2014 International Conference on Advances in ICT for Emerging Regions (ICTer): 134 - 141

Versatile Inter System Information Exchange Facilitator [VISIEF]

Muditha Rangani Tissera

Information Systems Engineering Department, Faculty of Computing, Sri Lanka Institute of Information Technology
New Kandy Road, Malabe, Sri Lanka
muditha.t@sliit.lk

Abstract— **No longer application systems operate as standalone applications. They are interconnected with various other businesses applications. This helps data interoperability among systems. Connectivity and data transmission may function in real time or batch/offline basis.**

Offline data transmission is very common in today's business applications. In addition to day-to-day business operations, this is widely used in data migrations, data warehouse data populations and many more.

Offline data transmission is performed using flat files (.csv, .txt etc.) and these are in different formats for different business functions. Hence, transformation process (processing the data files) can be very different from one business function to another. This influences the organizations to write hundreds of transformation scripts/program codes for hundreds of file formats. Further, this brings enormous difficulties when modifying the file format with respect to the time and the development effort.

Versatile Inter System Information Exchange Facilitator(VISIEF) is a novel software tool which assists business organizations by eliminating or minimizing the script writing in data exchange using flat files. Further, this provides a range of value adding features which improves the staff efficiency.

Keywords— **Offline data exchange, Data interoperability, Data migration**

I. INTRODUCTION

Presently, data interoperability between applications is a vital factor among the organizations. Offline data transmission using files supports the organizations to exchange data and perform batch processing. This optimizes the time, cost and the resources.

Standard information exchange protocols are widely used in different market segments. e.g. In financial markets, FIX (Financial Information eXchange Protocol)[1] and SWIFT(Society for Worldwide Interbank Financial Telecommunication)[2] protocols are commonly used. Those are specifically designed for real time messaging and supporting tools are also readily available.

On the contrary, there are quite a lot of applications which use non-standard/proprietary formats for their information exchange requirements. Due to their tailored nature, it is hard to find automated tools to support this kind of information exchanges.

Versatile Inter System Information Exchange Facilitator (VISIEF) is a software tool which supports such offline data transmission with non-standard/proprietary file formats.

Irrespective of whether it is a data migration or a data populating to a data warehouse or a business activity which requires offline data transformation, any offline data exchange process consists of the three main activities.

A. Extract:

Data is extracted from the source/external system. Normally, extracted data is downloaded to a text file. The format of the file is specific to a particular business function and pre-defined as well. There can be several business functions in an organization. Hence, information is exchanged across organizations in multiple file formats.

B. Validate/Transform:

Perform validations, transformations to data and prepare the extracted data for uploading to consumer application.

C. Transfer/Load:

Transfer the validated/prepared data to the consumer system (destination database).

In data warehousing, above mentioned three activities are called as ETL transactions. There are various commercial software tools available for ETL specifically designed for populating Data Warehouses[3]. Such tools may not provide optimized benefits to other uses such as data migrations (which normally perform as a onetime task) or frequent information exchanges (file uploads) to systems as pre-defined operational activities. On the other hand, there are software tools available specifically designed for data migration such as appMIGRATE. appMIGRATE is a powerful tool to extract, transform, validate, cleanse, correct and migrate data from an older version of Oracle E-Business Suite or any legacy system into newer version of Oracle EBS[6]. These tools are designed for specific tasks. Therefore, organizations tend to write tailored scripts for their various business information exchanges.

Ex: In some stock exchange websites (National Stock Exchange –NSE India[4] and Bombe Stock Exchange – BSE India[5]), they have specific pages developed in their web sites for facilitating data downloads for other business organizations such as Stock Broker Companies.

These offline data transmission procedures require high development effort on writing and maintaining tailor made program scripts. VISIEF novel tool eliminates these burdens from the business organizations.

The main idea behind this tool is to identify common activities in file processing and provide a generalized user interface to efficiently execute such common functionality with minimum or no development effort, irrespective of their proprietary nature.

Further, this tool provides various other value adding features, which are frequently used in offline data transmissions. (Discussed under section V - Value adding features)

978-1-4799-7732-1/14/$31.00 ©2014 IEEE

With respect to the development platform, this is a prototype developed using Oracle PL/SQL (Backend) and Java (Frontend).

VISIEF tool is greatly appropriate for data migrations due to its database centric processing nature and its higher customization capability. Further, this can be used as a low cost tool for data population in data warehousing too. This multipurpose capability brings the real sense of the word 'Versatility' to the VISIEF tool.

