Chapter 3

Framework of Hand Coded ETL in Data Warehouse Architecture and Strategy for Synthetic Data Generation

3.0 INTRODUCTION TO DATA WAREHOUSE

The term data warehouse can be stated as a collection of technologies that aims at enabling the knowledge worker to make better and timely decisions by exploiting the integrated knowledge acquired from across the organization. According to the most admired definition of data warehouse given by W.H. Inmon, “A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data used to support the strategic decision-making process for the enterprise”. Almost a decade and a half of research has been spent on the study of data warehouses, especially for its design and exploitation for decision-making purposes. During this course of time the data warehouse evolution has witnessed five major phases of development. Figure 3.1 captures the details of all the five phases of the data warehouse evolution.

Figure 3.1 The evolution of Data Warehouses
3.1 EVOLUTION OF DATA WAREHOUSES

From 1970 onwards, the global economy demanded a revolution in the decision making process to gain competitive advantage over rival organizations. The ever changing business rules and environment has made the decision making process both difficult as well as important. Hence to make accurate decisions there was a need to convert raw data into meaningful information, which may further be used for decision making process. The evolution of data warehouse has witnessed five major phases of development, which are discussed as under:

a) **First Phase:** The initial phase of data warehouse deployment concentrates on report generation from a single source of true-life data within an organization. In such a reporting environment generally the questions are known in advance. Thus, database structures can be optimized to dispense good performance even when queries require access to gigantic amounts of information.

b) **Second Phase:** In the second phase of data warehouse development the decision makers focused on Why? Instead of what happened? The uniqueness of second phase includes ad hoc queries and analysis. This means Questions against the database cannot be known in advance. Here performance relies primarily on advanced query optimizing capability of the RDBMS because query structures here can not be as predictable as they were in a pure reporting environment.

c) **Third Phase:** At this stage the need of some data mining tools was felt to develop predictive models for strategic planning. Such models were based on recorded historical facts of business organizations. Development of such business model typically involves derivation of hundreds of complex metrics.
d) from thousands (or more) of recorded observations. These derivations are required for training the predictive algorithms to achieve business objectives.

e) **Fourth Phase:** This phase focuses on premeditated decision support. Unlike applications for strategic planning which supports market segmentation, product category management stratagems, profitability analysis and demand forecasting, premeditated or tactical decision making is concerned with providing access to information for immediate decision-making in the concerned field.

f) **Fifth Phase:** This phase advocates the use of data warehouse in areas where the humans are unable to add significant value in the process of decision making. For example the use of automated decision making systems is inevitable for e commerce business. An active data warehouse serves information to enable decision support throughout an organization rather than being confined to strategic decision-making processes. However, tactical decision support does not replace strategic decision support. Rather, an active data warehouse supports the coexistence of both types of workloads.

### 3.1.1 Data Warehouse Architectures

Data warehousing has evolved from a simple theoretical vision to a very complex entity full of the variations and subtleties that mark the real world. Data warehouse history portrays two widely accepted definitions of data warehouse tossed by Ralph Kimball and W.H. Inmon. These two pioneer personalities have different opinion in defining and developing a data warehouse. According to Ralph Kimball “the data warehouse is a copy of transaction data specifically structured for query and analysis”[83]. The term data warehouse was originally coined by Inmon in 1990 and he defined the data warehouse as a subject-oriented, integrated, time-variant and non-
volatile collection of data in support of management’s decision making process”[104][105]. The architecture proposed by Inmon and Kimball is shown in figure 3.2 and 3.3 respectively. Table 3.1 presents a brief comparison of both these ideologies.

