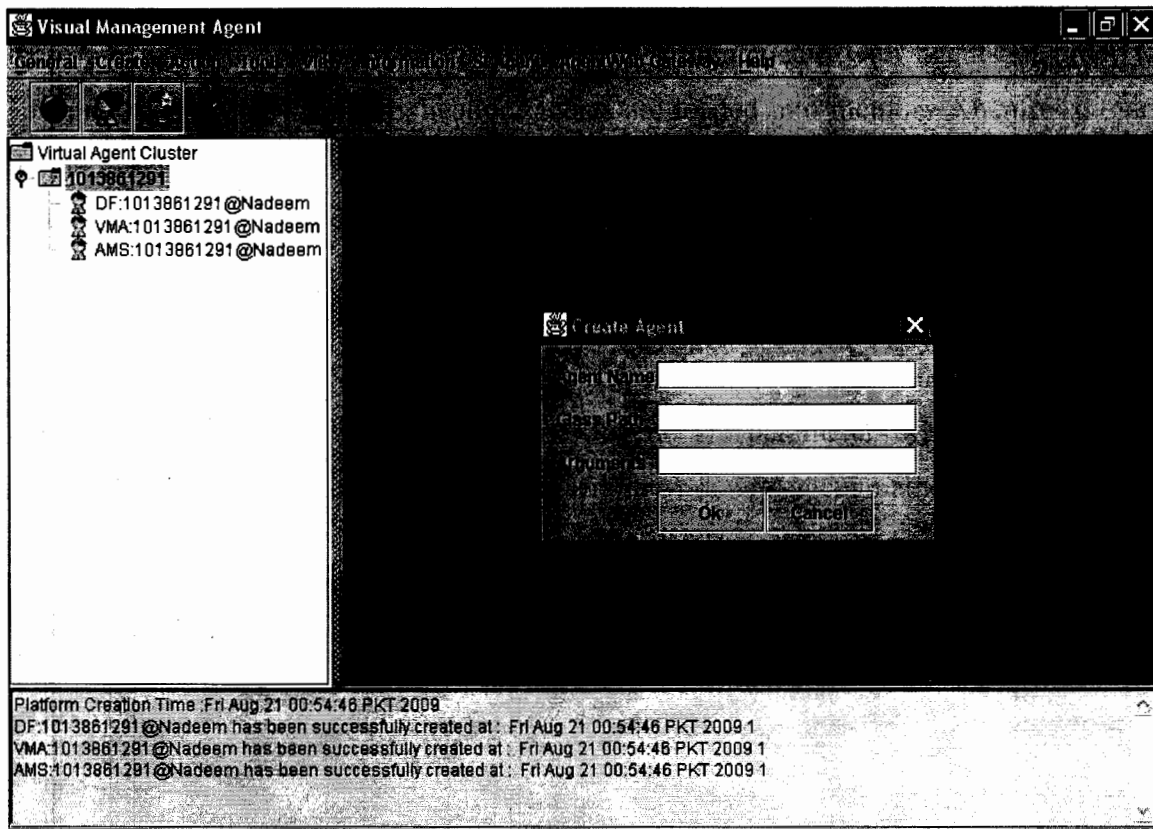
The actions which can be performed on an agent in SAGE frameworks are creating new agent, killing created agent, suspending an agent, resuming suspended agent, sending messages between agents, viewing information about agent, platform and

machine. All these actions are grouped in a popup menu and launched by right clicking on any node in agent tree.

We have included only creating and killing operations in our framework and these are invoked by the application needed to communicate with agents in order to generate keys. In our demo application, agents are loaded into memory when press start button in ETL Manager Window and they are terminated on completion of ETL process and when results are displayed on new window.

### 5.2.2.11 Agent Creation in SAGE

Figure 5.18 shows the agent creation windows. The main window opens a new dialog within previous window to create new agent. The parameters which are needed to supply in order to create agent in SAGE framework are agent name, class path and arguments.



**Fig 5.18: Agent Creation in SAGE**

We have inherited the same idea in our framework but getting information about agent as well as data on which it would operate. This data includes agent id, unique id,

system name, database name, schema name, table name, primary key field(s) and starting value. After creation of agent it is added into the agent tree and its information about agent name, date and time of creation is displayed on the status window at the bottom.

### **5.2.3 Proposed Framework Work Flow**

In figure 5.19 we have presented the visualization work flow of our demo application based on our proposed framework. In this representation data flow from process to process is replaced with actual table field(s) transformed and processed in each module. Data after transformation is mature enough to load into DWH, so we divide it into logically chunks according to the subject or dimension. Then each chunk is assigned to an agent which is responsible to generate key against each distinct record of that chunk. Then after getting the keys, data is loaded into dimension tables and at last all the data is present in temporary table is load into fact table by removing attributes with dimension keys.

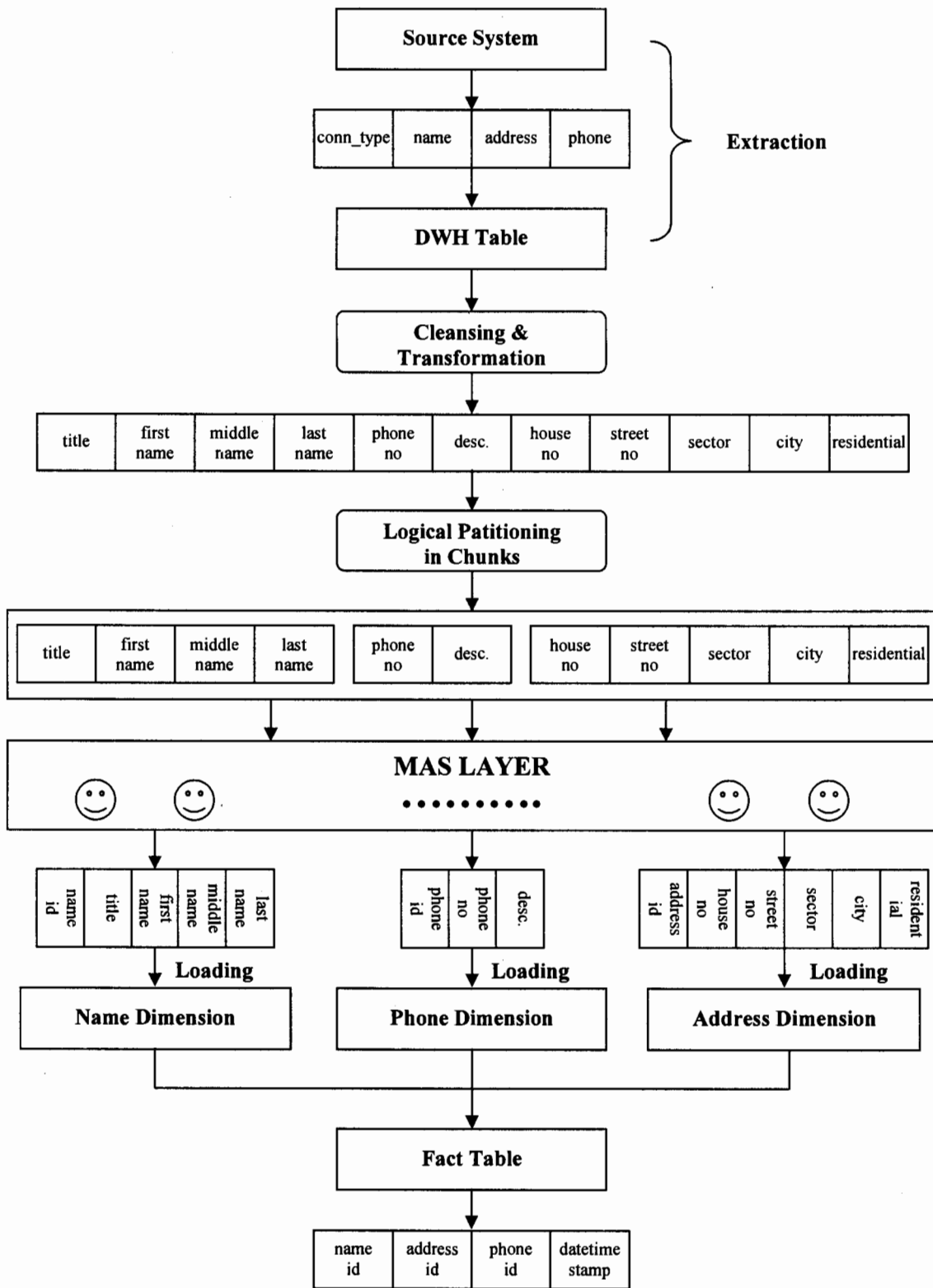


Fig 5.19: Proposed Framework Flow Visualization

### 5.3 Summary

In this chapter we have described in detail the implementation of our proposed framework in our demo application. We have taken a database as an example and perform the ETL process with key generation through agents and the loading of data into warehouse. We have also presented the visualization flow of our framework through which it is easy to understand.

