Chapter 4

ETL development and Quality Assurance

4.0 INTRODUCTION

In simplest words a data warehouse can be defined as collection of key pieces of information used to manage and direct the business for the most profitable outcome. The data warehouses are strategic investments which may require business process redesign in order to generate the projected benefits. Keeping in view the massive infrastructural requirements many organizations address the data warehouse implementation issue by sponsoring an initial prototyping activity to further understand the feasibility and benefits of data warehouse.

The Corporations, government agencies and not-for-profit groups are all flooded with enormous amounts of data. Combined data from numerous sources has a great potential to increase the usefulness of information. In the prevailing commercial environment the accuracy of business decisions is dependent on the quality of information procured by the company. The business reporting tools are responsible to provide condensed and formatted information useful in understanding the market scenarios quickly and efficiently. This understanding may lead to improved decision making only if accurate information is available to management. This however is not always true as the information accuracy is directly proportional to the quality of data processed to generate information. If the data to be processed is error prone then the decisions that are reached from it will be imperfect because of the imprecise result sets. The contemporary marketplace is virtually dependent on databases. The more the databases will grow the difficult it will be to trace errors. This scenario has given birth to the concept of Information architecture, which has further resulted in the progeny of Operational Data Stores (ODS), Data Warehousing and Data Marts etc. The
ETL development and Quality Assurance

aforesaid data management projects were started with the anticipation of data quality enhancement through extraction, transformation and loading routines. In the absence of predefined quality checks, generally the ETL routines are unable to refine incoming data streams. Data quality can either be assured manually or with the help of automated logic. To automate the quality assurance procedure, customized data purification routines need to be placed amid source and target systems. As these routines are programmed to run in juxtaposition with ETL routines, hence they distil the incoming data before synthesizing it in the target system. The preliminary automatic data purification tools relied on the paradigm of creating streams of programmatic logic to check and fix the data with the help of a graphical user Interface running at the front end application. Although this technique was quite effective but it led to complex insurmountable applications that required large amounts of resources, hence couldn't be implemented as an enterprise wide solution. Present day data purification tools combine business rules, validation techniques and pre defined procedures for data scrubbing during data movement. This methodology is considered as the most efficient and cost effective way for data quality management.

Data warehouse projects include operational processes which are designed for labour intensive workflows and for managing back stage collection, extraction, cleaning, transformation and transport of raw data. Collectively these operational processes are termed as Extraction, Transformation and Loading (ETL) tools.

4.1 THE ETL SYSTEM

Early data warehouses ETL systems were not proficient of managing the extensive processing required to perform the complex transformations involved in the warehouse load process. So third-party tools like IBM's WebSphere, DataStage and
Informatica were used to organize data movement between source systems and the data warehouse. Now days with the advancement in hardware and data warehouse development technology, the designers consider Extract Load and Transform (ELT) as an alternative to ETL. ETL logic states that before loading the data to the data warehouse, the data should be moved to an intermediate platform where the transformation rules should be applied. On the other hand the ELT follows a standard data transfer mechanism such as File Transfer Protocol (FTP) to transfer the bulk data directly to the data warehouse. The transformation rules are then applied to the data warehouse tables with the help of preloaded procedures instead of any intermediate staging area. A comparative analysis of the two options is summarised in table 4.1:

<table>
<thead>
<tr>
<th></th>
<th>ETL</th>
<th>ELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A dedicated external system is applied to take care of transformation logic for data standardization and business rules thus reducing the unnecessary burden from the data warehouse.</td>
<td>There is no dedicated external system responsible to tackle transformation logic and business rules. The transformations are done on the data already loaded into the data warehouse.</td>
</tr>
<tr>
<td>2</td>
<td>The whole data has to travel first from source to staging area and then from staging area to the data warehouse through the network thus causing excess network traffic when there is no dedicated link between the ETL server and the data warehouse.</td>
<td>The files are loaded from the source systems to the data warehouse via FTP or other secure file transfer methods, hence the network traffic is least affected.</td>
</tr>
<tr>
<td>3</td>
<td>The ETL server requires high performance CPU and huge disk capacity to sustain the transformation process. This can lead to the need for expensive and highly sophisticated hardware.</td>
<td>Transformation logic to be applied on the stored data of a data warehouse will utilize additional data warehouse resources for execution.</td>
</tr>
</tbody>
</table>
ETL Development and Quality Assurance