<table>
<thead>
<tr>
<th>Kimball Approach</th>
<th>Inmon Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Everyone is allowed to fabricate their database according to their requirements and department structure. All these independent repositories can be integrated as and when required. This methodology is known as bottom up approach</td>
<td>Inmon supports a top down approach. Here no one is allowed to develop any database independently. The database for an organization should be planned and designed centrally. Every department within the organization will follow the centrally designed schema to fabricate their database.</td>
</tr>
<tr>
<td>2. This structure is easier to build</td>
<td>The structure proposed is very typical one to craft.</td>
</tr>
<tr>
<td>3. It is a nimble approach</td>
<td>Rigorous analysis and designing is required.</td>
</tr>
<tr>
<td>4. Problematic to maintain as an enterprise resource</td>
<td>Easier to maintain as an enterprise resource.</td>
</tr>
<tr>
<td>5. Data is often redundant</td>
<td>Redundancy is regulated to a great extent.</td>
</tr>
<tr>
<td>6. Very difficult to integrate independent data marts with varying structure</td>
<td>Integration of data marts is comparatively easier.</td>
</tr>
<tr>
<td>7. This approach is flexible</td>
<td>This approach is comparatively rigid.</td>
</tr>
</tbody>
</table>

Table 3.1 Kimball Vs Inmon’s Ideology

Data warehouse literature identifies five different types of data warehouse architectures which can be stated as derivations of the primary data warehouse

- 56 -
definitions proposed by W.H. Inmon, who is known as the father of data warehousing and Ralph Kimball. The derived data warehouse architectures possess some unique characteristics and have advantages and disadvantages of their own. The varying organizational structures, ever changing business rules and disparity in technical standards among business houses has given birth to these architectures [40]. A brief introduction to these architectures is as follows:

![Data Warehouse Architecture as Proposed by W.H. Inmon](image1)

**Figure 3.2 Data Warehouse Architecture as Proposed by W.H. Inmon**

![Data Warehouse Architecture as Defined by Ralph Kimball](image2)

**Figure 3.3 Data Warehouse Architecture as Defined by Ralph Kimball**
3.1.1.1 Independent Data Marts

It is the most common data warehouse architecture among business houses. These data marts are developed and maintained independently according to the typical needs of a department within an organization. Although the data marts are dedicated to answer the set of queries from a specific department, but are unable to present a unified view of information. They typically provide inconsistent data definitions and use different dimensions and measures for representing similar entities. Because of this non conformed view of data it is very difficult to analyze data across the data marts.

![Figure 3.4 Independent Data Marts](image)

3.1.1.2 Hub and Spoke Architecture

This architecture has evolved iteratively and is committed to every department within the organization in a serial manner. Implementing hub and spoke architecture requires in-depth analysis of the business requirements. Normalized atomic data and dependent data marts are among the foremost characteristics of this architecture. These dependent data marts are generally developed for special purposes keeping in view the functional areas of the organization. The data mart may store normalized, de-normalized, summarized or dimensional data based on the requirements of the users.
Most of the user queries are handled by the local data mart and the central warehouse is rarely referenced where highly summarized data is placed.

![Figure 3.5 A Hub and Spoke Architecture](image)

### 3.1.1.3 Centralized Data Warehouse with no Dependent Data Marts

This is the most basic architecture. If we deduct dependent data marts from the hub and spoke architecture the resultant structure will be the central data warehouse. A central data warehouse stores atomic level data, Meta data and some summarized data along with some pre generated views of the stored data. The queries in this architecture are allowed to access relational data as well as the dimensional views.

![Figure 3.6 Centralized Data Warehouse Architecture](image)

### 3.1.1.4 Federated Architecture

This architecture basically recycles the existing decision support structures like operational systems, data marts, and data warehouses within an organization. Data is accessed from these sources according to the business requirements. The data is logically as well as physically integrated using shared keys, global metadata, distributed queries and other possible methods. This architecture is generally chosen
by those firms that have a pre-existing, complex decision support environment and do not want to redefine the same.

![Figure 3.7 Federated Data Warehouse Architecture](image)

**Figure 3.7 Federated Data Warehouse Architecture**

Data warehouses architectures are typically composed of two gears, the first gear is concerned with end-users who access the data warehouse with decision support tools, and the second gear is associated with the data warehouse administrators, responsible to populate the data warehouse with data obtained from divergent sources. The architecture of a data warehouse exhibits various layers in which data from one layer are derived from data of the previous layer. Although there are various data warehouse architectures in existence but the Extraction, Transformation and Loading procedures is indispensable for any architecture. Hence ETL can be stated as the heart and soul of data warehouse solutions. ETL processes bring together and combine data from multiple source systems into a Data Warehouse for providing a single version of the truth. The Data Warehouse Institute research presumes that ETL design and development consumes 60 to 80 percent of an entire business intelligence project. ETL routines reshape the relevant data from the source systems into useful information to be stored in the data warehouse. Compromising performance of ETL routines may result in substandard strategic information. The
fundamental reason for data warehouse development is to improve the quality of information within the organization. If the source data is not extracted correctly, cleansed, and integrated in the proper formats then the queries issued to the data warehouse may not provide optimal results. Keeping in view the importance of ETL procedure there is a need to understand the ETL framework in detail.