II. Design Architecture

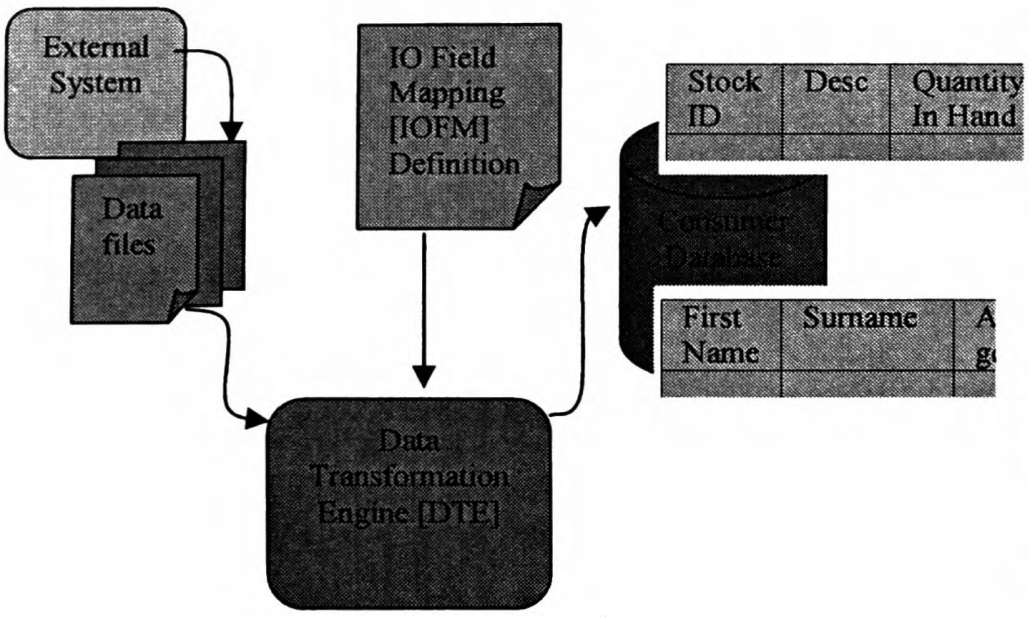


Fig. 1 System overview

As depicted in Fig.1, data is extracted from external system in to data files. A Legacy system in data migration and Transactional database in data warehousing, could also be considered as the External Systems here. VISIEF tool provides a generalized as well as a configurable user interface (UI) to convert such data files with proprietary file formats in to a machine readable object called Input Output Field Mapping (IOFM) Definition which describes format of the file.

A well sophisticated processing unit called Data Transformation Engine (DTE) reads the IOFM Definition, validates/processes and transfer the data in to the Consumer/Target system, according to the IOFM Definition that was read prior to processing. The main inputs for DTE are

1) *Data file.(Refer Fig. 2 for sample data file)*
2) *Related IOFM Definition*

III. VISIEF Components

A. Data file formats

Any business function which requires an offline data transmission will have a pre-defined file format. As an example, Tables I, II and III below, represent a sample data file format which can be used to transfer client information. These formats could be part of the Software Requirement Specification(SRS) too.

TABLE I
HEADER RECORD FORMAT OF CLIENT FILE

Field ID	Field Type	Field length	Value
Record Type	Character	01	'H'
File Date	Date	10	Ex. '01/01/2014'
File Name	Character	30	'Client_Registration.csv'
No of detail Records	Number	12	Ex. 10000

TABLE II
DETAIL RECORD FORMAT OF CLIENT FILE

Field ID	Field Type	Field length	Value
Record Type	Character	1	'D'
Client ID	Character	15	
First Name	Character	20	
Surname	Character	20	
Other Names	Character	100	
Date of Birth	Date	10	
..			

TABLE III
TAIL RECORD FORMAT OF CLIENT FILE

Field ID	Field Type	Field length	Value
Record Type	Character	1	'T'
Checksum	Number	32,2	Ex. 5000000

B. Input Output Field Mapping (IOFM) Definition:

This is the key component of the tool. This can be considered as the communication media between machine and the business. IOFM Definition is a machine readable standardized specification which represents the format of the business file (ie the **Input**) and the corresponding table and field of the destination system (ie. the **Output**). IOFM Definition is the place where this mapping is recorded. Hence, it is named as Input Output Field Mapping Definition (IOFM Definition). This tool provides a rich user Interface to capture business file format information. Any business user, with no programming experience, can create an IOFM Definition for a business function using this interface.

IOFM Definition creation is a onetime setting up. Once it is set up, any number of modifications can be easily performed and then system starts processing it accordingly. Changing the file formats is just a matter of changing the IOFM Definition. Code level modifications are not required. Hence, business users can implement format changes easily and on their own. Further, it supports processing files which have multiple record types. (e.g. Header record, Detail record types and a tail record etc.)

Captured data for IOFM Definition can be categorized in to two types as follows;

1) File Descriptor

This includes basic information relates to the business function and the data file such as business function ID, function name, file type (delimited or fixed length) etc.

2) Fields Identifier

This includes the details about the rows (record types) and columns (field types) of the data file. Data file formats discussed above (Section *A*) can be used to enter data in to this Fields Identifier.

Table IV and V below depicts the two categories of data which is captured in the IOFM Definition user interface.

TABLE IV
DATA REQUIRED FOR IOFM DEFINITION- FILE DESCRIPTOR

Field	Description	Example
DEFINISION_ID	The unique ID used for particular file format. This is the key field used for searching.	'CLIENTS'
FUNCTION_NAME	Description of the Definition	'Client Registrations'
FILE_TYPE	Format of the file. Whether it's a .csv (delimited) file or a fixed length file.	If .csv, 'CSV' If fixed length, 'FXD'
FLD_SEPERATOR	The delimiter used in the file, If it's a csv file	, or | or ~ etc.
IGNORE_LINES_FIRST	when uploading, number of lines that should be ignored from the top of the file	If a header record exists as the first line and it has to be ignored, value of this can be set as 1
IGNORE_LINES_LAST	when uploading, number of lines that should be ignored at the end of the file	If a Tail record exists as the last line and it has to be ignored, value of this can be set as 1
PRIOR_UPLOAD_VALIDATION_METHOD	Function name which performs prior upload validations. (File name and upload user validation can be done here.)	
VALIDATION_METHOD	Function name which performs validations upon the data of a file upload instance.	
PROCESSING_METHOD	Function name which performs additional processing upon the data of a file upload instance.	
FXD_LEN_REC_TYPE_LENGTH	If the file type is a fixed length file, It is required to define the length of the record type	Default value is 3
IS_AON	Whether the file should be processed as All or None basis or not	'Y' or 'N'
RECORD_TYPE_AT	In the data file,	Values:
	Record type is at the beginning of the record or at the end of the record(first column or last column)	'FIRST' or 'LAST'
LAST_USED_DEF_INSTNCE_ID	Last used definition instance ID.	This is maintained by the system.
CREATED_BY	User who created the definition	
CREATED_DATE	Definition created date	