<table>
<thead>
<tr>
<th></th>
<th>ETL</th>
<th>ELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ETL tools have the capability to interact with other external engines for data validation before the data is loaded on the data warehouse. Such as Geographic Information Systems (GIS).</td>
<td>Complex transformations which may require external sources of data are not easy to implement with the stored procedures of the data warehouse.</td>
</tr>
<tr>
<td>5</td>
<td>Errors if any that occur during the transformation process can be located and corrected before data is loaded in the data warehouse table thus reducing the need for time consuming database rollbacks.</td>
<td>Database roll-backs are inevitable in case an error occurs during the transformation process. Generally these rollbacks are taken on temporary tables.</td>
</tr>
<tr>
<td>6</td>
<td>Depending on the number of sources feeding the data warehouse, the ETL licensing may become a costly affair.</td>
<td>The cost for loading the data warehouse through ELT is quite lower than the ETL architecture as there is no additional software licensing required.</td>
</tr>
<tr>
<td>7</td>
<td>Time spent on bringing in the data to the data warehouse is higher.</td>
<td>Time for getting data to the data warehouse is reduced as there is no staging process required.</td>
</tr>
</tbody>
</table>

Table 4.1 Comparative Analysis of ETL and ELT

The fundamental nature of data warehouse solutions is “change and evolution”. Employing traditional ETL solutions may limit the ability to embrace this change but the unavailability of tools and techniques to implement ELT effectively, ETL is the current choice of data warehouse industry. Before selecting the loading procedure one must weigh up the data transformation requirements along with the desired data quality of the targeted database. If the transformation rules are intricate and cannot be carried out using stored procedures of the database than ELT architecture should be avoided.
ELT is best suited for environments where text parsing routines are required to implement data standardization and cleansing. For hefty environments where numerous sources with terabytes of transactional data are involved, ETL is the best suited architecture [4]. On the other hand ELT is best suited for loading small data sets where relatively simple transformation logic is applied. ELT is also best suited for manipulating limited business data of a departmental data mart. A data mart has a logical structure similar to that of the data warehouse.

The architectural difference between these two approaches is shown in figure 4.1 and 4.2 respectively.
4.2 BUILD vs. BUY DECISION

As business organizations investigate their data unification needs, the first decision they need to make is whether to build or buy an ETL tool. Although there are numerous ETL solutions offered by big brand names like Informatica, IBM and Microsoft but according to a research conducted by The Data Warehousing Institute, customized ETL solutions are still the first choice among business houses. ETL solutions from various brands are very much proficient in their functionality but still most of the organizations prefer to hand code their ETL programs rather than using off-the-shelf ETL software. These organizations/companies advocate their decision by aiming at the high cost of many ETL tools and the profusion of programmers on their staff. It is extremely difficult to state which the best ETL solution is, but to facilitate this build or buy decision a few aspects have been discussed in table 4.2.
<table>
<thead>
<tr>
<th>BUILD</th>
<th>BUY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 It is cheaper and quicker to code ETL programs rather than using a vendor ETL tool.</td>
<td>ETL tools are expensive to purchase and requires renewal of license.</td>
</tr>
<tr>
<td>2 It is cheaper to maintain as it is geared with a specific business.</td>
<td>They are developed for generalized use.</td>
</tr>
<tr>
<td>3 Code written is based upon custom specifications and meta data model.</td>
<td>Industry specifications are considered instead of custom specifications.</td>
</tr>
<tr>
<td>4 No need to pay unnecessary training or maintenance fee to any vendor.</td>
<td>One has to bear paid training sessions and heavy maintenance costs to introduce vendor ETL tools.</td>
</tr>
<tr>
<td>5 Easily available object oriented technology is best suited for ETL development.</td>
<td>Licensed vendor ETL tools are not desirable for small businesses.</td>
</tr>
<tr>
<td>6 Challenging factors like migrating source data into a data warehouse along with data cleansing jobs can easily be performed with the hand coded ETL routine.</td>
<td>Vendor ETL tools follow a generalized approach to migrate source data into a data warehouse. They are quiet about how to identify and clean dirty data or to build interfaces to legacy systems.</td>
</tr>
<tr>
<td>7 Metadata is rarely maintained</td>
<td>Metadata is highly maintained.</td>
</tr>
<tr>
<td>8 Complex mappings can be handled easily with custom built ETL code.</td>
<td>Only predefined mapping procedures can be carried out which generally elude complex mappings.</td>
</tr>
</tbody>
</table>
It is difficult to ensure adequate stability, reliability and performance. Adequate stability, reliability and performance are well ensured in advance.

Team of expert programmers is required to develop a customized ETL tool. Highly salaried and experienced programmers are hired by ETL vendors for developing, training and maintenance purposes.

To keep the costs down and for better turnaround times one can outsource the ETL development code to Asian countries. Generally they are developed by highly skilled, salaried and in house trained coders; outsourcing is avoided to avert any kind of data breach.