3.2 THE ETL FRAMEWORK

The Data Sources, also known as operational databases forms the first layer in data warehouse architecture; such sources may consist of structured data stored in On-Line Transaction Processing (OLTP) database and legacy systems or even in flat files. The next layer comprises the back stage part of a data warehouse, where the collection, integration, cleaning and transformation of data take place in order to populate the warehouse. A volatile storage known as Data Staging Area (DSA), employed for the purpose of data transformation, reconciliation and cleaning of incoming data streams plays a dominant role in this layer. The central layer of any data warehouse architecture is the global Data Warehouse. The global data warehouse layer keeps track of historical data resulted from the transformation, integration, and aggregation of detailed data found in the data sources. The last layer consists of the user querying the data warehouse using data mining and OLAP tools [74] [76].

The Data warehouse operational processes are generally labour intensive workflows which constitute an integral part of the back-stage data warehouse activities. To deal with this labour intensive workflow and in order to facilitate and manage the data warehouse operational processes, specialized processes are used, under the generalized title they are known as Extraction-Transformation-Loading (ETL) processes. ETL processes are responsible for extracting data from several sources, their cleansing, their customization and transformation, and finally, their
loading into a data warehouse. Irrespective of the fact that these processes transpired a new rapidly growing market area, the formal modelling and management of these processes has not been adequately dealt by the research community.

The very first task performed during the ETL process, is the extraction of the relevant information that has to be further propagated to the warehouse. Generally a tight ‘time window’ in the night is exploited for capturing data from source systems because the source systems are either offline or are not heavily used during this time. In order to minimize overall processing time usually a fraction of source data that has changed or the newly inserted records since the previous execution of ETL process are captured. These changed or possibly updated records are detected by the physical comparison of the two snapshots, one corresponding to the previous successful extraction and other to the current extraction attempt. Such physical comparisons are generally made using snapshot differential algorithms or log sniffing techniques.

The transformation and cleansing routines are devoted to manage schema level and data level issues. Such issues include naming conflicts, structural conflicts, different aggregation levels and discrepancies in value representations. The integration and transformation routines further perform a wide variety of functions, such as reformatting, recalculating, modifying key structures, time stamping, filling up default values, interaction with multiple sources, synthesizing data from multiple sources on a staging area, etc. The loading phase of data warehouse imposes technical challenges of its own. A critical issue here is to discriminate between new and existing data at the time of loading. The simple open loop fetch techniques, where records are loaded one by one can not be deployed for data warehouse loading process because of their slow speed and lesser flexibility. Bulk loading of data arranged in batches seems to be an ideal option for refreshing data in a data warehouse.
Data warehouse architecture is absolutely dependent on customary data extraction, transformation and loading (ETL) routines for data synthesis. Although there are robust tools available for ETL processing still they aren't very pervasive with business organizations. The possible reasons for choosing hand coded solutions might include ignorance about market standards, lack of resources and heavy licensing costs associated with readymade tools. The ready to use ETL solutions may not provide tailored services according to the predefined corporate standards required by individual business houses. According to Forrester research and TDWI (The Data Warehousing Institute), the customized ETL solutions are still the best choice for data synthesis. The estimated percentage usage of various ETL options is shown in the figure 3.9. Presuming the popularity of custom built ETL solutions this research is dedicated to understand the development, functionality and quality assurance of such customized ETL solutions. The research started with comprehending the ETL design cycle. A typical ETL design cycle is shown in figure3.8.

