International Research Journal of Computer Science (IRJCS)
Issue 6, Volume 2 (June 2015)
ISSN: 2393-9842
www.irjcs.com

An Enhanced Data Validation System for a Relational DataBase

Barida Baah*
Computer Science Department,
University of Port Harcourt, Nigeria

Ledisi Giok Kabari
Computer Science Department,
Rivers State Polytechnic, Bori, Nigeria

Abstract— This paper, an enhanced data validation system for a relational database, x-ray the perception of a relational database, data validation methods, techniques and how to develop an enhanced data validation system for a relational database. Indeed, one of the most neglected areas in systems implementation is data validation. A good number of application systems fail to achieve its set goals because there is no adequate and efficient data validation system to filter out input data that is not correct. Empirical studies show that improper input data is responsible for a large number of system failures; and such failures could have been avoided or reduced if proper input data had been fed into the system. In this paper, we have focused on enhanced data validation and elaborating on existing data validation methods like limits checks, allow character checks, range checks, presence checks, consistency checks, format or picture checks and data type checks to improve the performance of our application systems. We present algorithms and flowcharts of some of these data validation methods, an enhanced data validation and techniques. We also develop a trial program that integrates these validation techniques. Finally, a report was generated based on validated customer records created for Fractal Construction Nigeria Limited. The system was developed using PHP, MYSQL and XAMPP web server.

Keywords— Enhanced Data validation, Data Validation Techniques, Relational Database, Relational Model, Database Architecture, Database Specification

I. *INTRODUCTION*

Data which is seen as an integral part of any information system, must be handle with much more care in order to ensure that the right information needed for any decision making processes are actually achievable. To be able to achieve this, there is the need to validate such data before processing, in order to ensure that the right data are entered correctly in their respective field(s) after the program implementation of data validation system for a relational database, which is the purpose for this research work.

In most of the business application, data validation can be defined through declarative, data integrity rules or procedural based rules. Data that does not conform to these rules will negatively affect business process execution.

Therefore, data validity should start with business process definition and set of business rules with this process. It checks to ensure that data are valid, sensible, reasonable and secure before they are processed. Data validation involves the process of ensuring that a program that is developed operates on a clean, correct and useful data. The validation of this data is applied to a relational database [1].

The aim of the research paper is to develop software that will check data input in various field(s) to ascertain its correctness, thereby making sure that only valid data are accepted and invalid data are rejected. Specifically, the objectives are to develop an error free system for the processing of Customers records and to generate a report from the user input after ensuring that the data validated are clean, correct and useful for decision making processes.

II. *LITERATURE REVIEW*

A. *Data Validation*

Data validation is seen as the process of ensuring that a program operates on clean, correct and useful data. It often uses routines, usually called “Validation Rules” or “check routines”, that does the checking of data that are input to the system. The rules may be implemented through the automated facilities of a data dictionary, or by the inclusion of explicit application program validation logic.

A validation rule is a criterion used in the process of data validation, carried out after the data has been encoded onto an input medium and involves a data vet or validation program. This is distinct from formal verification, where the operation of a program is determined to be that which was intended, and that meets the purpose. The method is to check that data fall the appropriate parameters defined by the systems analyst. A judgment as to whether data is valid is made possible by the validation program, but it cannot ensure complete accuracy.

Luhn[2] introduce a formula (also known as modulus 10 or mod 10-algorithm) is used to generate and or validate and verify the accuracy of the credit-card numbers. Most credit cards contain a check digit, which is the digit at the end of the credit card number. The first part of the credit-card number identifies the type of credit card (Visa, MasterCard, American Express, etc.), and the middle digits identify the bank and customer. In order to generate the check digit, the Luhn formula is applied to the number. To validate the credit-card number, the check digit is figured into the formula.

Steps on how Luhn algorithm works: 1: Starting with the second to last digit and moving left, double the value of all the alternating digits. 2: Starting from the left, take all the unaffected digits and add them to the results of all the individual digits from step1. If the results from any of the numbers from step1 are double digits, make sure to add the two numbers first (i.e. 18 would yield 1+8). Basically, your equation will look like a regular addition problem that adds every single digit. 3: The total from step 2 must end in zero for the credit-card number to be valid.