These tools are flexible and can be adjusted in accordance with the changing business dimensions. These are not much flexible and one has to look in for readymade plug ins to cope with changing business dimensions.

Creating and integrating user defined functions is a cumbersome and difficult job. The well designed proficient modules are integrated in advance.

Rigorous testing and debugging effort is required. No need to perform testing and debugging at the user’s end, only maintenance is required.

Source data is well understood. They follow a generalized approach in understanding source databases.

<table>
<thead>
<tr>
<th>BUILD</th>
<th>BUY</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>It is difficult to ensure adequate stability, reliability and performance.</td>
</tr>
<tr>
<td>10</td>
<td>Team of expert programmers is required to develop a customized ETL tool.</td>
</tr>
<tr>
<td>11</td>
<td>To keep the costs down and for better turnaround times one can outsource the ETL development code to Asian countries.</td>
</tr>
<tr>
<td>12</td>
<td>These tools are flexible and can be adjusted in accordance with the changing business dimensions.</td>
</tr>
<tr>
<td>13</td>
<td>Creating and integrating user defined functions is a cumbersome and difficult job.</td>
</tr>
<tr>
<td>14</td>
<td>Rigorous testing and debugging effort is required.</td>
</tr>
<tr>
<td>15</td>
<td>Source data is well understood.</td>
</tr>
</tbody>
</table>

Table 4.2 Build vs. Buy Analysis
Following the facts presented by TDWI (The Data warehousing Institute) research 2007 this study too opines that the easiest way to migrate data is with the help of in-house developed solutions with hand-coded algorithms for Extraction, Transformation and Loading (ETL). This low level implementation makes the maintenance of such migration solutions a complicated job and can be a cause for hidden errors at logical or technical levels. There is no right or wrong decision about whether to build or buy ETL solutions. But once it has been decided to build an ETL project there begins a new chore of assuring its quality. Any hand coded ETL module can not be used on real life data management until and unless its performance has been assured by testing it rigorously. To identify prime testing zones for an ETL tool it was decided to hand code an ETL tool which should be capable of extracting, transforming and loading data from databases like Oracle, SQL, MS Access, MS Excel and flat files like those of MS word.

4.3 THE ETL DEVELOPMENT

The Extract-Transform-Load (ETL) system is the foundation of any data warehouse project. A well planned ETL system extracts data from the source systems, enforces data quality and consistency standards and finally delivers data in a presentation-ready format so that application developers can build applications and end users can make decisions. Although building the ETL system is a back room activity which is not very visible to end users, but the data warehouse literature presumes that it easily consumes 70 percent of the resources needed for implementation and maintenance of a typical data warehouse system.
4.3.1 ETL Constraints

The data warehouse world is very sensitive about the everyday batch cycles responsible for loading data. During these cycles the data warehouse administrator has to cope up with a number of issues like efficiency, managing dependencies, job completion deadlines and the global time constraints [49]. Keeping in view the aforesaid issues the most typical problem is to schedule the whole ETL process. The administrator has to schedule the right execution order for dependent jobs and job sets using existing hardware for the permitted time schedule.

4.3.2 Designing the Extract Module

The prevailing practice to implement any extract module is to duplicate the entire source data to the data warehouse. The business organizations prefer an extract module which is capable of sorting, cleansing and rearranging the selected source data before loading the same into business warehouse. The data sources are generally operational systems and they need to go offline while data extraction takes place. Business organizations can’t afford to suspend their operational systems for a long time hence the situation demands a balance in efficient ETL processing and to get the necessary source data within the allowed time slot. Prime requisite for a successful source extract is accurate field mapping. Mapping, selecting and merging data from the specified source databases is a challenging job because of different data definitions and high data redundancy in operational data stores. On the fly data scrubbing and loading may need extra efforts for time consuming table lookups and cross referencing of specific keys, instances through extremely complex data extraction routines. This on the fly data-cleansing process is expected to slow down the extract process, which in turn would make the operational systems busy longer
than is acceptable. These constraints of business environment forced the data warehouse industry to populate the BI databases in three different steps namely extract, transform and load. This concept introduced a staging area in between the source and target databases [70]. Now the source extract is responsible to fetch the selected fields on to the staging area within the allocated time slot. The data cleansing and transformation takes place simultaneously at the staging area before loading the consolidated data onto the business intelligence database. To develop an effective ETL routine one has to understand the prevailing business rules for applying transformations on the source data. The design of transformation logic initiates with activities like project planning, requirement definition and data analysis till application prototyping starts and is followed by Meta data analysis. A simple data extract module is shown in figure 4.3.