![Figure 3.8 The ETL Design Cycle](image)

Thereafter the customary ETL design was crafted followed by the realization of the ETL design. The ETL design logic with the help of UML diagram is shown in figure 3.10.
Figure 3.9 Estimated Percentage Usage of various ETL options
Source: The Data Warehousing Institute (TDWI 2007)
Figure 3.10 UML Representation of ETL Design Logic
3.4 SYNTHETIC TEST DATA GENERATION

Business houses are investing heavily on new generation databases so as to gain competitive edge. Like any software development project, a data warehouse also needs testing. The complexity and size of data warehouse systems make comprehensive testing both “more difficult and more necessary”. The fact, queries that perform satisfactorily on small datasets may fail miserably in the real life environment. This necessitates establishing a system that runs queries on fully scaled data. Legal implications and business ethics do not allow performing testing with real business data [98]. Hence this research has made efforts to generate synthetic test data ensuring correct balance and skew making sure that the ratio of fact to dimension is correct and so on.

Habitual practice of testers to make use of real data, however, can violate a number of data privacy regulations and even customer fallout. For example, Health Insurance Portability and Accountability Act (HIPAA) 1996 of USA permits restricted access to people’s personal health data on a “need to know” basis. Also, many states in USA assure total privacy to even the personal information which includes things as a person’s name and address, date of birth, social security number, and credit card and bank account numbers. Even an initiative has been spearheaded by IT trade association NASSCOM to tone down fears about data security in India and to promote the region as the safest place for IT and BPO amid rising competition from other off shoring locations. It has constituted an autonomous body Data Security Council of India (DSCI) to further enforce data and information security laws in India. These regulations make no distinction between production and testing environments. Even then, to ensure performance the application developers still need an access to “good enough” data to test their applications. Accordingly, many
organizations are developing customized scripts, or purchasing off-the-shelf software, to transform sensitive production data into safe but usable test data.

3.3.1 How to Generate Test Data

To craft real data safe for the testing environment, the test-data generation procedure will need to formulate conversion strategies which should be application specific in nature. These transformations are difficult to implement because the generated data must epitomize the real data. For example, an e-commerce application may examine a PAN card number to see if it seems to be real. Similar checks may occur for birth dates, driver’s license numbers, addresses, bank accounts etc [17]. A number of techniques are used to generate synthetic data that resembles the real data and is safe for use also. Some of these techniques include:

a) **Encrypt and mask**: This technique either encrypts or masks the real data to hide its identity. Figure 3.11 depicts the encryption and masking procedure.

![Encryption and Masking Logic](image)

**Figure 3.11 Encryption and Masking Logic**

b) **Randomize**: Randomizing substitutes the real data like PAN card number, driver’s license, and other numbers with random numbers. Figure 3.12 corresponds to an MS-Excel based random data generator.
Figure 3.12 Synthetic Records with Random Values

c) **Scramble:** This technique is used to muddle names or numbers to guarantee that they’re not real, but are close enough to the real thing to work. A scrambled record is shown in figure 3.13

![Figure 3.13 A Scrambled Record](image)

**Figure 3.13 A Scrambled Record**

d) **Concatenate:** Preserve essential information but leave out, substitute, or randomize the remaining variables using a predefined routine (Refer Figure 3.18).
e) **Propagate:** For interdependent fields and related database tables, one should develop algorithms to propagate information and to ensure relationships, along with the safeguarding of key database references. For example if any age field is utilized to generate synthetic values then it should be in accordance with date of birth field as specified in original record.

f) **Look-up fields:** Replace an entry or value like names or addresses from a set of predefined list. Figure 3.14 presents the usage of look-up fields.

![Figure 3.14 Use of Look-up Fields](image)

Many companies use scripting tools or their test automation tools to conceal or replace sensitive information [12]. Organizations with inflexible data regulations are just beginning to deal with test-data problems and may use test data generation tools like Compuware, Datavantage, Global Software Applications, Princeton Softech, Quest Software, SoftBase Systems, DTM and Worksoft etc.

The aforesaid synthetic data generation tools are based on traditional test data generation techniques which have limitations of their own. For instance randomization is typically based on random function which may repeat toning patterns at fixed intervals in generated synthetic data. Test data generated through concatenation may lose vital information from testing prospective. Moreover using
off the shelf synthetic data generators may not allow declaring desired attributes for any entity. They have limited capabilities to exploit resembling domain sets available at different data sources.