## **CHAPTER 6**

# **TESTING AND PERFORMANCE**

In this section we have described the criteria through which we have intended to judge the success and efficiency of our proposed framework. As discussed in the chapter 2, where we surveyed and analyzed different approaches to generate surrogate keys for the dimensions of data warehouse system and also presented the advantages and limitations of each approach. In continuation to the previous analysis of different parameters, we are now presenting our proposed solution for the aforementioned problems, by doing a comparison with the same parameters.

## 6.1 Test Scenarios

As we have to compare the performance of our framework with pre existing methods and we use multi agents system approach in which a task is divided and distributed between different agents to work individually and parallel, so we have tested our results by using three test scenarios.

- i. Key Generation by ordinary methods.
- ii. Key Generation by ordinary methods with parallel processes.
- iii. Key Generation by multi agents.

## 6.2 Testing and Results

We have created a demo application in which we have used our proposed framework, discussed in chapter 5. As complete ETL module is included in this demo application, so we have used same application to test our three test scenarios for different test cases and parameters.

## 6.3 Comparison of Key Generation by Agents and Ordinary Methods

We have presented the side by side comparison of the proposed solution with other existing technologies using following parameters and test cases.

- Source System Tracing.
- Agent based Framework Performance.
- Database Volume vs. Response Time.
- Transactions Volume vs. Response Time.
- Concurrency Optimization.
- Reduced I/O Operation.

- Gaps between sequentially generated numbers.
- Replication.

### 6.3.1 Source System Tracing

As data come from different sources held either homogeneous or heterogeneous in data warehouse and after ETL process where cleansing and transformation are performed to make data consistent, so there is no method, to the best of our knowledge, to trace out the source of data from which it has come into the DWH. In our proposed method, as agents generate keys for the DWH record and its three digit id is used as prefix of the generated key, so we can trace the source of the record in DWH from the key because information about agent id, source system name, database name, schema name and table name are stored at central location against each agent.

#### 6.3.1.1 Example Source Tracing: Ordinary vs. Agents

Name Id	Title	First Name	Middle Name	Last Name
101	Syed	Mansoor		Khan
102	Syed	Shabir		Ali
103		Tahir		Iqbal
104		Tariq	Naseem	Qureshi
105	Khan	Usman		

**Table 6.1a: Keys generated by Ordinary Methods**

Name Id	Title	First Name	Middle Name	Last Name
1001	Syed	Mansoor		Khan
1002	Syed	Shabir		Ali
1011		Tahir		Iqbal
1021		Tariq	Naseem	Qureshi
1012	Khan	Usman		

**Table 6.1b: Keys generated by Multi Agents**

In table 6.1a keys are numeric and sequential but seem to belong to a single source. There is no distinction that record is either come from one source or many sources. In table 6.1b keys are also numeric and sequential to some extent but there exists a distinction of record that it comes from which source. First three digits are the prefix



added to the key and they show the agent id. In above table key “1001” means that it is generated by an agent with id “100” with sequence number “1” and same is the case with keys “1011” and “1021”. Hence records with keys “1001” and “1002” belong to one source, keys “1011”, “1012” belong to second and “1021” belong to third source. it is not compulsory that keys come from different sources, they can be from different databases on single source, or different schemas in single database, or different tables in single schema.

### 6.3.2 Agent based Framework Performance

To test our technique we have run our proposed framework based demo application with the sample data. As we have used three test scenarios so in order to check the performance we have taken result for same sample data using ordinary key generation methods and key generation through parallel process.

Sample No. of Rows	Ordinary Methods (Min)	Parallel Process (Min)	Agents (Min)
1000	20	16	11
10,000	35	25	19
100,000	60	40	32
200,000	100	55	46
250,000	115	60	50

**Table 6.2: Time taken by each Method for ETL**

Table 6.2 shows the total time required by ETL process to extract data from source system to load data into DWH. Here we have taken different samples of data from source system to check against three scenarios. As clear from table that ordinary methods take too time by comparing others. So it can be improved by dividing the tasks and transactions into parallel processes. But using our agent based approach there is a significance performance difference against other methods. The main reason is that it is also using parallel processes but the task of generating keys is now performed by agents not by DBMS itself. During key generation time DBMS can do other tasks. So we can enhance the efficiency by reducing the burden from DBMS.

Other important observation from table 6.2 is that as the volume of sample data is increasing, the time for ETL is not increasing with same factor. Hence it can be concluded that as volume of data increase in ETL, their time for processing reduce due to caching.

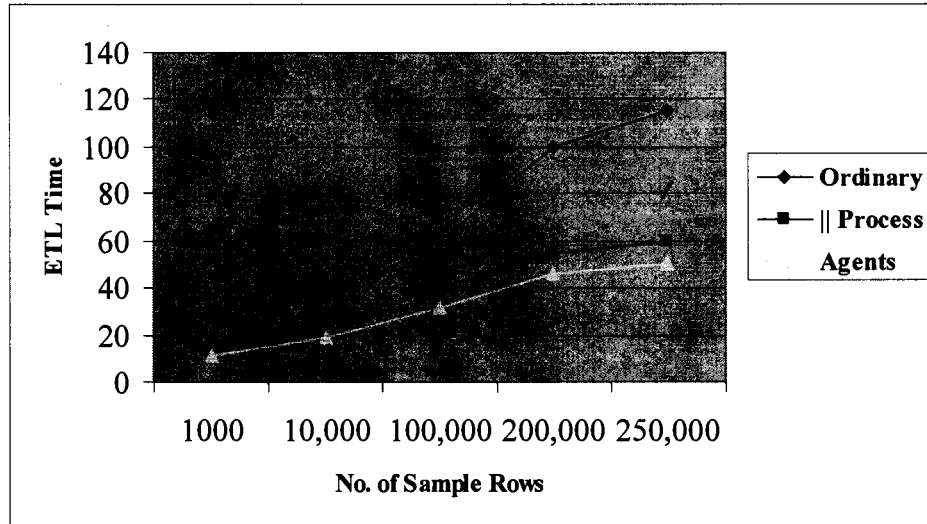


Fig 6.1: Time taken by Each Method for ETL

Figure 6.1 shows the performance of ETL process against each method graphically. It is clear that performance can be improved by using multi agent in ETL process.

### 6.3.3 Database Volume vs. Response Time

With the linearly growth of the size of the database, response times grow logarithmically.

Volume of Database (MB)	Ordinary Methods (Sec)	Parallel Process (Sec)	Agents (Sec)
1	30	16	15
100	55	25	22
200	80	40	34
300	105	55	46
400	130	60	50

Table 6.3: Data Volume vs. Response Time

Table 6.3 shows the three scenarios implemented in our demo application with respect to increasing size of database and its response time. As clear from the table agents based methods give improve results.

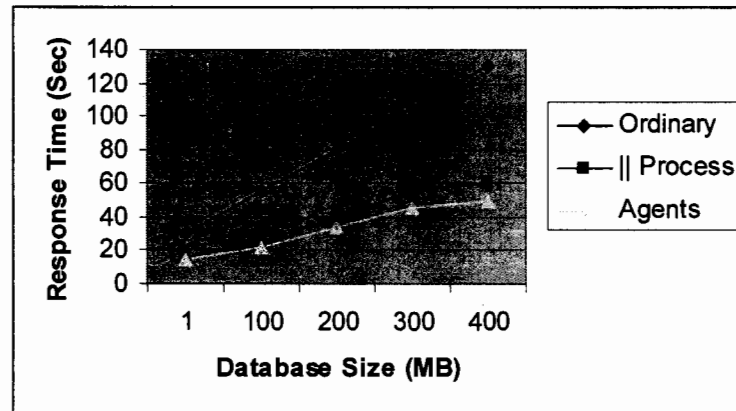


Fig 6.2: Database Volume vs. Response Time

Figure 6.2 shows the response time against increase in volume of database in each scenario. Response time can be improved by using parallel process in DWH but using agents give more satisfactory results. It is clear that in start the response time of parallel process and agents based methods are nearly equal when volume of database is small. As the volume increases, the efficiency of agents based methods also increase more rapidly than parallel process method.

### 6.3.4 Transactions Volume vs. Response Time

With the linearly growth of the transactions like size of database, response times grow logarithmically.

No. of Transactions	Ordinary Methods (Sec)	Parallel Process (Sec)	Agents (Sec)
1000	15	8	7.5
2000	27	12	11
3000	39	20	17.3
4000	50	26	23
5000	59	30	25

Table 6.4: Transaction Volume vs. Response Time

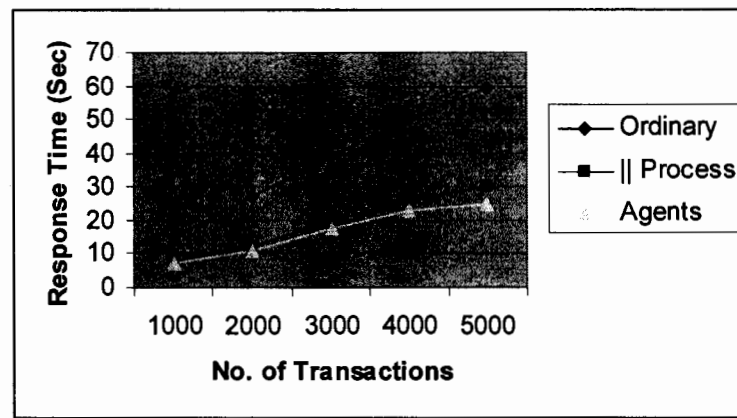


Fig 6.3: Transaction Growth vs. Response Time

Table 6.4 shows the three scenarios implemented in our demo application with respect to increasing number of transactions and its response time. As clear from the table agents based methods give improve results.