![Figure 4.3 A Simple Data Extraction Module](image-url)
4.3.3 Designing the Transformation Logic

There is a myth in data warehouse industry that almost eighty percent of ETL processing effort is associated with data transformation routines. The extract and load routines are expected to contribute only twenty percent of the ETL processing effort. Involvement of heterogeneous sources with independent schema definitions makes it difficult to consolidate targeted business intelligence database. Decision needs to be taken in advance if the backward references from the target database to the source databases have to be kept or not? Managing inconsistent, inaccurate and duplicate data values is a crucial concern for transformation routines. The varying data definitions and data formats in the source files demands specialized routines for consolidation and cleansing of data. Data cleansing is an ongoing process with every data load cycle. Further the transformation routines may transform the naming standards of source data according to the data warehouse business standards. Some data elements have to be split across different columns and on the other hand some data elements from different operational systems may merge into a single column in the data warehouse target database. Working of transformation logic is shown in Figure 4.4.

Figure 4.4 Data Transformation Logic
4.3.4 Designing the Data Load Module

The final phase in ETL processing is the loading phase. Its prime responsibility is to load the scrubbed and transformed records into the target data warehouse. Loading can be done in two ways, first way is the row by row insertion of data and the second option includes the bulk load methodology. The organizations make a choice among the two according to the prevailing time and resource constraints.

4.4 ETL SUB SYSTEMS

A hand coded ETL routine to extract and unify data is the most popular option among small and mid-sized enterprises. The ETL is a big term having many small independent sub systems of its own like:

a) **Aggregate building System**: It is for creating and maintaining of physical database structures.

b) **Backup system**: It is responsible for backing up data and metadata.

c) **Cleansing system**: It is a dictionary driven system for parsing of names and addresses of individuals and organizations etc.

d) **Data Change identification system**: It keeps an eye over Source log file readers, source date and sequence number filters etc.

e) **Error tracker and handler**: It is a widespread system for identifying and retorting to all ETL error events.
f) **Fact table loader:** It is equipped with push/pull routines for appending/updating transaction fact tables.

g) **Job schedule Handler:** It is for scheduling and launching all ETL jobs.

h) **Late arriving fact and Dimension Handler:** It is for insertion of fact and dimension records that have been delayed in arriving at the data warehouse.

i) **Metadata manager:** It is for assembling, capturing and maintaining all ETL metadata and transformation logic.

j) **Pipelining system:** It is required for implementing streaming data flows.

k) **Quality Checker:** It is responsible to check the quality of incoming data flows.

l) **Recovery and restart system:** It is responsible for restarting a job that has halted.

m) **Security system:** It is responsible for the security of data within an ETL.

n) **Source Extract system:** It includes Source data adapters along with push/pull routines for filtering and sorting at the source.

o) **Surrogate key Management System:** It is a Pipelined, multithreaded process for replacing natural keys of incoming data with data warehouse surrogate keys.

All the aforesaid ETL subsystems are categorized and bundled together in a layered fashion to provide a well defined set of services. Keeping in view these categories the anatomy of an ETL routine is shown with the help of figure 4.5.
4.4.1 Role of ODBC (Open Database Connectivity)

Open Database Connectivity (ODBC) is responsible for enabling users to access databases from their Windows based applications. The original intention for ODBC was to make applications portable, i.e. if an application’s underlying database is changed say from MS Access to Oracle then the application layer did not need to be
recoded and compiled to accommodate the change. Instead, simply a change in the ODBC driver is required, which is transparent to the application. ODBC drivers for practically every DBMS in existence on virtually any platform are readily available. One can also use ODBC to access flat files.

The drawback to ODBC’s flexibility is that it comes at a performance cost. ODBC adds several layers of processing and passing of data to the data-manipulation process. For the ETL process to utilize data via ODBC, two layers are added between the ETL system and the underlying database. The first layer is called ODBC manager and it accepts SQL from the ETL application and routes it to the appropriate ODBC driver. It also maintains the connection between the application and the ODBC driver which is the second layer imposed by ODBC framework. The ODBC driver is the real workhorse in the ODBC environment. The ODBC driver is responsible for translating simple ODBC based SQL queries according to underlying database.

As per the scope of this research a windows based ETL prototype is required to mimic and promulgate ETL functionality and quality assurance procedure, hence ODBC was considered to be the sole platform for extracting data from multiple sources. Here flexibility was preferred over performance because of the limited functionality of the prototyped ETL environment.