3.3.2 The Data Set Generator

Taking into account the aforesaid constraints and because of non flexible behaviour of ready to use synthetic data generators it was decided to develop a dataset generator (DSG) that can generate authentic personal records database while preserving integrity constraints. The architecture includes two phases. In the first phase desired fact table is generated by including relevant facts from multiple sources. In phase two, dataset is synthetically puffed-up while preserving integrity, bias, correct balance and skew, fact to dimension ratio and so on. Figure 3.15, illustrates the proposed architecture of the DSG.

![Figure 3.15 Synthetic Data Set Generator Architecture](image)

The synthetic data set generator makes use of a data multiplication algorithm for generating synthetic records. Figure 3.16, depicts the working of Data
Multiplication Algorithm (DMA) used to puff up facts table by genetically mutating different available fields while preserving their integrity.

In the example shown in figure 3.16, first instance of the Name column is multiplied with third instance of FatherName column and second instance of MotherName column then the first instance of address field is multiplied with the second and third instance of course and date of birth column respectively to generate the first record given in the new record example. The process of puffing up dimension table can be extended by crossover multiplication of even new synthetic data sets generated till a sizable dimension table is obtained. Knowingly multiplicative crossover was preferred over the randomized crossover so as to avoid auto correlation and to ensure more independence and uniformity of mimicking data sets through data set generator.

![Figure 3.16 The Multiplication logic of DMA Algorithm](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Father Name</th>
<th>Mother Name</th>
<th>Address</th>
<th>Course</th>
<th>Date of Birth</th>
<th>Rollno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rem</td>
<td>P Kumar</td>
<td>Shanti Devi</td>
<td>H no 20 Urban Estate Pla. , India</td>
<td>MCA</td>
<td>22-6-1980</td>
<td></td>
</tr>
<tr>
<td>Bansi</td>
<td>Jashid Ali</td>
<td>Zinath</td>
<td>375 teddi 22 Basing India</td>
<td>MBA</td>
<td>30-11-1979</td>
<td></td>
</tr>
<tr>
<td>Annie</td>
<td>Roberts Sr.</td>
<td>Gironia</td>
<td>55 gandhi road calcutta</td>
<td>M.Tech</td>
<td>21-3-1981</td>
<td></td>
</tr>
<tr>
<td>Peter</td>
<td>Jordan Philip</td>
<td>Ginny Philip</td>
<td>#78 Dwarika area new Delhi</td>
<td>MCA</td>
<td>26-8-1980</td>
<td></td>
</tr>
<tr>
<td>Sophie</td>
<td>David Gill</td>
<td>Sheela Gill</td>
<td>54 Noah Road W Adel</td>
<td>M.Tech</td>
<td>11-4-1979</td>
<td></td>
</tr>
</tbody>
</table>
Unlike traditional test data generators the synthetic data set generator developed during this research is capable of exploiting resembling domain sets for different attributes at different locations. As shown in figure 3.17, the employee name and father name columns of employee table are mapped with first name and father name columns of student table respectively. Similarly date of joining can be mapped with date of birth field. As depicted in figure 3.18, the data set generator can club the resembling domain sets from two sources in a table placed on the virtual staging area.

The fields for which resembling domain sets are not available like in last name, mother name and course fields of student table can be filled with random values from the same source. Such randomization can not impose auto correlation as resultant records are generated with crossover multiplication. The tester can also specify the range for valid primary key values according to the need of the testing environments.

*Figure 3.17 The ability of DSG to exploit resembling domain sets at different locations*
As shown in figure 3.11. As the roll no field in the destination table is the primary key hence its value can be specified by the user. The flowchart depicting the working of data set generator is shown in figure 3.19.

In reference to figure 3.19, initially the user defines a targeted database schema and then selects a donor database to exploit its resembling domain sets for generating synthetic records. If the user is contented with the defined target schema then the data set generator can be instructed to generate a given count of synthetic records. Else the user can append desired new fields and redefine the target database schema through DSG interface. A different remote database can be used to feed values for the newly appended fields. Where possible, the user can also declare a range of valid values to be filled in the newly appended fields.
Figure 3.19 Flow Chart depicting the working of DSG

After the declaration process is over and data generation constraints are defined then the data set generator multiplies the records by exercising the logic discussed earlier. The multiplication process continues until the count of newly generated records matches with the count specified by the user. In the end the newly generated records are appended in the target database thus jacking up the size of the target
database by multiple times. This puffed up database can now be used to test a big
database application like a data warehouse.