Figure 6.3 shows the response time against increase in number of transactions in each scenario. Response time can be improved by using parallel process in DWH but using agents give more satisfactory results. It is clear that in start the response time of parallel process and agents based methods are nearly equal when number of transactions is less. As the transactions increase, the efficiency of agents based methods also increase more rapidly than parallel process method.

### 6.3.5 Concurrency Optimization

Indexes play a major role in the key generation in a concurrent environment.

There are four scenarios for indexes include:

- No indexing table data.
- Clustered indexing on table key.
- Non Clustered indexing on table key.
- Non Clustered indexing on table key plus Clustered indexing on another key.

As there are different methods to generate surrogate keys, so these methods provide different results in different environment. Some are efficient in insert intensive environment and some are efficient in low activity environment. Now we consider tow performance parameters to compare different methods.

- i. User's request per second for fetching rows within certain time against user load.
- ii. Response time for fetching rows within certain time against user load.

### Case 1: Request/Sec vs. User Load

User Load	Ordinary Methods (fetched rows)	Parallel Process (fetched rows)	Agents (fetched rows)
20	50	52	53
40	75	78	80
60	85	87	95
80	90	90	100
100	92	91	110

Table 6.5a: Request per Sec vs. User Load

Table 6.5a shows the rows fetched by each method upon increasing number of concurrent process. Agents give us high values on increasing number of concurrent processes.

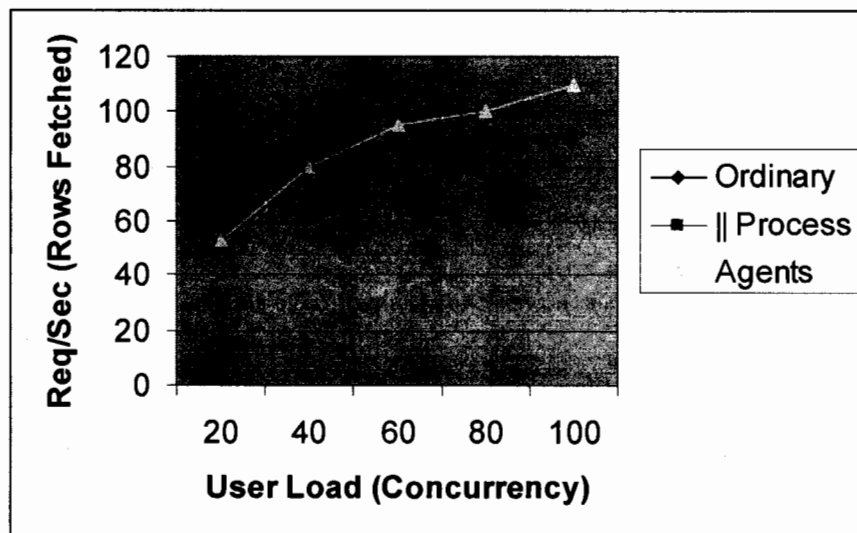


Fig 6.4a: Request per Sec vs. User Load

Figure 6.4a shows the rows fetched by each method in a concurrent environment. The efficiency of agent method is very clear from the figure. It can be

notices that row fetched by each method is increasing rapidly but with an increase in user load it is not increasing with the same ratio. The reason is the hot spot creation when multiple processes are trying to access same memory area. Hence dead lock occurs.

### Case 2: Response Time vs. User Load

User Load	Ordinary Methods (Response Time Sec)	Parallel Process (Response Time Sec)	Agents (Response Time Sec)
20	10	9	7
40	19	18	15
60	31	27	22
80	45	38	32
100	57	47	41

Table 6.5b: Response Time vs. User Load

Table 6.5b shows the response time by each method upon increasing number of concurrent process. Agents give us high values on increasing number of concurrent processes.

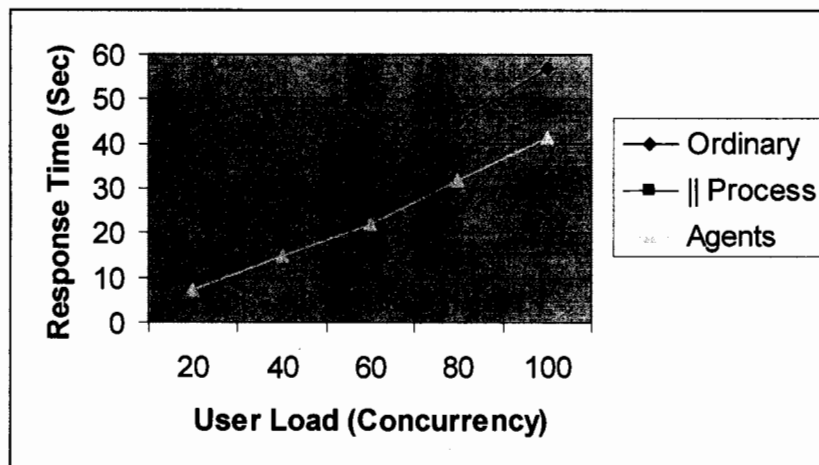


Fig 6.4b: Response Time vs. User Load

Figure 6.4b shows the response time by each method in a concurrent environment. The efficiency of agent method is very clear from the figure. The efficiency of agent method is clear from the figure.

### 6.3.6 Reduced I/O Operations

I/O operations play an important role in performance of any query in DWH environment. DBMS uses data pages for I/O operations which consist of indexed rows. I/O operations include disc access, memory read/write, fetching rows and inserting rows. If a data page having required row to get the results so it is read into the memory. If we use large keys so size of fact table will be increased and less rows will be stored on any data page. So with the increase in data pages more I/O operations are required in order to satisfy the query.

We now consider the I/O operations regarding DBMS. In our proposed approach, all the task of key generation is done by agents not DBMS, whereas in ordinary methods it is done by DBMS itself. As I/O operations are one of the main cause of bottleneck in performance, so we can reduce so I/O operations done by DBMS by shifting them to agents.

No. of Rows	Ordinary Methods (Sec)			Parallel Process (Sec)			Agents (Sec)		
	KG	I/O	Total	KG	I/O	Total	KG	I/O	Total
1000	3	3	6	2	2	4	0	2	2
10,000	5	6	11	3.5	5	8.5	0	5	5
100,000	20	25	45	13	16	28	0	16	16
200,000	35	40	75	25	28	53	0	28	28
250,000	42	47	89	33	35	68	0	35	35

Table 6.6: DBMS I/O Operations

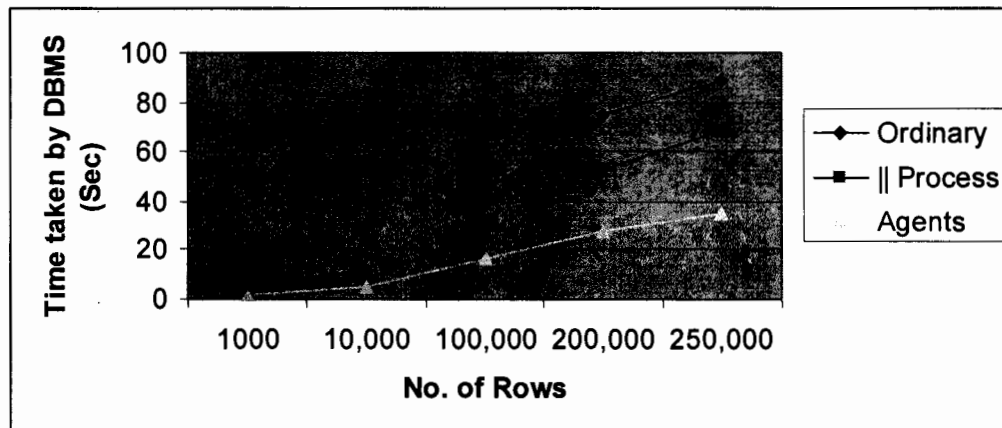


Fig 6.5: DBMS I/O Operations

Table 6.6 shows the time taken by key generation, other I/O operations and total time consumed for processing the sample number of rows for each key generation methods. In agent based method, the task of key generation is the responsibility of agents so time taken in key generation for DBMS is zero. It makes all the difference. Another observation is that time taken for I/O operations for parallel process and agent based method are same.

The efficiency of key generation by agents can be observed clearly from the figure 6.5. We can make the efficiency double by using agent based approach rather than using ordinary approaches.