4.4.2 ETL Prototype Workflow

Keeping in view the objectives of this research, the ETL anatomy and the facilities offered by the ODBC platform were analyzed to develop an ETL prototype. This prototype was named Data Extractor and was expected to virtually mimic the actual ETL routine in a simulated data warehouse environment. Table 4.3 presents the workflow of the prototyped ETL routine.
<table>
<thead>
<tr>
<th>ACTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start</td>
</tr>
</tbody>
</table>
| 1      | 1.0 Select Target Database Type  
|        | 1.1 Select Target Table           |
| 2      | 2.0 Select Source Database Type   
|        | 2.1 Select Source Table           
|        | 2.2 Validate Source Availability and Solve Connectivity Issues |
| 3      | 3.0 Copy Source Data Schema to Staging Area |
|        | 3.1 Copy Target Data Schema to Staging Area |
|        | 3.2 Map Source and Target Schema Fields |
| 4      | 4.0 On Merge Consult Metadata Repository  
|        | 4.1 Verify and Validate Data Extraction Rules  
|        | 4.2 Extract the Specified Source Data onto the Staging Area |
| 5      | 5.0 Begin Data Validation, Scrubbing and Transformation Procedure  
|        | 5.1 Maintain Log File for Rejected Records |
| 6      | 6.0 Load the Purified Records to the Target Database  
|        | 6.1 Commit the Target Data Alterations |
| Stop   |             |

Table 4.3 Workflow of ETL Prototype

4.4.3 The ETL Prototype

The Data Extractor is a tool developed for synthesizing data from multiple logical resources into the specified target database sink. This primary ETL structure can extract records from numerous different resources; can scrub incoming data streams to the extent possible to ultimately load the cleansed records into a target database. Like any hand coded ETL routine The Data Extractor functionality is limited to managing personal records data.

4.4.3.1 System Requirements: following are the hardware and software requirements of the ETL prototype:
Software Requirement

<table>
<thead>
<tr>
<th>SOFTWARE</th>
<th>VERSION</th>
<th>SOURCE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>98 or higher</td>
<td>Microsoft Corporation</td>
<td></td>
</tr>
<tr>
<td>SQL ODBC driver</td>
<td>2000.81.9041.40</td>
<td>Part of OS on W2K, XP, Vista.</td>
<td>Only to import data from/into MS SQL Server</td>
</tr>
<tr>
<td>MS Access ODBC driver</td>
<td>4.00.6364.00 or higher</td>
<td>Part of OS on W2K, XP, Vista.</td>
<td>Only to load data from/into MS Access Database</td>
</tr>
<tr>
<td>MS Excel Driver</td>
<td>4.00.6364.00 or higher</td>
<td>Part of OS on W2K, XP, Vista.</td>
<td>Only to load data from/into MS Excel Sheet</td>
</tr>
<tr>
<td>MS Text Driver</td>
<td>4.00.6364.00 or higher</td>
<td>Part of OS on W2K, XP, Vista.</td>
<td>Only to load data from/into Text Files</td>
</tr>
<tr>
<td>Oracle Client</td>
<td>7.3.4 or higher</td>
<td>Provided by Oracle</td>
<td>Only to import data from/into Oracle</td>
</tr>
</tbody>
</table>

Table 4.4 Software requirements for the ETL prototype

Hardware Requirement

Processor Intel Celeron 2 GHz or above

Ram 64 MB or above 256 Recommended

Hard disk 20MB of free space

Screen Resolution 1024*768 Recommended

4.4.3.2. Key Features of the ETL Prototype:

a) Extraction:

- Multiple Delimited or Fixed width Text files
- Multiple Excel files
b) **Validation:**

- Validation Functions
- String Validation
- Number Validation
- Date Validation
- Time Validation
- Validation Against list of values
- Regular Expressions: Post codes, Phone Numbers etc

c) **Transformation:**

- String Transformation
- Number Transformation
- Date Transformation
- De-duplication
- Translating coded values (e.g., if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female)
- Managing null values
- Joining together data from multiple fields
- Generating surrogate key values
- Customized Transformations
4.4.3.3. ETL Prototype Usage

a) Start The Data Extractor Application and the interface shown in figure 4.6, will appear.

b) In source database frame select source database type.

c) From the populated table list select the source table of choice.

![Image of the Data Extractor Application interface]

**Figure 4.6 Selection of Data Source and Source Data type.**

d) In the Target database frame select the target database type (default database type is set to MS-Access).

e) Select the table from the selected database in which you wish to append the extracted information.

f) After successful connections between source and target databases convert button will become visible (refer figure 4.7).
ETL Development and Quality Assurance

Figure 4.7 Successful Connections between Source and Target Databases.

- **g)** Click convert button to view the next form.
- **h)** Replicated source schema can be viewed under the source table tab as shown in figure 4.8.

Figure 4.8 Replicated Schema of Source Database to be extracted.
i) Replicated target schema can be viewed under the target table tab as shown in figure 4.9.

![Figure 4.9 Replicated Schema of Target Database.](image1)

j) Map the fields of source as well as of target schema through the interface shown in figure 4.10.