TABLE V
DATA REQUIRED FOR IOFM DEFINITION- FIELDS IDENTIFIER

Item	Description	Example
DEFINISION_ID	The unique ID used for particular file format	'CLIENTS'
RECORD_TYPE	Record type of the record in the upload file. If the file has single format of data in all the records and, value of this field defaulted as 'N/A'	Header record = 'H', Detail record = 'D', Tail record ='T' etc. or '01','02' or 'N/A'
FIELD_ID	ID which demarcate column name	'LAST_NAME'
FIELD_TYPE	Type of the column data	VARCHAR2 or NUMBER or DATE (PL/SQL data types are used here with the hope of using them as is when auto creating the intermediate table(discussed under Instance Handling)
FIELD_TYPE_LEN	Length of the VARCHAR2 column or Number column	'31,2'
DATE_FORMAT	Format of the date. Mandatory if the field type is DATE.	Ex: 'D D/MM/YYY'
FIELD_ORDER	Exact order of the columns in the upload file (whether the field comes in the file as the first column or second column etc.)	
DEST_TABLE	Which table the data should be transferred to	Ex: 'REGISTERED_CLIENS'
DEST_TABLE_COLUMN	Which column the data should be transferred to	Ex: 'SURNAME'

Item	**Description**	**Example**
FIELD_STATUS	Whether to consider the field as an Active one or Inactive one.	
FIELD_LABEL	Label to display in the FE	
FIELD_OFFSET	Used in fixed length files. It consists of starting point and the no of characters of the fields	Field Offset of first column = 1,1 Second column = 2,7etc. e.gHxxxxxxx
IS_MANDATORY	Perform Mandatory validation upon the field data.	'Y' or 'N'

Once the required data for IOFM Definition is captured, it is stored in the data base in two tables (with Master Detail relationship connecting using DEFINISION_ID) as in TABLE VI and TABLE VII. (Refer Fig 3 and Fig 4 to see a sample definition created in the database)

TABLE VI
FILE DESCRIPTOR DATABASE TABLE (MASTER)

DEFINISION_ID	FUNCTION_NAME	FILE_TYPE	FLD_SEPERATOR	..
CLIENTS	Client Registrations	CSV	~	
..	..	..	..	..

TABLE VII
FIELD IDENTIFIER DATABASE TABLE (DETAIL)

DEFINISION_ID	RECORD_TYPE	FIELD_ID	FIELD_TYPE	FIELD_LENGTH.
CLIENTS	H	RECORD_TYPE	Varchar2	1
CLIENTS	H	FILE_DATE	Date	10
CLIENTS	H	FILE_NAME	Varchar2	30
CLIENTS	H	DETAIL_RC_COUNT	Number	12
CLIENTS	D	RECORD_TYPE	Varchar2	1
CLIENTS	D	CLIENT_ID	Varchar2	15
CLIENTS	D	FIRST_NAME	Varchar2	20
CLIENTS	D	SURNAME	Varchar2	20
CLIENTS	D	OTHER_NAMES	Varchar2	100
CLIENTS	D	..	..	
CLIENTS	D	..	..	
CLIENTS	T	RECORD_TYPE	Varchar2	1
CLIENTS	T	EOF INDICATOR	Varchar2	2

C. *Data Transformation Engine (DTE)*

This is the most important processing component of the tool. It's an API that consists of several methods that can be called by any external application. Implementation of this component is done using PL/SQL stored procedures that use ORACLE.DBMS.UTL_FILE library[8]. The two main inputs for this engine are,

1) *Data file*

2) *Definition ID of its related IOFM Definition.*

In order to initiate the processing at DTE, above inputs should be provided. In this prototype tool, File Upload Interface(FUI) is currently in place for the same purpose.

D. *File Upload Interface (FUI):*

This interface provides the facility to the user to upload data files for processing. There are two options available in this interface. The first option is, upload from the local host. Assuming that the file is already available in the local machine, FUI provides a standard open window to select the file for processing. Then the uploaded file is converted to a CLOB and sent to the DTE (Backend).

The second option is, user can keep the data files anywhere in the database server (Backend). File Upload Interface shows the database server locations and helps the user to select the correct file and the location. Then it sends the file path and the file name to the DTE. DTE locates the file using the given path and start processing it.

Since, DTE is written as an API, this FUI is not mandatory to be used. API methods can be called by any application as long as it provides the necessary inputs correctly.

Processing in DTE:

Processing in DTE is done based on the relevant IOFM Definition. It reads the data file, extracts data according to the definition, performs validations (field type validation, mandatory check etc.) and writes the data in to the target database.

e.g validation: If definition indicates that the third column of the Detail type row ('D') should be a date field and format should be 'ddMonYYYY', then, while reading the file, system performs that validation, when it finds a (D)detail type row.

While processing the file line by line iteratively, it generates the dynamic insert statements for the destination table/s. This SQL statement is generated dynamically by referencing the IOFM Fields Identifier.

e.g.

'Insert into <Destination table name> fields (<Destination column1>, <Destination column2>,<Destination column3>) values (<Data column1>, <Data column2>,<Data column3>);'

Data is loaded in to the destination database table using such insert statements. These dynamic insert strings are executed in the database using the EXECUTE IMMEDIATE Statement in Oracle PL/SQL[7].