Verhoeff[3] generated an algorithm which is called the Verhoeff algorithms, a checksum formula for error detection. Like the more widely knownLuhn algorithm, it works with strings of decimal digits of any length. This algorithm perform much better than the Luhn algorithm due to the fact that it is able to detect all the transposition error which was found in the Luhn algorithm (switching of two adjacent digit) as well as catching many other types of errors that pass the Luhn formula undetected. The algorithm uses the properties of D2 (the dihedral group of order 10) a non-commutative system of operations on ten elements, corresponding to the results of rotating or reflecting (flipping) a regular pentagon. It is implemented using three tables: a multiplication table, a permutation table and an Inverse table.

Steps on how Verhoeff algorithm works: 1: Create an array *n* out of the individual digits of the number, taken from right to left (rightmost digit is n_0, etc.). 2: Initialize the checksum c to zero. 3: For each index i of the array n, starting at zero, replace c with d(c, p(i, n_i)).

The original number has a valid check digit if and only if $c = 0$. If the original number ends in a zero (i.e., $n_0 = 0$), then *inv(c)* is the proper value to use as the check digit in place of the final zero.There are five most recent approaches that attempt to address the lack of data validation problems, which are application-level gateway approach, client-side encryption approach, WAVES approach and Pixy, and Saner.

Scott and Sharp[4] have proposed a gateway model which is an application-level firewall on a server for checking invalid inputs and detecting malicious script (e.g SQL injection attack and cross-site scripting attack). They have developed a security policy description language (SPDL) base on XML to describe a set of validation constraints and transformation rules. This language is translated into code by a policy compiler, which sent to a security gateway on a server. The gateway analyzed the request and augments it with a Message Authentication Code (MAC1). Mac is used to protect the data integrity. For example, the MAC is used to secure session information in a cookie at the client-side. There are many ways to compute a MAC such as one-way hash algorithms (MD5,SHA-1) to create a unique fingerprint for data within the cookie.

However, this approach has a number of limitations. One of these is that data tampering is still a potential problem because dynamic data that is generated on fly is not validated. It is also difficult to define all the policies and rules of a legacy web application for every single data entry point.

Hassinen and Mussalo[5] have both also proposed a client-side encryption system to protect confidential, data integrity, and user trust. They encrypt data inputs using a client encryption key before submitting the content of an HTML form. The client encryption key is stored on the server and transferred over an HTTP connection. It uses a one way hash function. The message validation includes finding a new hash value from the decrypted message and comparing it to the hash value which is received with the message. If they are the same, the data is accepted, otherwise, the data is deemed to have been altered and the validation will fail.

Huang et al [6] were the first who attempted to address data validation issue in the context of PHP applications. They used a lattice-based analysis algorithm derived from type systems and type-state systems. The type of analysis in this algorithm is static analysis. WAVES approach is targeted for mitigating threats to web application at the server-side.

Jovanovic et al. [7] have developed Pixy, which is the first open source tool for statically detecting XSS vulnerabilities in PHP 4 code by means of data follow analysis which is based on a static analysis technique. They adopted PHP as target language since it is commonly used for developing web applications and substantial numbers of security advisories refer to PHP programs.

Cova et al [8] have presented an approach to the analysis of the sanitization. In this approach they combine both the static and dynamic analysis techniques to identify faulty sanitization procedures that can be bypassed by the criminal. Saner implemented the approach on both static and dynamic analysis techniques in a number of real-world web applications. The static analysis technique is characterizes the sanitization process by modeling the way in which a web application processes input values. This permits us to define the cases where the sanitization is incorrect or incomplete. Furthermore, they introduced a dynamic analysis technique that is able to reconstruct the code that is responsible for the sanitization of application inputs, and then execute this code on malicious inputs to identify faulty sanitization of application inputs, and then execute this code on malicious inputs to identify faulty sanitization procedures.

Furthermore, for any business applications, data validation can be defined through declarative data integrity rules, or procedure-based business rules. Data that does not conform to these rules must negatively affect definition and set of business rules within this process. Rules can be collected through the requirements capture exercise.

The simplest data validation verifies that the characters provided come from a valid set. For example, Phone numbers should include the digits and possibly the characters +, - (, and) (plus, minus and parenthesis). A more sophisticated data validation would check to see the user had entered a valid Country code, i.e., that the number of digits entered matched the convention for the Country or area specified.