Figure 3.20 shows the initial screenshot of data set generator interface. It
shows the default fields of the target test database. Here the user may change the name
or data type of the default fields according to identified testing requirements. The data
set generator is flexible enough to accommodate changes made by the user to target
database schema. Here the user can append or delete dimensions in the existing
schema as per the needs of the testing environment. The detailed system requirements
and usage of the data set generator are described as under:

3.3.2.1 System Requirements for the Data Set Generator

The data set generator is a Visual Basic 6.0 based application. Like any
computer program the data set generator too need some system resources. Following
are the minimum hardware and software requirements to successfully run the data set
generator.

Software Requirement

To run this software minimum software requirements are Windows 95 or
above. Other files/packages needed are pre included in the Data Set Generator
executable (DSG) file.

Hardware Requirement

a) Processor Intel Celeron 2 GHz or above
b) Ram 64 MB or above 256 Recommended
c) Hard disk 20MB of free space
d) Screen Resolution 1024*768 Recommended
3.3.2.2 Data Set Generator (DSG) Usage

   a) Double click the Data Set Generator exe file and the interface as shown in figure 3.20 will appear.

   b) In the new table name field specify a name for the synthetic database table to be generated. Replication of Table names is not allowed.

   c) The column name and data type fields shown in figure 3.20 are set as default and user can change or redefine these field values as per his/ her requirements.

   d) Click next to view the next form.

Figure 3.20 Liberty to specify desired column names and data types in the synthetic database table.
Figure 3.21 Selection of the Data Source Donor

- **e)** Select the donor source database followed by the selection of database table whose attributes we wish to exploit for the purpose of generating synthetic test data (Refer figure 3.21).

- **f)** Click next to view the next form or exit to relocate the donor database.

- **g)** On the left pane the fields defined for synthetic test database are shown where as on the right side the fields available in the selected source table of donor database will become visible as shown in figure 3.22.

- **h)** Now map the index values of resembling domain sets from the donor with the fields declared for test database, like address in left pane with index value 2 can be puffed up with address values from right pane by inputting 2 in the text box before address field as shown in figure 3.23.
Figure 3.22 DSG showing mapping options before performing table merge operation.

Figure 3.23 Mapping of resembling domain sets and declaration of number of synthetic records needed

i) Here the DSG too provides the option to remove duplicate records, define valid range of input values and to append, remove or edit any value we are going to use for generating synthetic records (refer figure 3.24 and 3.25).
Figure 3.24 The DSG seeking directions before generating synthetic records.

Figure 3.25 Field values extracted from donor database are presented for user review.
j) When satisfied with the quality of donated records from resembling domain sets click on merge table 2 to table 1 button as shown in figure 3.22.

k) One can add another donor table for exploiting resembling domain sets if required by following the aforesaid procedure.

l) Specify the number of synthetic records to be generated.

m) Select the key columns of your choice for performing crossover multiplication.

n) Click Generate data button to generate synthetic test database. Figure 3.26 presents the synthetic records generated by the DSG.

<table>
<thead>
<tr>
<th>Generated Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Nita</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>Maria</td>
</tr>
</tbody>
</table>

Figure 3.26 Sample of generated synthetic records

Testers need to test the cumulative performance of the system using predefined set of test data values. The process of generating such data values is known as testbed preparation. For successful testbed origination, there is a need to
follow a systematic approach for generating test data. Database applications are indispensable for every business organization, still business houses need to realize the importance of testing their database applications. The database applications are extremely complex from testing prospective as they are composed of many components stacked in layers which are subject to a constant change. Database oriented applications are designed to execute concurrent queries from different clients; hence a single fault in database can result in unrecoverable data loss. Testing an ETL routine determines the validity of the data warehouse solution to a business problem. As ETL routines are primarily responsible for assuring quality of data in a data warehouse hence defining only an appropriate testbed is not sufficient. Generation of effective test cases according to the database schema and the execution of the same are also necessary for quality data warehouse solutions.