IV. ERROR HANDLING

While the data file is being processed, numerous errors can turn up. Those errors are logged in to a table called error log and it is viewable by the users. If any critical error found (showstopper) system terminates the processing by giving a meaningful error message to the user.

A. *All Or None (AON) Flag:*

This is an attribute which can be found in the File descriptor of the IOFM Definition.

AON flag = 'N' ->If an error occurred in a line, skip that line and continue processing.

AON flag = 'Y' ->If an error occurred in a line, terminate the file processing.

V. VALUE ADDING FEATURES

A. *Instance handling*

There are some situations which the business requirement is to upload data in batch wise and maintain each batch as a separate instance of the IOFM Definition. Further, they need to perform their own validation methods before transfer to target table. Considering this valid requirement, Instance handling feature was introduced to the VISIEF tool.

1) Method of implementation:

Once the IOFM definition is created, system generates an intermediate table using its Field Identifier. That table can be considered as a union of fields in the Field Identifier. Rather than uploading data directly to a destination table, system loads the data in to this intermediate table. Each batch of data is tagged with an instance ID. A separate interface has been created to maintain these instances. Following activities can be performed for the instance data.

1. Uploaded data is viewable. A running variable is maintained to monitor the status of the instance data. (Uploaded ->validated->Transferred etc.)
2. User can write their own validation and transfer methods (PL/SQL) which can run on top of the instance data stored in the intermediate table. 'VALIDATION_METHOD' and 'TRANSFER_METHOD are two attributes in the File Descriptor of the IOFM Definition to capture the names of such methods. (Refer TABLE IV) Interface allows the user to invoke to this validation and Transfer methods. Errors are logged in to an error log.
3. There could be a necessity for passing parameters to validation and processing methods discussed above at run time. At the time of file upload, these parameters can be concatenated into one string using a delimiter and input as one string. It is stored in the database as an attribute of the instance. Validation or Transfer process should know the format of this string and extract it from the table which instance information is recorded in the database, before executing the Validation/Transfer method.
   e.g. If interest should be calculated before transfer to target table, prevailing interest rates can be passed as parameters.
4. Once the instance data is transferred to destination table, it is no longer required to store in the intermediate table. Hence, user is given the facility to purge the instance data.

B. Possibility to upload files with multiple record types

Many file formats consist of multiple record types. IOFM Definition allows the user to define fields of multiple record types. When the DTE extracts a line, definition tells which column represents the 'Record type'. Then it fetches field information of that Record type from the IOFM Definition. Line processing is performed according to that information.

If the data file format consists of single record type (all records have the same format), when Field Identifier is created in IOFM definition, Record type is defined as 'N/A'. Then, DTE considers that format as a file format which has a single record type. Refer RECORD_TYPE at TABLE V.

C. Support different file types

This tool provides the facility for the user to transfer data through both delimited and fixed length file types. If it is a fixed length file, FIELD_OFFSET becomes a mandatory item for the Field Identifier. (Refer FIELD_OFFSET at TABLE V)

D. Possibility to ignore Header and Footer lines

Few lines as header information at the top and few lines as footer information at the bottom of the file are common in some file formats (Like a report). If such data is not required to process, Field Identifier of IOFM Definition allows the user to indicate number of lines to be ignored at header area and footer area in the data file. Refer IGNORE_LINES_FIRST and IGNORE_LINES_LAST fields at TABLE IV.

E. Prior upload validation method

There are some instances that require some validations to be done, prior to upload the file.

e.g. file name validation, user validation

Facility is available here to define a validation method in the File Descriptor of the IOFM Definition. Prior to start file processing by DTE, it will be executed first. If such validation is failed, processing is not proceeded. Refer PRIOR_UPLOAD_VALIDATION_METHOD at TABLE IV.

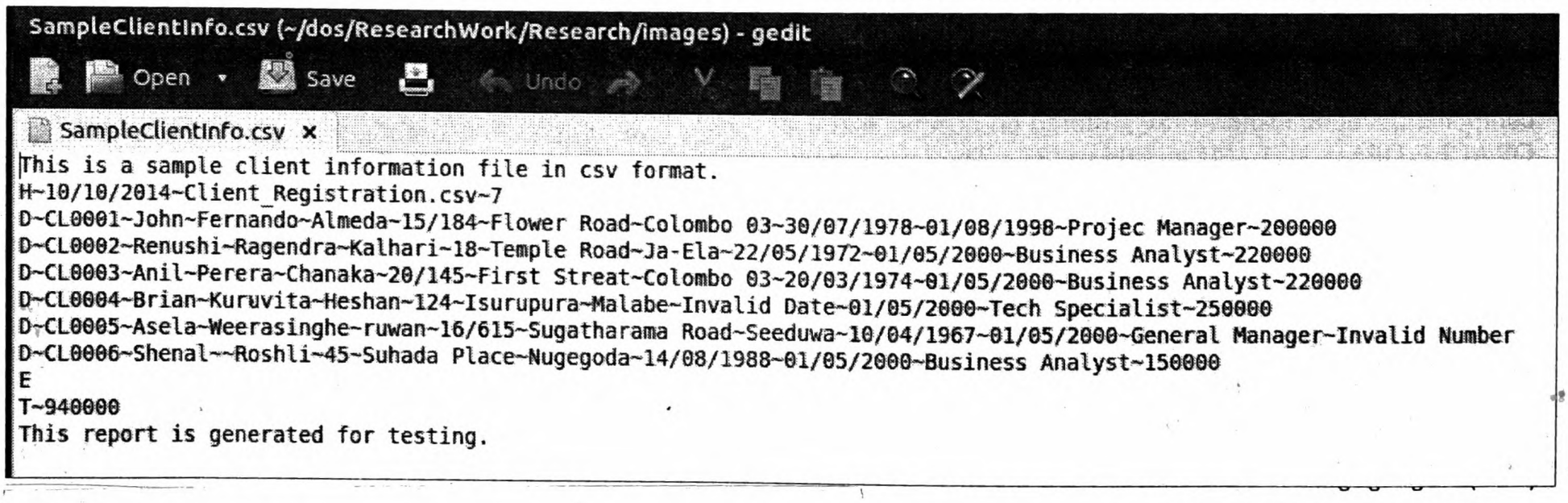
VI. RESULTS AND EVALUATION

The primary intention of this section is to discuss about the practical usage of the tool while evaluating the results.