### 6.3.7 Gaps between sequentially generated numbers

The auto-increments key like Identity method is considered to be most efficient methods for key generation for DWH. But the main disadvantage with this method is gaps created between sequentially numbers. As the volume of DWH data is in TB (Tera Bytes), so there can be huge number of missing sequences wastage. There are four major caused of gaps generation within numbers.

- i. Abortion of DML Statements (Automatically Role Back).
- ii. Deletion of Rows.
- iii. Resetting Seed Value.
- iv. Rolling Back Transactions Manually.

In our proposed method the gaps between sequentially number are handled by agents itself. So it can be controlled easily.

#### 6.3.7.1 Example: Gaps Generation

Name Id	Title	First Name	Middle Name	Last Name
101		Yasir		Habib
102		Zaheer		Haider
110		Zahoor		Ahmed
111		Zulfiqar		Ahmed
125		Zulqurnain		Akhter

**Table 6.7a: Keys generated by Ordinary Methods**



Name Id	Title	First Name	Middle Name	Last Name
1001		Yasir		Habib
1002		Zaheer		Haider
1003		Zahoor		Ahmed
1011		Zulfiqar		Ahmed
1012		Zulqurnain		Akhter

**Table 6.7b: Keys generated by Multi Agents**

Table 6.7a shows the key generation by ordinary methods. By going through the “Name Id” field, the gaps between numbers are very usual. This drawback is overcome by using agent based method and it can be seen from table 6.7b. There are two sequences in our sample data generated by two agents but without gaps.

### 6.3.8 Replication

Replication is very important with administration point of view. As DWH holds historical data, so its back is very necessary for accidental situations. Replication can be used for backup purposes. In replicating the data table, not whole image is copied but insert one by one each row of the table. So tables having columns with identity property to generate numbers are generated again. There can be a situation where the keys in original table are not generated in the same order in replicated place as they were in source system. We can explain it with example using tables as:

#### 6.3.8.1 Example: Replication

Name Id	Title	First Name	Middle Name	Last Name
101		Yasir		Habib
102		Zaheer		Haider
103		Zahoor		Ahmed
104		Zulfiqar		Ahmed
105		Zulqurnain		Akhter

**Table 6.8a: Keys generated by Ordinary Methods in Source**

Table 6.8a shows the sample data of a table having “Name Id” as Identity field and it is the five records in the source system. In replicating this data there can be

possibility that Identity field generates value with other seed value rather than as in source system.

Name Id	Title	First Name	Middle Name	Last Name
110		Yasir		Habib
111		Zaheer		Haider
112		Zahoor		Ahmed
113		Zulfiqar		Ahmed
114		Zulqurnain		Akhter

**Table 6.8b: Keys generated by Ordinary Methods in Replication**

Table 6.8b shows the key sequence of sample table after replication. By comparing both tables, it is clear that in replication DBMS has different seed value for that table. So there is obvious difference between keys of both systems. As surrogate keys are meaningless keys and they are invisible to users but still they are meaningful by DBMS to manage integrity between dimensions and fact tables as well as look ups. By using our agent based approach this problem can be overcome because agents are customized so we can reset its starting value easily when deploying replicated table. Using agents for key generation both tables at source and replication point would be same with respect to keys order.

#### 6.4 Other Benefits

Other benefits of key generation using multi agents rather than using ordinary key generation methods are:

- i. Portability.
- ii. Scalability.
- iii. Simplified Joins.
- iv. High Performance Joins.

#### 6.5 Summary

As there are different methods to generate key for the DWH, and each has its own pros and cons with respect to implemented environment. In this chapter we have compare our proposed approach with other approaches against different scenarios and performance parameters. It can be concluded that the efficiency can be achieved in DWH

environment by using parallel processing instead of standalone ordinary methods. Our proposed method has same mechanism by using multi agents in key generation. It can be concluded from the facts and figures described in this chapter that by using multi agents a very huge performance benefits can be achieved.

# **CHAPTER 7**

## **CONCLUSION AND OUTLOOK**

## 7.1 Conclusion

Agents are used for DWH for out put i.e. reporting and multi level analysis purposes which includes sorting, filtering, alerting, threat identification in business and monitoring related tasks. We have used and implement agents for surrogate key generation and we have taken extra ordinary performance enhancing benefits from it by comparing with other methods. So we conclude that using agents during DWH design and implementation is a good practice for developers.

## 7.2 Contribution

We have used "SAGE" agent framework for the implementation of proposed approach. SAGE is open source created in JAVA. SAGE has its own GUI for creating and managing agents. Instead of running separate application for SAGE, we have inherited the Agent creation and managing which includes killing, suspending, resuming and sending messages process in our application and made our own module i.e. it will execute the "exe" files of each agent. Agents are called with proper parameter to generate the keys and they return the key for that particular row.

## 7.3 Out Look

The performance of agents can be improved by adding more intelligence to them. So in future we will automate the process of agent creation rather than creating manually as they are customizable. Further more it can be improved for concurrent transactions as well as its new number generation algorithm.

# **APPENDIX A**

## **RESEARCH PAPER**

# Surrogate Key Generation: A Generic Approach

Muhammad Nadeem Yasin, Muhammad Shoaib, Ainan Sadiq, Muhammad Imran Saeed  
International Islamic University, Islamabad

Emails [mnadeem.yasin@hotmail.com](mailto:mnadeem.yasin@hotmail.com), [shoaibshaaqiiu@yahoo.com](mailto:shoaibshaaqiiu@yahoo.com), [ainansadiq@gmail.com](mailto:ainansadiq@gmail.com),  
[imran@iiu.edu.pk](mailto:imran@iiu.edu.pk)

**Abstract**— A central repository to store huge amount of data from multiple information based sources is known as data warehouse. The data in warehouse is transformed and formalized into single format for the analysis process. The primary reason for using data warehousing is to provide analytics result to businesses from data mining, OLAP, score carding and reporting. Sorting of the data warehouse records facilitate the searching and mining algorithms. The sorting can be performed either by using dimensional approach or through normalized approach. Since primary keys can not be applied on the data of heterogeneous databases so to uniquely identify a record from billions of heterogeneous records, data warehouse introduces the concept of surrogate keys. Researchers have proposed many algorithms to generate the surrogate keys but all of them have some lacking with respect to different parameters. Few techniques creates gaps among the records, few results in deadlock and hot spots, some of the techniques assign the same key to multiple records, some increases the searching and mining delay and few techniques generates the bigger keys hence increases the storage and processing load. We in this paper introduced a new technique to generate the surrogate keys through software agents. The proposed technique will give better results in all the dimensions discussed above. It will avoid the deadlock, reduce the delay and size, eliminate the chances of duplication, and will support the searching and mining techniques. The most important advantage of the proposed approach is to trace the source system of the particular record as records come from heterogeneous environment.

**Index Terms**— ETL, Surrogate Keys, Data Warehousing, Multi Agents, Dimensions.

## I. INTRODUCTION

There are many dimensional modeling concepts and techniques that are critical to implement dimensional models. The key concepts are: surrogate keys and slowly changing dimensions.

Surrogate keys are keys that are created and maintained within the data warehouse instead of the taking natural keys of source systems. SK contains not any information and independent of data itself. These are used in place of heavy meaningful composite keys of the source systems due to performance reasons [9]. **Performance** and **Semantic Homogeneity** are the basic reasons for this replacement [16]. Surrogate keys are named by many other aliases, such as non-natural keys, system generated keys, dummy keys,

meaningless keys, database sequence numbers, arbitrary unique identifier, artificial keys, non-intelligent keys, entity identifiers, synthetic keys, technical keys and integer keys. Dimension tables should always be built with a surrogate key assigned by the ETL process. It is recommended that SK are created and used as the PK of all the dimension tables [8]. The surrogate keys joins the dimension tables to the fact table [19]. Surrogate keys serve as an important means of identifying each entity or instance inside a dimension table [11]. SK are not defined in Logical Model [25]. SK provides better performance with respect to joins due to their numeric nature [13].

Surrogate Keys and Primary Keys can be differentiate on the basis of the type of the database whether **Temporal** or **Current** database [6]. SK can be used as primary key in current database because it stores the data that is currently valid and the relationship between PK and SK is one-to-one. However the SK can not be used as primary key in temporal database because the correspondence between PK and SK is many-to-one. Hence there is a need of an attribute other than SK to uniquely identify any object in temporal database.

Surrogate key generation is the one major step in getting the data from source system and delivering it to the target user and it takes a lot of time i.e 35% [27] of overall data warehousing process as depicted in table 1.