![Figure 4.10 Mapping of Source and Target Database Fields.](image2)
k) After mapping the source data will be extracted to the staging area where data, already extracted from various sources according to target database schema can also be seen (refer figure 4.12 and 4.13 respectively).

l) On clicking the merge button the application will enforce all the predefined transformations and validate each record at staging area before loading the same into the data warehouse (refer figure 4.11).

m) Rejected records will be shown in the results tab as shown in figure 4.14.
### Figure 4.12 Source Data at Staging Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 4.13 Data from different sources at Staging Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Address</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ETL Development and Quality Assurance**
4.5 ANOMALIES FOUND IN SYNTHESIZED DATA

As a hand coded ETL routine is fanatical to a specific entity domain, this routine was customized to handle personal records from different sources. The synthetic data generated with the help of data set generator was placed on different logical locations and synthesized using this primary ETL structure. Following are the anomalies which were found in the data, synthesized from different locations.

4.5.1 Lexical Anomalies: Syntactical errors are generally lexical errors which would name discrepancies between the structure of the data items and the specified format. In a relational data base the data is stored in the table form with each row representing a tuple and each column representing an attribute. If there are five attributes of a relation or table then each tuple will also have five attributes. But if some or all of the rows contain only four columns then
the actual structure of the data will not confirm to the specified format. This will result lexical errors. For example record no. 125 is lexically erroneous because the name column is representing a numeric value and the roll no. Column is representing a character string. Identified anomaly is shown in figure 4.15.

![Figure 4.15 Lexical Anomalies](image)

4.5.2 **Format Errors:** These errors correspond to the value of a given attribute does not conform to the anticipated domain. For example record no. 56 is representing a correct name but in a wrong way. As the last name is specified first and the first name is specified as the last name hence it is violating the domain format. Format error is shown figure 4.16.

![Figure 4.16 Format Errors](image)

4.5.2 **Irregularities:** Irregularities are apprehensive with non consistent use of values, units and abbreviations etc. For example in the target database where data has been merged from different sources the sex attribute has been represented as M/F and Male/ Female. Such anomalies will create a bottleneck for developing a generalized formula or query for data processing. Irregularities are shown in figure 4.17.
4.5.4 Integrity Constraint Violations: Integrity constraint violation illustrates that set of tuples, which do not satisfy the imposed integrity constraints. Integrity constraints are essential to empathize the mini world by confining the set of valid instances. A constraint can be defined as a rule for the representation of the knowledge about the domain along with the set of valid values for representing the concerned facts. For example the Dates 30th and 31st February are not possible and hence are violating the integrity constraints. Such values become apparent especially when data is imported from text/flat files where there is no provision to declare any integrity constraint. In the same way age cannot be zero if there is an entry for the Date of Birth column. Such anomaly is shown in figure 4.18.

4.5.5 Contradictions: Contradictions refers to the values within a tuple of a relational database that contravene some kind of dependency between values. For example the formula for calculating age is (Current date (Date on which transaction was recorded) - Date of Birth). If this constraint is not followed correctly it may result in contradictions. For example in the figure given below the second record has date of birth value equal to 05 Apr 86 and the age is 24. A contradictory record is shown in figure 4.19.
4.5.6 **Duplicates:** Duplicates can be stated as tuples representing the same world entity from the concerned mini-world. The values of these tuples do not need to be completely similar. These records may also result in contradictions as they represent the same entity but with different values for all or some of its properties. Like in figure 4.20 the record no. 8 and 9 are representing the same entity from the mini-world because the name and registration no. fields are same in both cases but information provided here is bewildering. Duplicate records may also lead to contradictions. An example of such duplicate record is shown in figure 4.20.

**Figure 4.20 Duplicate records with different serial number**

4.5.7 **Data Type Mismatch:** The data type mismatch can impose a serious problem while merging data from various sources. For example, it is possible to store an integer value in text data type but it is not possible to store a text value in an integer data type without conversion. Such conversions may or may not provide desired results.

4.5.8 **Missing Values:** These values are the result of omissions during data collection. The major reason for missing values is constraint violations as if we have null values for attributes where there exists a non null constraint for
them. If missing values are present this means there is no such constraint imposed during data extraction. These values impose a major problem when one wants to substitute values for such records for example one cannot insert null values into an integer field because by default its value is set to zero. This insinuates that if we try to consolidate data by filling dummy values in null fields then numeric data types may require special attention because their default value is set to zero. Figure 4.21 shows missing values.

Figure 4.21 Missing Values

4.5.9 Typographical Errors: Typographical errors are those errors, which grounds due to typing mistakes. These errors are almost impossible to identify with automatic checks. For example due to typing mistake male can be written as amle. The possible solution is to rearrange the data in ascending or descending order and analyze it manually afterwards.