B. *Existing Methods of Data Validation*

There are several existing methods of validating data, some of which are enumerated and discuss in this paper.

1) Allow Character Checks: This is one of the methods for validating data, it checks to ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as personal name might disallow characters such as < and >, as they could be evidence of a mark-up based security attack. An e-mail address might require exactly one @ sign and various other details. A regular expression is effective ways of implementing such checks.
2) Batch Total Checks: In this case, it checks for missing records, numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g. add the 'Total cost' field of a number of transactions together.
3) Cardinality Checks: The cardinality checks ensure that record has a valid number of related records. For example if contact record classified as customer it must have at least one associated order (cardinality >0). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record Payroll database is marked as "Former employee", then this record must not have associated salary payments after the data on which employee left organization (Cardinality=0).
4) Digit Checks: It is basically used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered.
5) Consistency Checks**:** This is use to checks fields to ensure that data in this fields corresponds, e.g. if Title = "Mr" then Gender ="Male"
6) Control Total Checks: Here, it check to ensure that a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of customers.
7) Cross-System Consistency Checks: In the cross-system consistency checks it compare data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems to ensure it is consistent, e.g., the address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First_Name (John), Last_Name (Doe) AND Middle_Name (Quality); to compare two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL: Last_Name !!','!! First_Name!! Substr(Middle_Name,1,1) would convert the data from the second system to look like the data from the first 'Doe, John Q'.
8) Data Type Checks :The data type checks only checks the data type that is inputted and give an error message(s) if the input data does not match with the chosen data type, e.g., In a input box accepting numeric data, if the letter 'o' was typed instead of the number zero, an error message would appear.
9) File existence Checks :It checks that a file with a specified name-exists. This check is essential for programs that uses file handling.
10) Format or Picture Check: The format or the picture checks is basically use to check that the data is in a specified format (template), e.g. dates have to be in the format DD/MM/YYYY. Note here that regular expression should be considered for this type of validation.
11) Hash Total Checks: It is a type of checks that is a batch total done on one or more numeric fields which appears in every record. This is a meaningfulness total, e.g., add the telephone numbers together for a number of customers.
12) Limit Checks: In this case, data is checked for one limit only, upper or lower, e.g., data should not be greater than 2 (<=2).
13) Logic Checks: For the case of logic checks, its check that input does not yield a logical error, e.g., an input value should not be 0 when there will a number that divides it somewhere in a program.
14) Presence Check: Checks that important data are actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.
15) Range Check: This is use to checks to ensure that data lie within a specified range of values, e.g., the month of a person's date of birth should be between 1 and 12.
16) Referential Integrity Checks: Referential integrity in a relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism, then they should be validated. Validation of the foreign key checks that referencing table must always refer to a valid row in the referenced table.

III. EXISTING DATA VALIDATION TECHNIQUES

A general rule is to accept only "Known Good" characters, i.e. the characters that are to be expected. If this cannot be done the next strongest technique is "Known bad", where we reject all known bad characters. The issue with this is that today's known bad list may expand tomorrow as new technologies are added to the enterprise infrastructure. Basically, there are number of models to think about when designing a data validation technique, which are listed from the strongest to the weakest as follows: Exact Match (Constrain), Known Good (Accept), Known Bad (Reject) and Encode Known bad (Sanitise).

IV. *ENHANCED DATA VALIDATION METHODS*

In this section we discover enhanced data validation methods as well as additional validation techniques that will improve the system performance. In our system, our enhanced data validation methods are as a result of combining or merging two or more existing validation techniques to ensure the consistency of data. A good example is the consistency check method that combines exact match techniques with the presence check techniques to test the title field. Assuming a user has selected Male (which satisfies the exact match technique), then the user must also select "Mr." to be consistent with the gender information that is required. Another example is what we call dependency checks.

This implies that there are certain checks that are dependent on others. These checks must be adjudged to be satisfactory before others checks can be performed. An example of this check is format check. In the format or picture check it check if the date field is of the format (dd/mm/yy) within this dd/mm/yy it dependants on the range or limit checks for parameters "dd" , "mm" and yy.