Table 1: Data Warehousing Steps

STEP #	DESCRIPTION	TYPICAL SHARE OF TIME
1	Request a data package.	0%
2	Extract data from the source.	10% of the ETL process
3	Transformation rules/transfer to the PSA.	15% of the ETL process
4	Update rules.	40% of the ETL process
5	Surrogate key generation and database updates.	35% of the ETL process
6	Subsequent steps.	(varies)

The most critical cost of using surrogate keys is that it places the burden on the ETL process [22]. Data normalization, query optimization, data disassociation, business process modeling [6] are also the cost factors. Other

disadvantages of Surrogate keys include extra space due to addition of a new column, hence extra index has to be built on SK column, difficulty in changing SK, data verification is not possible, as cluster index can give fast searching mechanism but cluster index on SK is useless and getting next value [9][21].

Surrogate Keys are generated in the data-staging. Also the maps to the operational systems, current loads of operational data, and any atomic data not currently used in the data marts is stored in the Data Staging. According to Vincent McBurney, there are two way to generate Surrogate Keys in the data load of ETL process [13]. First, Surrogate Keys can be generated in the ETL job. Second, Surrogate Keys can be generated in the target database. As surrogate keys are generated for the dimension tables during the load process of ETL, a master cross-reference table for each dimension must also be maintained. This table keeps the track that a particular SK is assigned against which business key along with time information.

## II. RELATED WORK

### A. Surrogate Keys Generation Methods and Limitations

Joe Celko and Markus Zywitzka have described the basic method to generate SK by using a **counter** [9]. This counter is maintained by DB engine and usually starts with one. A function is used to increment the value of counter locally by one [5]. In Sybase SQL the function NUMBER(\*) is the example of this method.

The basic problem with this method is that it can not be useful in multi client environment and also for clusters or an environment where data comes from heterogeneous sources. The second problem with this method is the overflow of number range. This method is also not useful under the circumstances where data from different databases of organization have to merge. There can be clash of counter value that is an object represented in one database can be different in other database. So in order to gain consistency a lot of work has to be done.

Another method to generate SK is to use the Oracle "**Sequences**" [2][4]. It automatically generates a unique number which can be used as key of any kind of table. This is customized in such a way that user can set the incremented value between two numbers and also the limit of numbers. The main limitation of this technique is that it is only supported in Oracle.

An automatically generated value with "**IDENTITY**" keyword in SQL Server, Sybase can also be used for SK generation [10], as described by Gary Meyer. It is also customized as user can set the initial value and increment value. The value of identity column is sequentially unique numbers and generated on saving of record. A table can have one identity column [11]. The main advantages of using this method includes the keys are small, high performance on joining and indexing, debugging is easy and key updating is not allowed.

There are also some drawbacks of using this technique. The main drawback is that it is problematic in replication. Second,

generated values can only be accessed after insertion. Other includes the gaps between generated numbers, uniqueness in a single table, not portable, creates insertion hot spot in concurrent environment, must declare as primary key to avoid duplication [18].

According to Scott W. Ambler the simplest solution to surrogate primary key generation is the "**Max Plus One**" [7]. In this strategy a column with numeric data and starting from one is used. On insertion of any new record, the maximum value on this column by using Max function is calculated and then one is added.

The main disadvantage of this approach is that it affects the performance when the size of the table is large as mostly in warehouse. Also it gives uniqueness within single table. It can be problematic after deletion of last row because its value can be reused for next record [24]. It causes serious problem in concurrent environment when concurrent transaction read the same value for the next record [20]. So this algorithm is best suitable for low intensity applications. The solution given by Mayer is "**Enhanced Max Plus One**" algorithm in which the deficiency of Max Plus One algorithm is improved by including the keyword "**hold-lock**" in the query [10]. However, this algorithm also causes deadlocks in a highly concurrent environment.

Another approach to generate surrogate primary key is the "**Composite Key**" devised by Meyer [10]. In this concept two approaches are applicable including one column with multi part and other is combination of more than one columns. In first approach, the first part is the object id which controls the inserts into the table and the second part is datetime which ensures uniqueness. The main disadvantage of this approach is the size of the key and the other is, no mechanism to check and control the gaps generated between keys. This method also degrades the performance of DWH Process [28].

Joe Celko devised an approach to build a "**Separate Table**" that holds the next value to be generated. The table has single column usually integer type for that storing next value. Each time to generate SK a SP is invoked to get next value and then update that table [9]. The main advantage of this approach is the high portability [4].

This approach causes gaps between numbers when operated between concurrent transactions and also slows their processing by causing Hot-Block problem [29].

Another technique described by Roy Hann is use of a **Pseudo Random Number Generator** [15]. The values generated by this technique are the integers in random order not random number. The repetition of numbers does not appear for billions of values. A next seed table is used get the key value for the generator. This key value is hidden. The main advantage of this technique is that the keys are distributed uniformly. The other advantage of this technique is the security in sense that next value can not be predicted and it is also useful in concurrent environment. The disadvantages of random surrogate keys include cyclicity in a sense it generates duplicates, size and complexity [9].

Surrogate keys can also be generated and inserted in the DW by **DB Triggers**. The main disadvantage of triggers is the severe bottleneck created in ETL [26].



According to Chuck Ballard and Daniel M. Farrell, GUIDs can also be used as surrogate keys [8][14]. GUIDs are Microsoft's standard. The technique is hashing the current datetime with the MAC address of network card to produce the key value. In the absence of network card, the software id is used. The main advantage of this technique is the uniquely generation of SKs over each table. It also allows the integration of data comes from different sources without any clash. It is helpful in replication. GUIDs increase the performance by reducing the joins [14]. It is highly portable [17] and hot spot can be avoided.

Some authors suggest to avoid the usage of GUIDs in DWH due to several reasons. The main reason is storage [8]. GUIDs takes a large amount of space than integers. The other reason is that as GUIDs are four times larger than integer key so indexes on GUID columns are slower integer keys. The process of debugging GUIDs is hard task and these are updateable [18].

UUID is another method to generate the surrogate key, described by Markus Zywitza and Scott W. Ambler [5][7]. Open Software Foundation create the algorithm for this approach. UUID is 128 bit value. This value is generated by a hash of the datetime of the currently used system and network card id. In the absence of card, an equal software representation is used. There are two representations in UUIDs, **hex** and **compact** [5]. In hex representation the id is converted and represented by hex values and string is made which is readable. In compact representation the id is converted into array of byte and then represented into string.

The disadvantage of this method is same as GUIDs. It includes large size, slower indexing and hard debugging.

According to Joseph Sack Surrogate Keys can be generated and managed by using **ROWGUIDCOL** property columns which provides the uniqueness at very high level [11]. This is very useful in heterogeneous environment. The main limitation of this technique is that a table can have only one rowguidcol column.

The integrity of data of column can be attained by the use of constrains in SQL Server as devised by Joseph Sack [11] and also can be used to generate SKs. It includes **Unique Constraint** and **Default Constraint**. Unique Constraint can be used to generate distinct values on non key columns and Default Constraint can be used on column where data type is not known. The major drawback of these techniques is the duplication in concurrent transaction environment when clock is set to backward.

Ralph Kimball and Joe Caserta have described another technique to generate the SK for DWH by concatenation of system's **natural key with datetime stamp** [26]. It also describes the insertion time of particular record.

There are various drawbacks of this technique. First is the dependency on source system. Secondly it is only supported in Homogeneous Environment. The major drawback is the less control, huge size and degraded query performance.

Scott W. Ambler describes another technique named High/Low strategy [7]. According to this strategy, any key is logically consists of two parts i.e. High and Low. High value comes from the source and Low is n digit value that is

assigned by the application itself. The procedure is as on obtaining high value every time, the low value is set to zero. If the high value is 101 and n=3 then the key will be 101000,101001, 101002 and so on. If the high get the value 201 and n= 3 then the sequence will be 201000, 201001 and so on.

An approach which is useful in concurrent environment is the **Recycled Series** described by Gary Meyer [10]. In this approach every process in concurrent transaction environment has conceptually next key table with specific range. For example one process has its range from 101 to 1000 and other would have 1001 to 2000. This approach is most effective in insert intensive scenarios where cache hit ratio and concurrency can affect the performance.

### B. Software Agents

Software Agent is a stand alone computer based program works for predefined tasks in dynamic situation on behalf of user or any entity. It can work without any control for an extended period [23][12]. In a Multi Agent System, there are more than one software agents and they can interact with each other in different scenarios including competitive, cooperative or autonomous. They can produce results for entities that initiated them and these entities can also terminate their instance. They can have a user interface [1]. A software agent can run manually by user in foreground or automatically in background. An agent call other agent, they are portable and can be replicated, serve for specific task. Other characteristics include persistence, reactive, social/collaborate and flexible. The application area of software agents includes data presentation, event notification, pattern recognition, data collection, data sorting and filtering, optimization and planning. Agent Technology is used rapidly in data warehousing with growing size of DWH.