First, VISIEF tool (Backend) should be installed in the destination database. It should be run in the destination database. Fig. 2 is a sample data file which can be assumed as downloaded from a source application and ready to upload to the destination database using VISIEF tool.

Fig. 2 Sample data file for offline Client Registrations

The above file contains some downloaded data from the



source application system that relates to Client Registrations. It has three record types; Header 'H', Detail 'D' and Tail 'T'. Further, first and last lines carry non-processing information to be ignored while uploading. (TABLE VIII consists of few test cases which help to evaluate the results.)

Next, the IOFM Definition should be created for "Client Registration" offline data transmission. As mentioned earlier, it should have both a File Descriptor and a Fields Identifier. Fig. 3 and Fig. 4 show, how the created definition is stored in the database tables.

Once the IOFM Definition is created, an intermediate table is created by the system which used to store upload data.

Refer Fig. 5 to view the system generated table created in the database to store the upload data of the "Client Registration" Definition.

Definition Creation and intermediate table creation are one time tasks which is facilitated by rich user interfaces and any business user can do it alone.

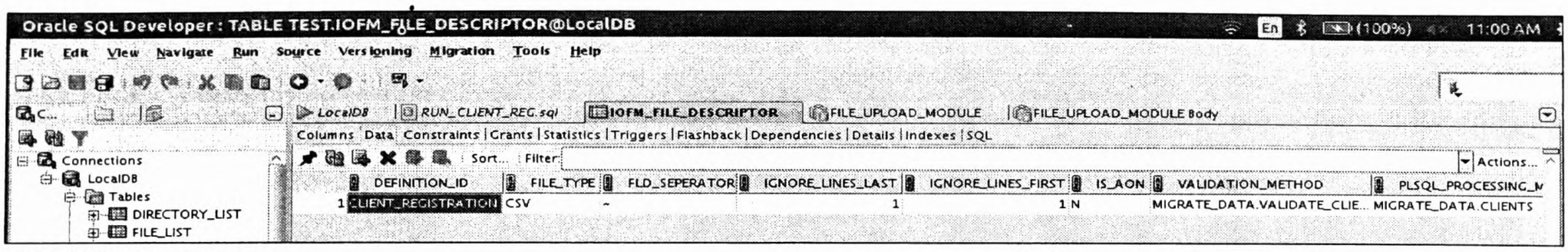


Fig. 3 File Descriptor Data

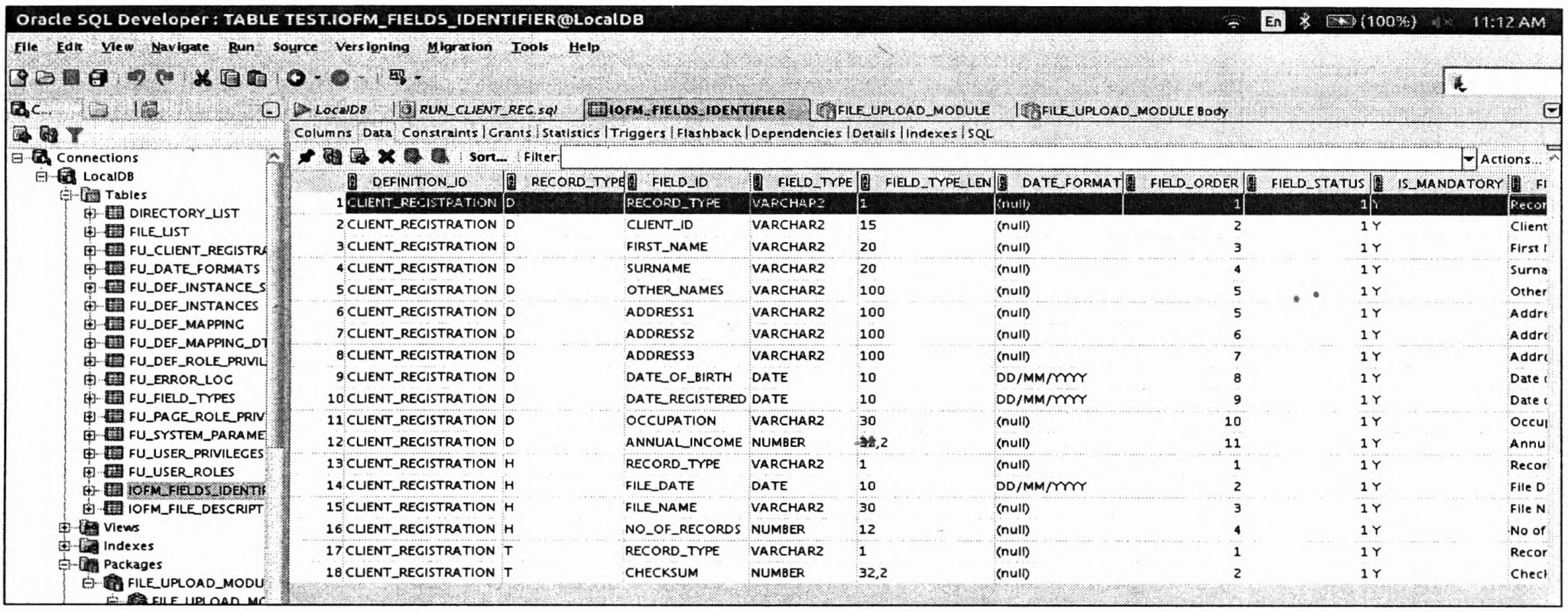


Fig.4 Fields Identifier Data

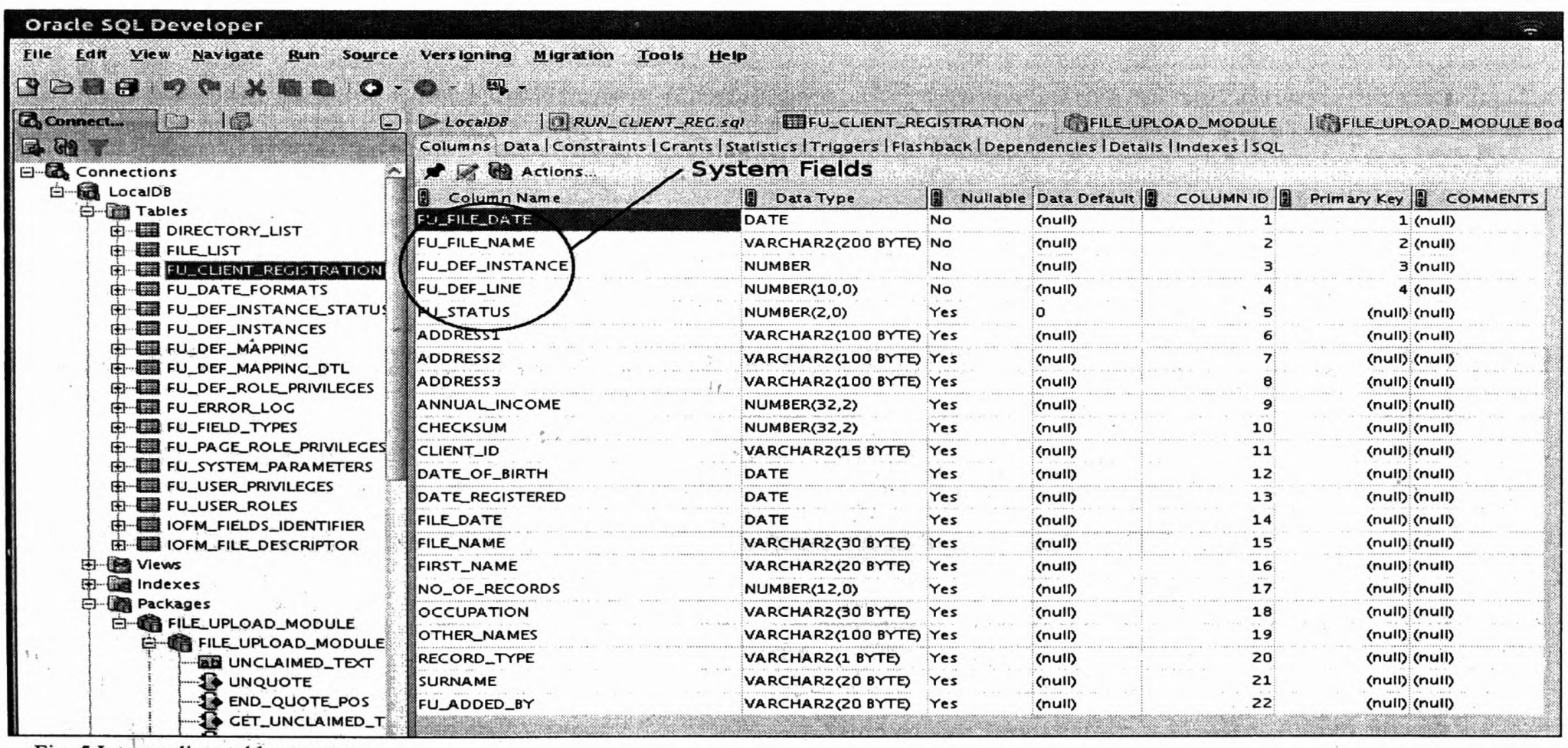


Fig. 5 Intermediate table structure

Below shows the signature of the main function created in the DTE API. It reads the data file (P_FILE_NAME) from the P_FILE_PATH and start processing according to the IOFM Definition passed in the P_DEFINITION_ID parameter.

```
FUNCTION READ_FILE( P_DEFINITION_ID        IN
        IOFM_FILE_DESCRIPTOR.DEFINITION_ID%TYPE,
        P_FILE_NAME                        IN VARCHAR2,
        P_FILE_PATH                        IN VARCHAR2,
        P_USER_ID              IN VARCHAR2,
        P_FLUSH_DATA           IN VARCHAR2,
        P_OTHER_PARAMS         IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2 IS
```

Using the tool, sample data files were uploaded under CLIENT_REGISTRATION Definition. Some test cases executed were discussed in TABLE VIII. Fig.6, and Fig.7 are the screen shots (sections from left to right) taken from the intermediate table view which, data has been successfully populated from the sample data file.