4.5.10 Transformation & the preservation of meaning: The transformation challenge is expressed in terms of successfully matching data fields. But this is not the best way to think about it. The real challenge is preserving the meaning held in records throughout the transformation process. To eliminate data inconsistency and integrity problems (as stated above) if one tries to purify data through data transformations it is again not an easy job.
4.5.11 Normalization Break-up: Illogically or inconsistently stored data can cause a number of problems. A poorly designed database may provide erroneous information, may be difficult to use, or may even fail to work properly. Most of these problems are the result of two bad design features called: redundant data and anomalies. Redundant data is unnecessary reoccurring. Anomalies can be defined as any occurrence that weakens the integrity of your data due to irregular or inconsistent storage for example delete, insert and update irregularity that generates the inconsistent data. Basically, normalization is the process of efficiently organizing data in a database to reduce redundancy. Hence the two main objectives of the normalization process can be stated as to eliminate redundant data, which means storing the same data in more than one table, and to ensure that the data dependencies should make sense. Both of these are valuable goals as they reduce the amount of space a database consumes and ensure that data is logically stored. Generally it is considered as impossible to normalize a database but it can be done with the help of a smart ETL routine, which should be competent enough to normalize the target database before populating data into it.

4.6 TESTING GOALS AND TEST CASES

As ETL routines are accountable for synthesizing quality data for decision making hence assuring the quality of custom built ETL is vital [90]. Keeping in view the importance of data quality in decision making table 4.5 presents some general goals for testing an ETL application:
ETL Development and Quality Assurance

<table>
<thead>
<tr>
<th>S NO.</th>
<th>TESTING GOAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completeness</td>
<td>This test ensures that all expected data is loaded completely.</td>
</tr>
<tr>
<td>2</td>
<td>Transformation</td>
<td>It ensures that data transformations have followed business rules correctly.</td>
</tr>
<tr>
<td>3</td>
<td>Data quality</td>
<td>It is to ensure that the ETL application is aware of data quality definition and can reject, corrects or ignores and reports invalid data.</td>
</tr>
<tr>
<td>4</td>
<td>Performance</td>
<td>It is required to verify that data loads and queries perform within expected time frames and that the technical architecture is also scalable.</td>
</tr>
<tr>
<td>5</td>
<td>Integration testing</td>
<td>It ensures that the ETL process is cooperating well with other data warehouse sub systems.</td>
</tr>
<tr>
<td>6</td>
<td>User-acceptance testing</td>
<td>It is to verify that the data warehouse ETL solution meets current user requirements and anticipates their future expectations.</td>
</tr>
<tr>
<td>7</td>
<td>Regression testing</td>
<td>It guarantees that the system performance remains intact each time a new release of code is completed.</td>
</tr>
</tbody>
</table>

Table 4.5 Testing Goals for an ETL Application

The aforesaid ETL testing goals provide vital information for developing prime test cases for a hand coded ETL routine. One can consider the following aspects of these testing goals while developing the test cases:

4.6.1 Data Completeness

The basic test of data completeness is to ensure that all expected data is loaded into the data warehouse. This test validates that all records, all fields and the full
contents of each field are loaded. The following test cases are possible in this category:

a) Boundary value analysis is required to find out database limitations if any.

b) Each data field should be tested for truncation of data values.

c) The count of source data records should be equal to the count of data records loaded to the warehouse plus rejected records.

d) The range and value distributions of the fields in a data set can be verified using a data profiling tool.

e) The unique values of key fields in source data should be preserved in the data warehouse.

4.6.2 Data Transformation

Testing the transformations is required for validating that data is transformed correctly according to the prevailing business rules. Testing the transformation logic of an ETL application is considered as the most difficult task of testing. One of the easiest technique is to select some sample records and "stare and compare" them to validate data transformations. This technique can be useful but it involves step wise manual testing and experienced testers who can understand the ETL logic. An amalgamation of automated data profiling tools and automated data movement validations can be a better methodology for ensuring ETL quality. The following data movement are important from ETL prospective:

a) Correctness of ETL generated fields like surrogate keys have to be ensured.
b) If possible range and distribution of values in each field between source and target data should be compared.

c) One has to make sure that the data types in the data warehouse are same as declared in the design document.

d) Referential integrity should be maintained.

e) Synthetic test data should mutate the expected inputs of the ETL routine to ensure flexibility as the data definitions and validations may change according to changing business rules.

f) The type of input data required and expected results according to business rules should be recorded into a testing document. This technique can also be used to design the ETL routine.

g) There should be criteria to process incomplete and obsolete records.