## III. PROBLEM

After conducting an extensive literature survey we conclude that all the current methods to generate the surrogate keys have some limitations. First of all few techniques generate gaps in the records listing, which results in confusions and de-organization of data. Secondly the issue of concurrent access is not dealt efficiently in few of the techniques. Two concurrent records may obtain the same surrogate key. In few of the techniques the algorithms designed to handle the concurrency results in deadlock. The deadlock occurs due to the combination of shared lock and exclusive lock mechanism. So due to the chances of deadlock these techniques can only be efficient in low intensity environment. The size of the key is also one of the main issues that need to be dealt properly. In few techniques the size of the generated keys becomes bigger which indirectly increases the storage overhead and results in searching delays. Few techniques required the value of the next seed. If multiple transactions are allowed to access and retrieve the value of the next seed then multiple transactions can get same seed value and if only one transaction is allowed to get the value of the next seed then the delay arises.

As data come from different sources in DWH, there is no method available, to our best of knowledge, to keep the track

of a record that it belongs to which system. The reason may be the reformation of data during ETL before loading of data into WH.

Methods which generate the SK automatically by DWH with DBMS supported type are considered best. The main problem with these types of method is the mismatch of values generated in replication and the originally in DWH.

The major cost of SK is that it places huge burden on the ETL system. First SK is assigned to each distinct dimension row. Secondly substitution is made on keys of transactional systems in fact table with SK from DT [22].

Till now, agents are used in DW system for reporting purposes including trend checking, event notification, pattern recognition etc. They can be used to maximize the performance of ETL process by sharing the load of processing and calculations.

As there are many components of ETL which are required to be considered for advancement but we have selected Surrogate Key Generation for our focus point. There are many techniques available but we are going to propose a framework to generate surrogate keys through multi agents. Now generation of unique number across the DT is the duty of agents rather than DBMS itself. Agents are invoked and terminated by the WH system. By using the advantages of multi agents, the limitations of prior method to generate keys can be resolved and burden on ETL system can be reduced by dividing the task between agents. Tracing the source of the record present in DT is a major advantage of using this technique, which was not possible in prior methods (to the best of our knowledge).

#### IV. PROPOSED APPROACH

We start with discussing the base on which idea is inherited, SK generation stage, agent technology, proposed framework architecture in detail and consequences of solution.

##### A. Agent Technology

Software Agent is a stand alone computer based program works for predefined tasks in dynamic situation on behalf of user or any entity. It can work without any control for an extended period. In a Multi Agent System, there are more than one software agents and they can interact with each other in different scenarios including competitive, cooperative or autonomous. The major characteristics of agents include persistence, reactive, social/collaborate and flexible. The application area of software agents includes data presentation, event notification, pattern recognition, data collection, data sorting and filtering, optimization and planning.

Agent Technology is used rapidly in data warehousing with growing size of DWH. We have used multi agents for SK generation for the WH. They reside on separate layer and interact with WH in order to get datarow and then return the key for that row.

##### B. Data Sharding

Data warehousing is used very commonly for last few years due to size of application database and huge volume of

transactions. Google engineers devised the term sharding. It is "shared nothing" partitioning approach in which huge volume of database is partitioned into smaller *shards* and then distributed over many servers for achieving scalability, throughput and performance increase.

We have inherited the idea of database sharding with little change to our proposed solution. Instead of dividing whole database, its all related tasks and distributing it on other server, we distribute the task of surrogate key generation to multiple agents interacting with database application. The burden of generation and management of SK is now responsibility of agents. Agents are invoked and terminated by DB application itself. During the time to key generation DBMS can perform other task without depending on the return of generated keys.

In staging area agents are invoked to generate SK for DWH. Agents being held on separate layer, they can interact with DW system through this staging area.

##### C. Increasing ETL Performance

In common practice, the data loading of ETL process is the slowest phase due to index creation, integrity maintenance and concurrent environment. So bulk loading operations can be used to accelerate ETL process but it can also create bottleneck during access to database. The ETL performance can be increased by partitioning table in small size and hence indexes, disabling validation, integrity checking and triggers during load process on target DB, generate ids in ETL instead of database, dropping index before load process and creating after load process in DWH, using bulk loading in parallel and minimizing dependency of jobs to each other [3].

Most authors suggest that ETL processing should be done outside the DB in order to get higher performance. For example, by using *distinct* we can remove duplicates from the table but it is slow in DB so we can perform it outside the DB. We have proposed the solution based on it with a change that key generation is done outside the DB and return back to it.

##### D. Parallelism in ETL

The performance of ETL process can be improved by implementing parallelisms. It can be achieved in three ways by splitting a large table into small chunks for parallel accessing, by pipelining multiple components on same dataset and by pipelining several components on different dataset [6].

We in our proposed solution provide parallelism by dividing a data row from selected dataset into chunks according to subject area. Then agents automatically generate keys according to their subject based chunks and then loaded into the DW.

##### E. Proposed Architecture

We have proposed the said technique named **Generating Surrogate Keys through Multi Agent** after comparing all the prior methods. By using this technique cost, bottleneck, deadlock, hot spot and complexity can be reduced and performance can be maximized.

As data modeling for WH is dimension modeling and hence dimensions and fact tables are maintained within DWH.

Surrogate keys are also called primary keys of DW. They are the part of the dimension table and the replacement of the keys coming from the source system, either from homogeneous or heterogeneous environment and also referenced into the fact table as foreign key. According to proposed architecture there is MAS layer in contact with staging area (staging database). Multiple agents reside on MAS layer. In staging area, cleaned and transformed data is stored for temporary basis and data is going to load into the different dimensions. As the numbers of dimensions to be made in DWH are known, so one software agent is reserved for each dimension in MAS layer to generate key for it. Agent will guarantee the generation of unique key against each source system key. Each agent will add a new surrogate column to the data collected at staging database on staging area of ETL layer.

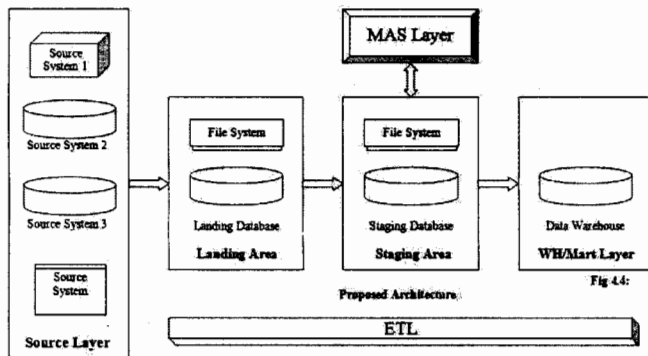


Fig 1: Proposed Architecture

#### F. Methodology

Information about each agent and its concerning is stored at central point in MAS layer. Information about agent includes agent id, unique id and start value where as it's concerning information includes source system id, database name, schema name, table name and the primary key field(s).

Surrogate Key is made by concatenating the unique id of particular agent with start value. Hence a unique number is generated. For example if an agent unique id is 100 and next value is 1 then the key will be 1001 and it will be the desired key for the dimension table. After generating the number it will return this number to the calling module in data staging area and then it immediately increments the start value by one for next generation of key.

Agent id differentiates each agent with other present in MAS layer. Unique id is the prefix used for keys which shows that this key is generated by which agent and hence its dimension and source can be easily found. Start value is the seed value which will increment sequentially by agent.

#### G. Agent Architecture

An agent is present against each dimension table which is responsible for the generation of the surrogate key against each record for that particular dimension. Being autonomous system, agents are not part of the DBMS but can communicate with it. When the data is ready to be loaded into the dimension of the data warehouse, a request of surrogate key value is made against each record of the cleaned data resides in data

staging area. Agents are constantly listening to the request for the next key value. After the request is made by ETL module, the particular agent according to the contents of data is invoked and key is returned.

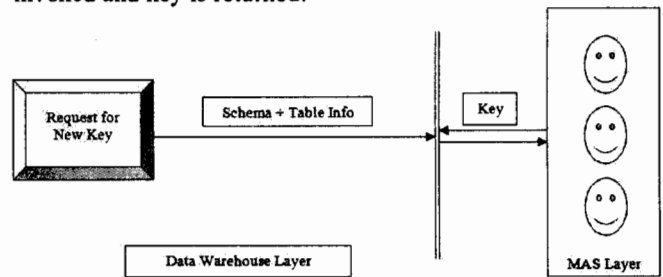


Fig 2: Agent Architecture for Key Generation

#### H. Abstract Flow of Framework

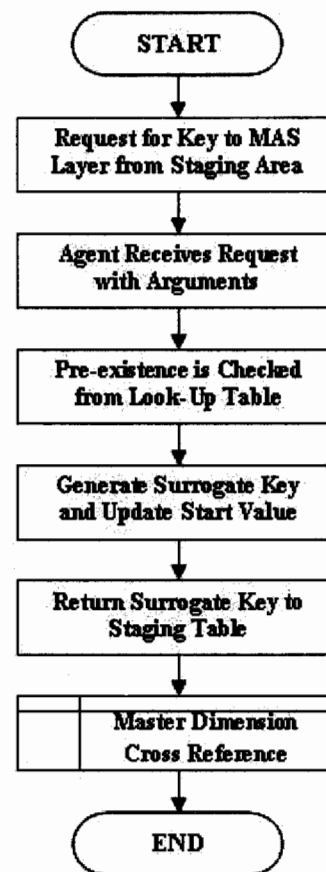


Fig 3: Abstract Flow of Proposed Framework

#### I. Major Capabilities of Proposed Framework

The main purpose of our proposed framework is to remove the maximum issues involved in existing SK generation methods. Our framework will be distinguished from other methods in the following ways:

- 1) It will provide the mechanism to trace the source of a particular record present in DWH.
- 2) The resultant keys will be generated without gaps.
- 3) It is also portable and supports the replication and will

give effective and accurate results.