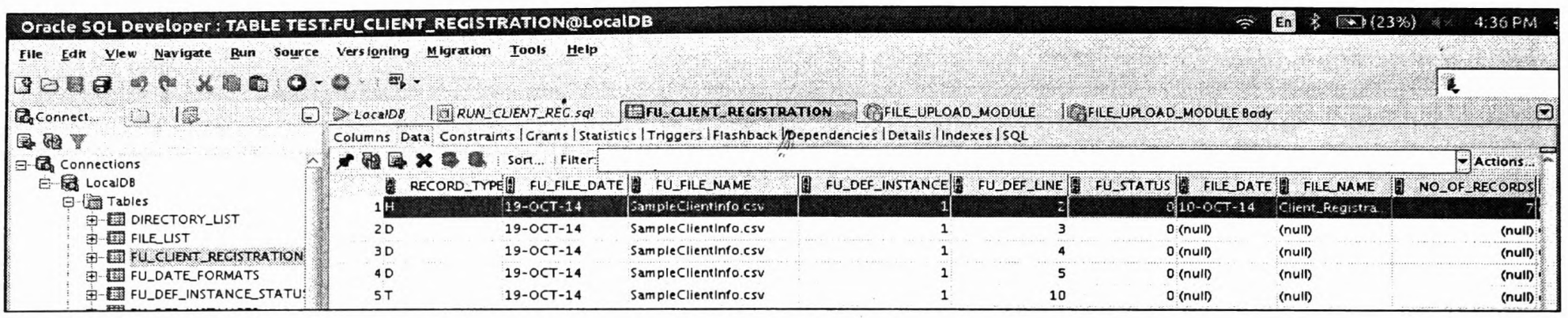


Fig. 6 Populated data section 1

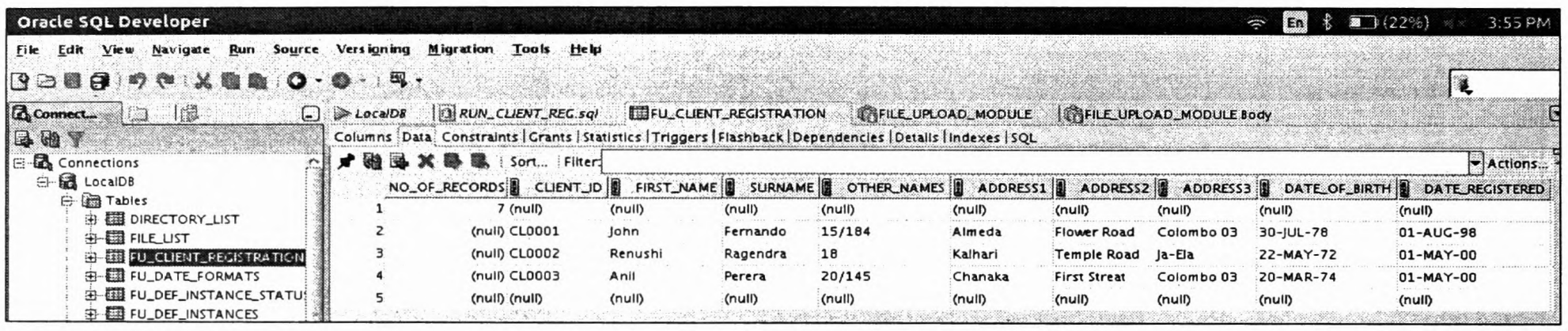


Fig. 7 Populated data section 2

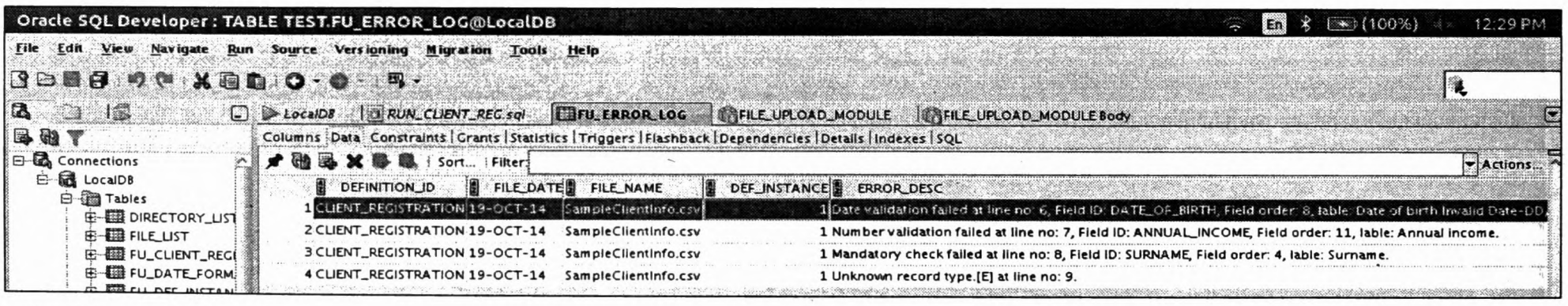


Fig. 8 Error Log

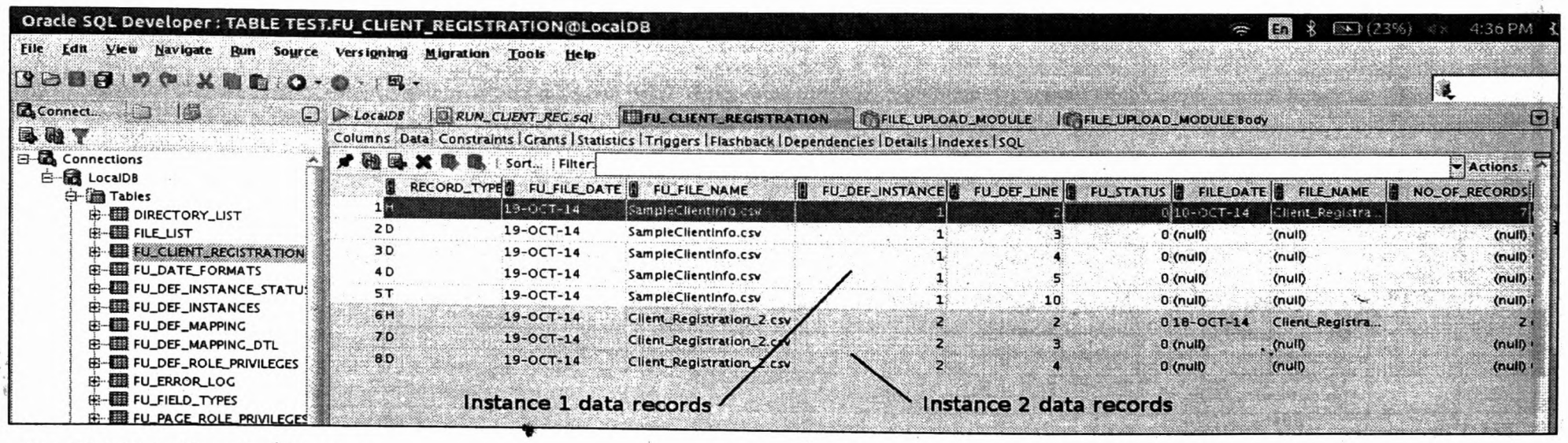


Fig. 9 Multiple file upload Instances

TABLE VIII

TEST CASES TO EVALUATE RESULTS

Test case	Description	Results
1	In the IOFM FILE DESCRIPTOR, IGNORE_LINES_FIRST and IGNORE_LINES_LAST fields has been set as 1	First line and Last line of the data file were some data which does not need to process and the tool has ignored it while uploading.
2	In the IOFM Fields Identifier, FIELD_TYPE parameter has been set as DATE for Date of Birth. However, in line number 6 at the data file has an Invalid value for it.	Line number 6 has not been upload to the intermediate table but error has been logged in to the Error Log. Refer Fig. 8.
3	In the IOFM Fields Identifier, FIELD_TYPE parameter has been set as NUMBER for Annual Income. However, in line number 7 at the data file has an Invalid value for it.	Line number 7 has not been upload to the intermediate table but error has been logged in to the Error Log. Refer Fig. 8.
4	In the IOFM Fields Identifier, IS_MANDATORY parameter has been set as 'Y' for the field 'Surname'. But it is null in the data file at line number 8	Line number 8 has not been upload to the intermediate table but error has been logged in to the Error Log. Refer Fig. 8.
5	Line number 9 has unknown record type which is not defined in the IOFM Definition.	Line number 9 has not been upload to the intermediate table but error has been logged in to the Error Log. Refer Fig. 8.
6	In the IOFM File Descriptor, IS_AON parameter has been set as 'N' for the Definition ID 'CLIENT_REGISTRATION'	VISIEF has ignored error lines and other lines have been successfully uploaded.

Once the definition is created as a onetime task, the user can upload any number of files for the same definition. While upload, data records are tagged with the Instance ID. User can query data records of a particular instance using the Instance ID and perform further business validations and transform to other tables. System facilitates this by allowing the user to define the pre-defined procedures at the IOFM File Descriptor ('VALIDATION_METHOD' and 'TRANSFER_METHOD).Fig 9 shows how data is tagged with an instance ID and stored in the database.