Allow-Combine Characters Check. The allow-combine characters check is the new method of data validation that we developed from the existing allow character check which allow for a check of just a single character like "@", dot (.) etc. but in the newly developed allow combined character checks tries to allow for the checks of more than one characters for example in the GSM field which is a numeric field will test to allow combine set of characters for a network particularly the first 3 digits, if it is correctly entered or not e.g. 080,070,081,082 etc.

V. *DATABASE ISSUES AS ARE USED*

A Database is a conglomeration of two words data and base; the data which is seen basic raw fact and the base refers to where the data resides. Database consists of an organized collection of data for one or more uses, typically in digital form.

A. *Relational Database*

A relational database can also be defines as a set of tables containing data fitted into predefined categories. Each table (which is sometimes called relation) contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns. For example, a typical business order entry database would include a table that described a customer with columns for name, address, phone number, and so forth. Another table would describe an order: Product, Customer, date, Sales price, and so forth. A user of a database could obtain a view of the database that fitted the user's needs. For example, a branch office manager might like a view or report on all customers that had bought products after a certain date. A financial service manager could from the same tables, obtain a report on accounts that needed to be paid.

B. *Relational Model*

The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as select to identify tuples, project to identify attributes, and join to combine relations. Relations can be modified using the inset, delete, and update operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting. It is necessary for each tuples of a relation to be uniquely identifiable by some combination (one or more) of its attribute values. This combination is referred to as the primary key.

C. *Base and Derived Relations*

In a relational database, all data are stored and accessed via relations. Relations that stored data are called "Base Relation", and in implementations are called "tables". Other relations do not store data, but are computed by applying relational operations to other relations. These relations are sometimes called "derived relations". In implementations these are called "view" or "queries". Derived relations are convenient in that though they may grab information from several relations, they art as single relation. Also, derived relations can be used as an abstraction layer.

D. *Relational Database Management System*

A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a relationship among a set of values. A relational database as implemented in relational database management system, have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data and much more.

Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use, even though they are much more less efficient as computer power has increased, the inefficiencies of relational database, which made them impractical in earlier times, have been outweighed by their ease of use. The three leading commercial relational database vendors are Oracle, Microsoft and IBM. The three leading open source implementation are MYSQL, PostgreSQL and SQLite.

VI. *RESEARCH METHODOLOGY*

The research methodology used here in this paper is called the structural system analysis and design method which is a waterfall method for the production of an information system design. Structural system analysis and design method (SSADM) can be thought to represent a pinnacle of the rigorous document-led approach to system design.

A. Why SSADM

Structural System Analysis and Design Method (SSADM) is a systems approach to the analysis and design of information systems. one particular implementation of structural system analysis and design method which is builds on the work of different schools of structured analysis and development methods, such as Peter Checkland's, software system methodology, Larry Constantine's Structured Design, Edward Yourdon's Structured Method, Michael A Jackson's, Jackson Structured Programming and Tom DeMarco's Structured Analysis [9].

This research methodology was chosen because The SSADM is mature, SSADM provide a clear separation of logical and physical aspects of the system, It is well-defined techniques and also well documented, and It also provide an environment for the user involvement.

B. *Database Architecture*

Database architecture consists of three levels, *external*, *conceptual* and *internal*. Clearly separating the three levels was a major feature of the relational database model that dominates 21st century databases. The external level defines how users understand the organization of the data. A single database can have any number of views at the external level. The internal level defines how the data is physically stored and processed by the computing system. Internal architecture is concerned with cost, performance, scalability and other operational matters. The conceptual is a level of indirection between internal and external. It provides a common view of the database that is uncomplicated by details of how the data is stored or managed, and that can unify the various external views into a coherent whole.

C. *Database Specifications*

In this paper, we used a relational DBMS called MySQL. The name of the database is called fractalConstruction.sql. It contains one major table-customers. Other tables are supportive in nature. The basic table structure is shown below in table 1.1

Table 1.1: Customer Table

Field	Type	Null
Sn	int(11)	No
date	varchar(10)	No
title	varchar(4)	No
name	varchar(50)	No
email	varchar(32)	No
Gsm	varchar(11)	No
gender	varchar(6)	No
address	varchar(100)	No
product_item	varchar(50)	No
amount	varchar(50)	No

D. *Architecture of the Proposed System*

The fig. 1, fig. 2 and fig. 3 below shows the architectural design of our proposed enhanced data validation for a Relational Database which contains: initial view of home page of the web application, the interface design for data input and an Interface design report design of data validated.