4.6.3 Data Quality

The data quality management mechanism of an ETL routine is responsible for defining the data quality. This system manages data by selective substitution, rejection and correction of incoming data while preserving the actual information. The data quality defined during design process can be ensured in following manner:

a) A well defined mechanism is required to identify and eliminate duplicate records.

b) Data quality rules are usually not visible to the users hence it is important to decide how to process invalid data. Rejected invalid records can be stored in a log file which can further be used to publish data quality reports. These data
ETL Development and Quality Assurance

quality reports are vital in identifying logical and systematic issues with data sources.

c) Rectify and validate correct value if possible like city name can be rectified based upon the postal pin code.

d) Rejection of records is necessary if incoming data is not compliant with the data definitions of the data warehouse.

e) Validation of correct values can be made like age can be calculated from date of birth.

f) Where to substitute null in case there is no value available?

4.6.4 Performance

With the expansion of data warehouse the ETL load times may increase and the response time of queries is also expected to grow because of rising complexity and huge volumes of data. However these issues may deteriorate the performance of ETL routine but strong technical architecture and good ETL design can pose a remedy for such issues. ETL performance can be evaluated on the following grounds:

a) Comparison of ETL processing times for extracting smaller data sets and larger data sets is necessary to anticipate scalability issues.

b) ETL should be tested for maximum expected data loads.

c) Simple and complex queries should be run on large database volumes to validate query performance.

d) Timing of reject process is crucial for analyzing huge volume of incoming data.
4.6.5 Integration Testing

Integration testing ensures that the ETL routine is cooperating well with other data warehouse sub systems. Here instead of testing the ETL application alone one has to identify and test the possible interactions between various subsystems. Process failure handling mechanisms and data recovery techniques at staging area also need to be tested during integration testing.

4.6.6 User-Acceptance and Regression Testing

To ensure user acceptance ETL routine should be tested with almost real looking data. The ETL routine should behave as per the expectations of the user. ETL interface should be user friendly as such changes at later stage may prove expensive. The end users should be involved in interface design right from the beginning to develop an easy, acceptable and familiar user interface. Regression testing is required to validate the existing functionality with every software code developed to rectify a defect or to enhance the performance of the ETL routine [80]. The simplest possible technique for regression testing is to compare the results of previous successful results with the results of the newly developed code using the same source data sets. It is much quicker to analyze results only than to run an entire data validation procedure.

A successful test case is one which makes the system halt at the occurrence of any faulty event. The primary ETL structure was later analyzed for possible vulnerabilities. Accordingly test cases of prime importance were developed and deployed to plug in the loop holes found in primary ETL routine. The test cases developed for assuring the quality of hand coded ETL prototype are shown in table 4.6. However it was also realized that to substantiate the need of automated testing, ETL performance needs to be analyzed statistically.
Table 4.6 ETL Test Cases
The aforesaid test cases were considered to be of prime importance for assuring the data quality hence all of these tests were automated and included in the ETL routine developed. It has been observed that if quality checks are imposed during the data extraction stage than the effort required to refine and transform data can be reduced considerably resulting in the saving of time and money.

**Figure 4.22 The Loading Logic of ETL Prototype**

The primary reason for the convolution of the data extraction and transformation functions are the diversity of the source systems. This diversity includes bewildering combination of computing platforms, operating systems, database management systems, network protocols, and legacy source systems etc. Hence there is a need to pay special attention to the various sources and to begin with
one should generate a complete record of the source systems. With this record as a starting point one should work out all the details of data extraction. The difficulties encountered in the data transformation function should also be related to the heterogeneity of the source systems. The loading procedure might seem to be the simplest one but it is solely responsible for consolidation and integration of targeted database hence its performance evaluation is also crucial [42]. Refined data from the staging area is to be loaded into the data warehouse according to the logic shown in figure 4.22 through the ETL prototype. The ETL data loading mechanism is responsible to generate surrogate keys for the data warehouse usage before loading the purified data into data warehouse tables. The surrogate key is a combination of four field values. The first field corresponds to a system generated primary key value. The second field is the identification number of the source to whom incoming data actually belongs to. Third field stores the primary key given to a specific record at its source database while the fourth field manages the time stamps for incoming data streams. As can be seen in the example shown in figure 4.22 amalgamation of all these four fields may result in a foolproof data management strategy. While comparing source primary key field only one can identify the duplicates within the target database. Comparison of timestamp fields may help in the identification of records with slowly changing dimensions. The ETL performance was recorded before and after the imposition of automated testing. To verify the effectiveness of automated testing the recorded results needed to be analyzed statistically. Chapter 5 concludes the statistical analysis of the ETL routine performance.