VII. CONCLUSION AND FUTURE WORK

Today, there are numerous technologies and software tools available for data exchange. Most of them serve real time data transmissions. Facilities available in the current industry for offline data exchange are not adequate to obtain competitive advantage by the business organizations. This situation is worst in data exchange using non-standard/proprietary file formats.

Versatile Inter System Information Exchange Facilitator [VISIEF] is merely designed to fill this gap. It provides a user friendly interface to convert the entire data exchange requirement in to a machine readable object. This minimizes/eliminates the development effort and expedites the file format modifications. Hence, speeding the Time To Market(TTM).

In addition, VISIEF enables the business users to actively participate in implementation of business function. This, in turn improves the employee motivation.

Automation of generating data files by extracting data from external (Source) systems, is not covered in this attempt. This tool can be further improved by adding that facility in to the scope.

ACKNOWLEDGMENT

I would like to convey my gratefulness to my husband and other family members to motivate me in developing this prototype.

REFERENCES

[1] Darren DeMarco,'*Exploiting Financial Information Exchange (FIX) Protocol?*', 2012 The SANS Institute

[2] Liam Sherlock,'SWIFT Messaging The testing journey' White paper

[3] PanosVassiliadis, *A Sumrvey of Extract–Transform–Load Technology*, International Journal of Data Warehousing & Mining, 5(3), 1-27, July-September 2009.

[4] Boston Stock Exchange (BSE) file download page, [Online] http://www.bseindia.com/members/downloads.aspx?

[5] National Stock Exchange (NSE) file download page, [Online] http://www.nseindia.com/products/content/derivatives/equities/data_report_margin.htm.

[6] appMIGRATE - a next generation data migration tool for Oracle E-Business Suite R12/11i,[Online] http://www.chain-sys.com/appmigrate.shtml?gclid=CJ2Y26PduMECFRcMjgodfjEALw

[7] Sheila Moore, Belden E., 'Oracle Database PL/SQL Language Reference, 11g Release 1 (11.1)', 271-276, 2009

[8] Denis Raphaely,'Oracle® Database PL/SQL Packages and Types Reference 11g Release 2 (11.2)',5089-5125, 2013