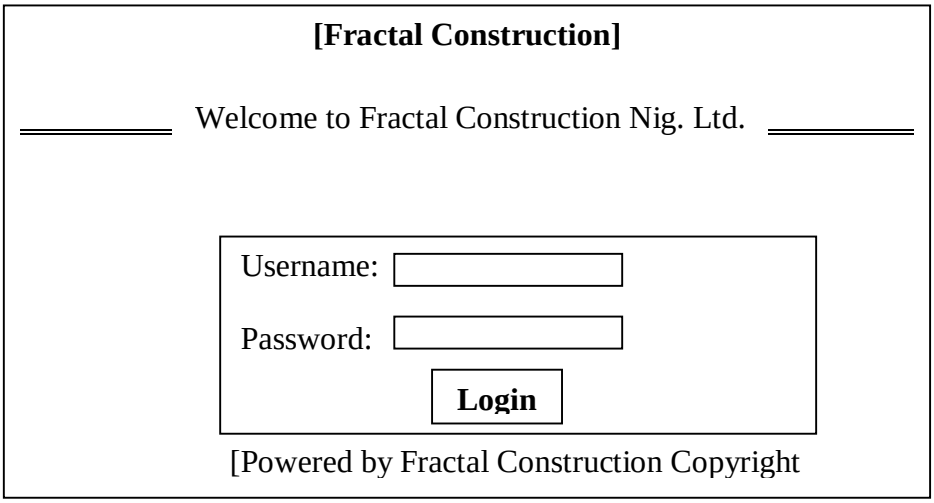

Fig. 1: Main menu design for the Proposed System

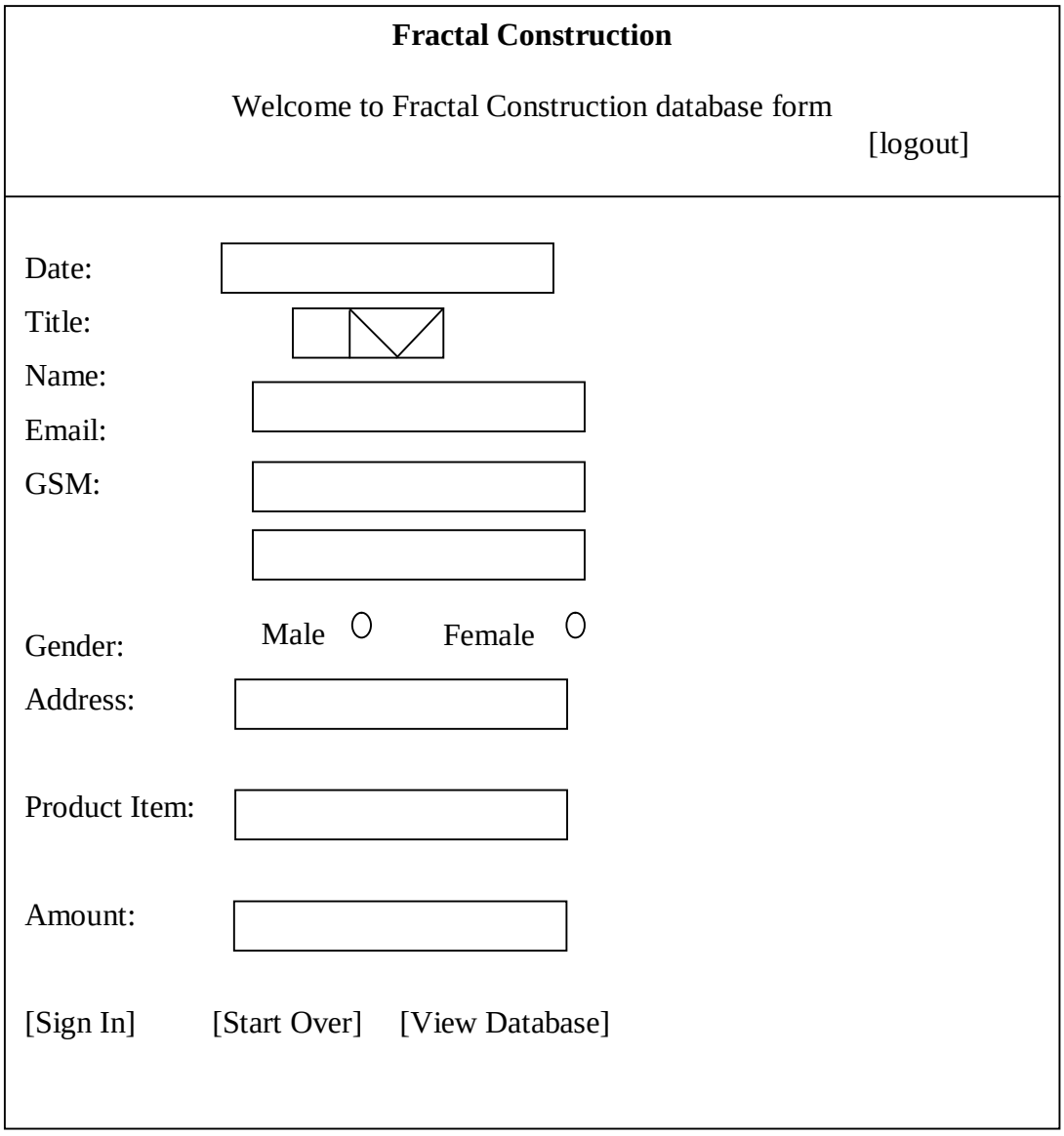

Fig. 2: An Interface Design of Data Input for the Proposed System

Fig. 2: An Interface design of Data Input for the Proposed System

Report of Data Validated for Fractal Construction Limited

S/N	Date	Title	Name	Email	GSM	Gender	Address	Product	Amount
---	-----	----	-------	-------	-----	---------	---------	--------	---------
---	-----	----	-------	-------	-----	---------	---------	--------	---------

GO BACK

Fig. 3: An Interface design of Report of data validated

VII. *RESULTS*

The results show the output of the program after execution .The figures 4-9 below show sample output:

Fig. 4: Home Page of the Data validation System

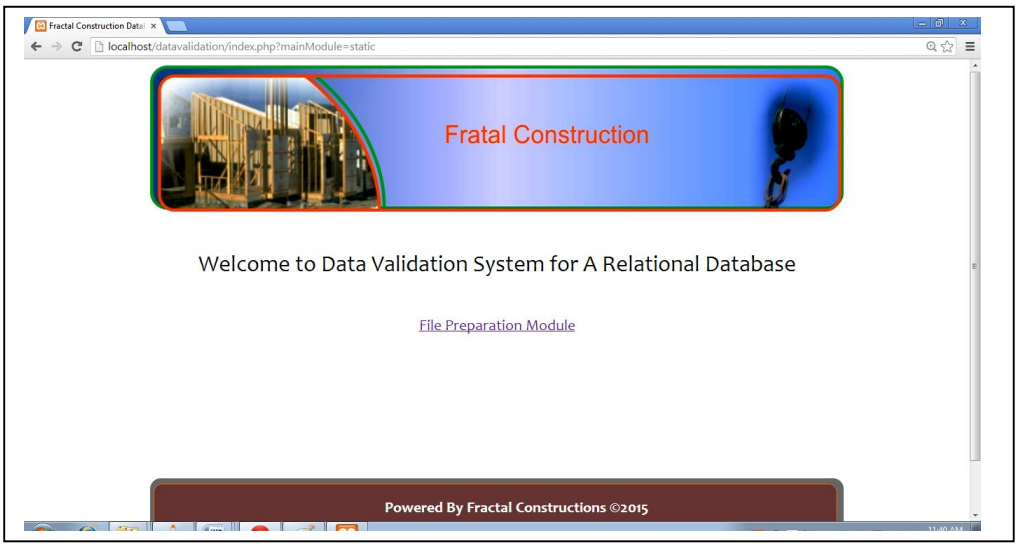

Fig. 5: A Welcome Page with File Preparation Module

Fig. 6: Title and Gender Enhanced Data Validation

Fig. 7: GSM Number Enhanced Data Validation checking for Network type

Fig. 8: Data Validation was successfully done

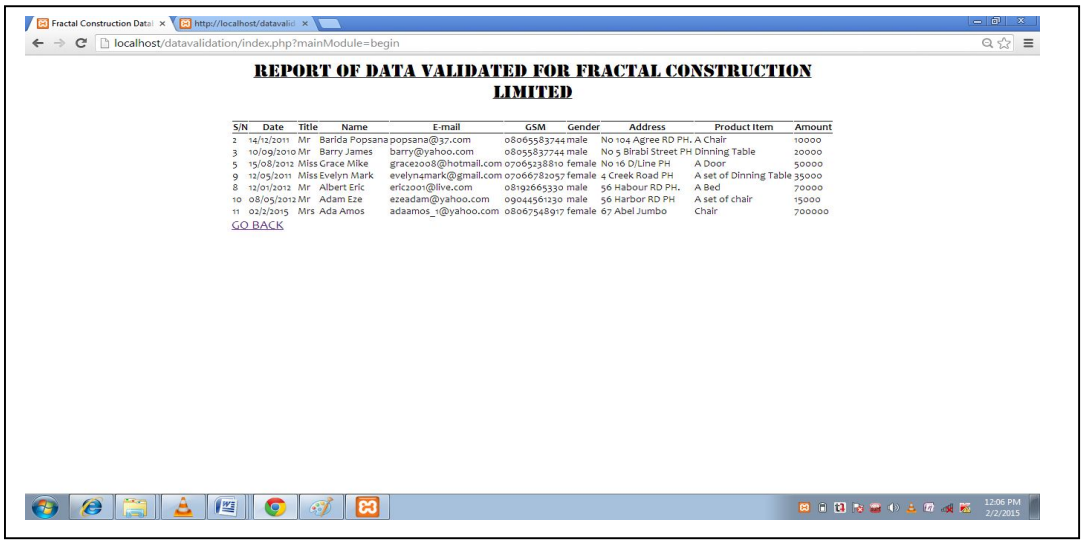

Fig. 9: Sample of Report Generated after successful enhanced data validation

VIII. *CONCLUSION*

Conclusively, there is the need to validate data for any information management system such as the one designed for Fractal Construction Nigeria since it ensures that data inputs are valid, sensible, reasonable and are also secure before they are been processed, all these in turn are useful in good decision making process and also will enhance the productivity of the company also.

IX. *RECOMMENDATION*

The data validation system that is implored here in this research work is purely the client side of validation which is at the end-user input. Future research could focus on validation at the database level. It is recommended that the local machine used to development web application should have the same general features and capabilities of the server(i.e. the remote machine) on which the final solution will finally be deployed. However, if the fractal construction limited that uses the system plans to outsource the maintenance of the application to another person or company, then proper considerations for mimicking the setup of the development server on the staging server can save a great amount of time.