- 4) Cost and performance will be optimized due to reduced size of the surrogate keys in fact table.
- 5) The bottleneck created due to the deadlock in processing huge amount of data (TB of data) in warehouse will be eliminated.
- 6) It will support the surrogate key generation process in an intensive concurrent environment.
- 7) Hot spots created by concurrent transactions will be eliminated.
- 8) Number of I/O operations will be minimized.
- 9) The load on ETL layer and DBMS in managing almost terabytes of data and keys will be reduced.

## V. TESTING AND PERFORMANCE

We have developed an application based on framework in Visual Studio .Net 2008 with SQL Server 2005 as the DBMS. Our Demo source data is the PTCL Telephone Directory which is created in MS Access 2003 Database. Our application is basically web based and agent framework used for this project is "SAGE" which is open source created in JAVA. SAGE has its own GUI for creating and managing agents. Instead of running separate application for SAGE, we have inherited the Agent creation and managing which includes killing, suspending, resuming and sending messages process in our application and made our own module.

We have presented the side by side comparison of the proposed solution with other existing technologies using following parameters and test cases.

### A. Source System Tracing

As data come from different sources held either homogeneous or heterogeneous in data warehouse and after ETL process where cleansing and transformation are performed to make data consistent, so there is no method, to the best of our knowledge, to trace out the source of data from which it has come into the DWH. In our proposed method, as agents generate keys for the DWH record and its three digit id is used as prefix of the generated key, so we can trace the source of the record in DWH from the key because information about agent id, source system name, database name, schema name and table name are stored at central location against each agent.

**Table 2a: Keys generated by Ordinary Methods**

Name Id	Title	First Name	Middle Name	Last Name
101	Syed	Mansoor		Khan
102	Syed	Shabir		Ali
103		Tahir		Iqbal
104		Tariq	Naseem	Qureshi
105	Khan	Usman		

**Table 2b: Keys generated by Multi Agents**

Name Id	Title	First Name	Middle Name	Last Name
1001	Syed	Mansoor		Khan
1002	Syed	Shabir		Ali
1011		Tahir		Iqbal
1021		Tariq	Naseem	Qureshi
1012	Khan	Usman		

In table 2a keys are numeric and sequential but seem to belong to a single source. There is no distinction that record is

either come from one source or many sources. In table 2b keys are also numeric and sequential to some extent but there exists a distinction of record that it comes from which source. First three digits are the prefix added to the key and they show the agent id. In above table key "1001" means that it is generated by an agent with id "100" with sequence number "1" and same is the case with keys "1011" and "1021". Hence records with keys "1001" and "1002" belong to one source, keys "1011", "1012" belong to second and "1021" belong to third source. it is not compulsory that keys come from different sources, they can be from different databases on single source, or different schemas in single database, or different tables in single schema.

### B. Gaps between Sequentially Generated Numbers

The auto-increments key like Identity method is considered to be most efficient methods for key generation for DWH. But the main disadvantage with this method is gaps created between sequentially numbers. As the volume of DWH data is in TB (Tera Bytes), so there can be huge number of missing sequences wastage. There are four major caused of gaps generation within numbers.

- 1) Abortion of DML Statements (Automatically Role Back).
- 2) Deletion of Rows.
- 3) Resetting Seed Value.
- 4) Rolling Back Transactions Manually.

In our proposed method the gaps between sequentially number are handled by agents itself. So it can be controlled easily.

**Table 3a: Keys generated by Ordinary Methods**

Name Id	Title	First Name	Middle Name	Last Name
101		Yasir		Habib
102		Zaheer		Haider
110		Zahoor		Ahmed
111		Zulfiqar		Ahmed
125		Zulqurnain		Akhter

**Table 3b: Keys generated by Multi Agents**

Name Id	Title	First Name	Middle Name	Last Name
1001		Yasir		Habib
1002		Zaheer		Haider
1003		Zahoor		Ahmed
1011		Zulfiqar		Ahmed
1012		Zulqurnain		Akhter

Table 3a shows the key generation by ordinary methods. By going through the "Name Id" field, the gaps between numbers are very usual. This drawback is overcome by using agent based method and it can be seen from table 3b. There are two sequences in our sample data generated by two agents but without gaps.

### C. Replication

Replication is very important with administration point of view. As DWH holds historical data, so its back is very necessary for accidental situations. Replication can be used for backup purposes. In replicating the data table, not whole image is copied but insert one by one each row of the table. So tables having columns with identity property to generate numbers are generated again. There can be a situation where the keys in original table are not generated in the same order

in replicated place as they were in source system. We can explain it with example using tables as:

**Table 4a: Keys generated by Ordinary Methods in Source**

Name Id	Title	First Name	Middle Name	Last Name
101		Yasir		Habib
102		Zaheer		Haider
103		Zahoor		Ahmed
104		Zulfiqar		Ahmed
105		Zulqurnain		Akhter

Table 4a shows the sample data of a table having "Name Id" as Identity field and it is the five records in the source system. In replicating this data there can be possibility that Identity field generates value with other seed value rather than as in source system.

**Table 4b: Keys generated by Ordinary Methods in Replication**

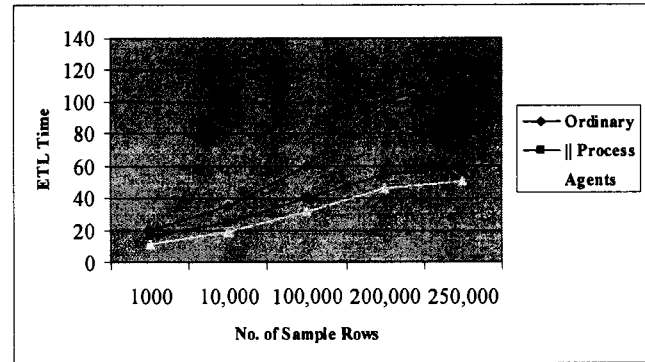
Name Id	Title	First Name	Middle Name	Last Name
110		Yasir		Habib
111		Zaheer		Haider
112		Zahoor		Ahmed
113		Zulfiqar		Ahmed
114		Zulqurnain		Akhter

Table 4b shows the key sequence of sample table after replication. By comparing both tables, it is clear that in replication DBMS has different seed value for that table. So there is obvious difference between keys of both systems. As surrogate keys are meaningless keys and they are invisible to users but still they are meaningful by DBMS to manage integrity between dimensions and fact tables as well as look ups. By using our agent based approach this problem can be overcome because agents are customized so we can reset its starting value easily when deploying replicated table. Using agents for key generation both tables at source and replication point would be same with respect to keys order.

#### D. Agent based Framework Performance

To test our technique we have run our proposed framework based demo application with the sample data. As we have used three test scenarios so in order to check the performance we have taken result for same sample data using ordinary key generation methods and key generation through parallel process.

Fig 4 shows the total time required by ETL process to extract data from source system to load data into DWH. Here we have taken different samples of data from source system to check against three scenarios. As clear from fig 4 that ordinary methods take too time by comparing others. So it can be improved by dividing the tasks and transactions into parallel processes. But using our agent based approach there is a significance performance difference against other methods. The main reason is that it is also using parallel processes but the task of generating keys is now performed by agents not by DBMS itself. During key generation time DBMS can do other tasks. So we can enhance the efficiency by reducing the burden from DBMS.



**Fig 4: Time taken by Each Method for ETL**

Other important observation from fig 4 is that as the volume of sample data is increasing, the time for ETL is not increasing with same factor. Hence it can be concluded that as volume of data increase in ETL, their time for processing reduce due to caching.

## VI. CONCLUSION AND FUTURE WORK

Agents are used for DWH for out put i.e. reporting and multi level analysis purposes which includes sorting, filtering, alerting, threat identification in business and monitoring related tasks. We have used and implement agents for surrogate key generation and we have taken extra ordinary performance enhancing benefits from it by comparing with other methods. So we conclude that using agents during DWH design and implementation is a good practice for developers.

### ACKNOWLEDGMENT

We pay thanks to Higher Education Commission of Pakistan and International Islamic University Islamabad for their cooperation. Without their financial aid and resources this research work was not possible.