References

[1] M. Arkady, “Data Quality Assessment”, Technics Publication, LLC pp.1-2. 2007.
[2]. P. L. Hans, “Computer for Verifying Numbers”, U.S. Patent 2,950,048. pp. 1-7, 1960.
[3]. J. Verhoeff, “Error Detecting Decimal Codes”, Mathematical Centre Tract 29, The Mathematical Centre, Amsterdam. pp. 2-6, 1969
[4]. D. Scott, and R. Sharp, “Specifying and enforcing application-level web security policies”, IEEE knowledge Data Engineering, vol. 15, no. 4, pp. 771-783, 2003.
[5]. M. Hassinen and P. Mussalo, “Client controlled security for web applications”, in Proceedings of the IEEE Conference on Local Computer on World Wide Web, New York, NY, USA, ACM Press pp. 215-224, 2005.
[6]. Y. Huang, S. Huang, T. Lin, and Tsai, “Web application security assessment by fault injection and behavior monitoring”, in Proceedings of the 12th International Conference on World Wide Web, New York, NY,USA, ACM Press, pp. 148-149, 2003.
[7]. N. Jovanovic, C. Kruegel and E. P. Kirda, “A Static Analysis Tool for Detecting Web Application Vulnerabilities(Short Paper)”, in Proceedings of the 2006 IEEE symposium on Security and Privacy, Washington, DC, IEEE Computer Society, pp. 258-263, 2006.
[8]. D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, and G. S. Vigna, “Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications”, in Proceedings of the 2008 IEEE Symposium on Security and Privacy, Washington, DC, IEEE Computer Society, pp. 387-401, 2008.
[9]. G. Mike and R. Karel, “History of SSADM, SSADM an Introduction” pp. 2-5, 1999.