### REFERENCES

- [1] John W. Krupansky, "What is a software agent?", January 2006
- [2] Oracle Corporation, "Automatic Primary Key Generation", 2008.
- [3] Lev Selector, "ETL - Extract, Transform, Load", 2007
- [4] Oracle Corporation, "How to Configure Primary Key Generation", December 2006.
- [5] Markus Zywitz, "Primary Key Mapping", February 2009.
- [6] <http://www.wikipedia.com>
- [7] Scott W. Ambler and Pramod Sadalage, "Choosing a Primary Key: Natural or Surrogate", 2005.
- [8] Chuck Ballard, Daniel M. Farrell, Amit Gupta, Carlos Muzuela and Stanislav Vohuik, "Dimensional Modeling: In a Business Intelligence Environment", March 2006.
- [9] Joe Celko, "Data and Databases: Concepts in Practice", 2001.

- [10] Gary Meyer, "How Surrogate Primary Key Generation Affects Concurrency and the Cache Hit Ratio", Aug 21, 1998.
- [11] Joseph Sack, "SQL SERVER 2005 T-SQL RECIPIES", 2006.
- [12] IBM, "Agents", January 2009.
- [13] Vincent McBurney, "Why database generated surrogate keys drive me nuts", May 25, 2006.
- [14] Sameer, "GUID Or Int Primary Key", June 25, 2007.
- [15] Roy Hann, Rational Commerce Limited, "Key Points About Surrogate Keys", August, 1996.
- [16] Alkis Simitsis and Dimitri Theodoratos, "Data Warehouse Back-End Tools", 2005.
- [17] Jeff Atwood, "Primary Keys: IDs versus GUIDs", March 2007
- [18] ABC, "What should I choose for my primary key", June 2005.
- [19] Cesar A. Galindo-Legaria, Torsten Grabs, Sreenivas Gukal, Steve Herbert, Aleksandras Surna, Shirley Wang, Wei Yu, Peter Zabback, Shin Zhang, "Optimizing Star Join Queries for Data Warehousing in Microsoft SQL Server", 2008 IEEE.
- [20] Jason Zhang, "How to implement a DB2 UDB primary key with a surrogate key", July 2004.
- [21] Lee Richardson, "Surrogate vs Natural Primary Keys", August 2007.
- [22] Joy Munday and Warren Thornthwaite, "The Microsoft Data Warehouse Toolkit", 2006.
- [23] Paulraj Ponniah, "Data Warehousing Fundamentals", 2001
- [24] Scott W. Ambler, "Agile Database Techniques: Effective Strategies for the Agile Software Developer", 2003.
- [25] Sharon Allen and Evan Terry, "Beginning Relational Data Modeling", 2005.
- [26] Ralph Kimball, Joe Caserta, "The Data Warehouse ETL Toolkit", 2004s
- [27] McDonald, Andreas Wilmsmeier, David C. Dixon, W.H. Inmon, "Mastering the SAP Business Information Warehouse", 2002
- [28] Claudia Imhoff, Nicholas Glemmo and Jonathan G. Gaiger, "Mastering Data Warehouse Design", 2003
- [29] Gavin Powell, "Beginning Database Design", 2006.

## REFERENCES

## **REFERENCES**

- [1]. Pwosboy, "A Pattern for Primary Key Generation", August 2007.
- [2]. Oracle Corporation, "Automatic Primary Key Generation", 2008.
- [3]. Troels Arvin, "Comparison of different SQL implementations", March 2009.
- [4]. Oracle Corporation, "How to Configure Primary Key Generation", December 2006.
- [5]. Markus Zywitza, "Primary Key Mapping", February 2009.
- [6]. Andrus Adamchik, "Primary Key Generation", 2008.
- [7]. Scott W. Ambler and Pramod Sadalage, "Choosing a Primary Key: Natural or Surrogate", 2005.
- [8]. Chuck Ballard, Daniel M. Farrell, Amit Gupta, Carlos Muzuela and Stanislav Vohuik, "Dimensional Modeling: In a Business Intelligence Environment", March 2006.
- [9]. Joe Celko, "Data and Databases: Concepts in Practice", 2001.
- [10]. Gary Meyer, "How Surrogate Primary Key Generation Affects Concurrency and the Cache Hit Ratio", Aug 21, 1998.
- [11]. Joseph Sack, "SQL SERVER 2005 T-SQL RECIPIES", 2006.
- [12]. Pete Stiglich, "Performance benefits of surrogate keys in Dimensional Models", 2007.
- [13]. Vincent McBurney, "Why database generated surrogate keys drive me nuts", May 25, 2006.
- [14]. Sameer, "GUID Or Int Primary Key", June 25, 2007.
- [15]. Roy Hann, Rational Commerce Limited, "Key Points About Surrogate Keys", August, 1996.
- [16]. Alkis Simitsis and Dimitri Theodoratos, "Data Warehouse Back-End Tools", 2005.
- [17]. S. M. Deen, "An Implementation of Impure Surrogates", September 1982.
- [18]. Neoklis Polyzotis, Spiros Skiadopoulos, Panos Vassiliadis, Alkis Simitsis, and Nils-Erik Frantzell, "Meshing Streaming Updates with Persistent Data in an Active Data Warehouse", July, 2008.
- [19]. Cesar A. Galindo-Legaria, Torsten Grabs, Sreenivas Gukal, Steve Herbert, Aleksandras Surna, Shirley Wang, Wei Yu, Peter Zabback, Shin Zhang, "Optimizing Star Join Queries for Data Warehousing in Microsoft SQL Server", 2008 IEEE.
- [20]. Brian Walker, "Why Use Surrogate Keys", January 2006.



- [21]. Lee Richardson, "Surrogate vs Natural Primary Keys", August 2007.
- [24]. Graeme C. Simsion, Graham C. Witt, "Data Modeling Essentials", 2005.
- [25]. Sharon Allen and Evan Terry, "Beginning Relational Data Modeling", 2005.
- [26]. John V. Petersen, "Absolute Beginner's Guide to Databases", 2007.
- [28]. Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy and Bob Becker, "The Data Warehouse Lifecycle Toolkit", 2004.
- [29]. Gavin Powell, "Beginning Database Design", 2006.
- [30]. Claudia Imhoff, Nicholas Gallemmo and Jonathan G. Gaiger, "Mastering Data Warehouse Design", 2003.
- [31]. Beixin (Betsy) Lin, Yu Hong, Zu-Hsu Lee, "Data Warehouse Performance", 2005.
- [32]. Kevin McDonald, Andreas Wilmsmeier, David C. Dixon, W.H. Inmon, "Mastering the SAP Business Information Warehouse", 2002.
- [33]. Ralph Kimball, Joe Caserta, "The Data Warehouse ETL Toolkit", 2004.
- [34]. Scott W. Ambler, "Agile Database Techniques: Effective Strategies for the Agile Software Developer", 2003.
- [36]. Paulraj Ponniah, "Data Warehousing Fundamentals", 2001.
- [38]. Djoni Darmawikarta, "Dimensional Data Warehousing with MySQL", 2007.
- [39]. Ralph Kimball, Laura Reeves, Margy Ross and Warren Thornthwaite, "The Data Warehouse Life Cycle Toolkit", 2005.
- [40]. Ralph Kimball and Margy Ross, "The Data Warehouse Toolkit", 2002.
- [41]. Joy Munday and Warren Thornthwaite, "The Microsoft Data Warehouse Toolkit", 2006.
- [42]. Jason Zhang, "How to implement a DB2 UDB primary key with a surrogate key", July 2004.
- [44]. Joshy George, "Surrogate Key Generation in DataStage", February 2008.
- [45]. [www.DWHInfo.com](http://www.DWHInfo.com)
- [46]. Brian Krow Aker, "Myths, GUID vs Autoincrement", March 2007.
- [47]. Donald M. Farmer, "Surrogate Key Generation and Utilization", May 2007.
- [48]. ABC, "Why are there gaps in IDENTITY / AUTOINCREMENT column", March 2005.
- [49]. ABC, "What should I choose for my primary key", June 2005.

- [50]. Jimmy Nilsson, "The Cost of GUIDs as Primary Keys", March 2002.
- [52]. AgileWare, "Performance Comparison - Identity(), NewId(), NewSequentialId", January 2009.
- [53]. Priya Dhawan, "Performance Comparison: Data Access Techniques", January 2002.
- [55]. Jeff Atwood, "Primary Keys: IDs versus GUIDs", March 2007.
- [59]. Van Scott, "Extraction, Transformation, and Load Issues and Approaches", January 2000.
- [61]. IBM, "Agents", January 2009.
- [62]. Oracle Corporation, "Business Intelligence", 2007.
- [63]. <http://www.wikipedia.com>.
- [64]. Lev Selector, "ETL - Extract, Transform, Load", 2007.
- [65]. John W. Krupansky, "What is a software agent?", January 2006.
- [66]. Thomas Connolly and Carolyn Begg, "Database Systems, 3<sup>rd</sup> Edition", 2003.
- [67]. Fred R. McFadden, Jeffrey A. Hoffer, "Modern Database Management, 4<sup>th</sup> Edition", 1994.
- [68]. Hafiz Farooq, "Persistent Architecture for Context Aware Lightweight Multi Agent System", the Fifth International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2006), Japan, 1994.

