Improving Data Integrity within Relational Database Using Distinct Function

Eugene S. Valeriano and Rosanna A. Esquivel

***Abstract*—Relational Database Management System is being used for more than decades in a variety of applications. Due to the rising issue of duplication of data via entity resolution and record linkage, there are unnecessary data that makes a mess in the relational databases. Migrating of data from different sources and Data Warehousing causing data redundancy. Data Cleaning is one of the challenges to improve the quality of data within the databases. The Restaurant Dataset of (ZAGAT and FODORS) benchmark database was utilized with the goal of facilitating the effects on improving of data, cleaning and providing data quality. A new proposed method to enhanced data cleaning and consistency without compromising record loss was developed.**

***Index Terms*—Data cleaning, data integrity, quality, record linkage.**

I. Introduction

Relational Database Management System have been existing for thirty years as a concrete usage for data management, storage and exploration. It was added that "RDBMS has a decent character supporting ACID (Atomicity, Consistency, Isolation, and Durability) properties of database and by performing the SQL language which has become a common language" for database administrator. Nevertheless, in spite of its effect, RDBMS have failed to meet the scalable modern applications requirements. That is why support to address the issues when it comes to huge number of record arises for new database management systems. Furthermore, NoSQL database systems allow a flexible design, whereas RDBMS requires a strictly defined scheme [1].

Computer systems nowadays are used to store information from which records can be easily transferred and modified. Systematic using of storage and searching of data is a disparaging issue since stored data from these records are sometimes redundant and adds up to storage space. Computing application requirements fundamental is retrieving of records which requires search operation to perform CRUD functions from different data structures that are enormous databases being implemented [2].

The Market increasingly needs graduates with knowledge in data warehousing, databases and statistical competencies in order to meet the basic needs of the industry and to be able to catch up with the latest trends and technologies being used by the different IT Company which could have an implication for the academic curricula [3]. Aside from the education sector, it could also have an implication to manage E-commerce sites, sensors, cameras, and mobile apps which produces a large amount of data with different periodicity. This huge amount of data must be processed and analyze in order to explain business phenomena and to make predictions which could be the basic assumption of Big Data that can learn from the given set of data.

The researcher aims to enhanced the distinct function of SQL and remove record duplications in the relational database and maintain the referential integrity of records. Based from the researchers experience as a Computer Programmer and a Database Administrator in a University, system applications inadequate validation of user input which also mentioned by [4] and forgotten to established a Primary Key and a clustered index for a certain table [5] are the causes data duplication and creates issues on data integrity in the database.

Temporary table query technique will be used to store unique records and later be compared to the original table for choosing the right primary key to retain. If found duplicate records, the method finds the related foreign keys of all related tables and fields, one record will be retained and will update the corrected reference keys to the soon deleted tuples with the corresponding primary keys in the table.

This proposed method motivates the researcher to improve the data integrity of the SQL database by removing duplicate records [5]. However, DBMS's offers default strategies imposing cascading which prescribe by the SQL standard, but still do not have focus on the quality of relationships table. By maintaining the referential integrity, it would be the result for improving the performance of Standard Query Language execution for Select statements especially on joining tables that will lessen comparison of tuples and reducing of storage space and saving memory allocation to the server.

II. Objectives of the Study

The objective of the study is to developed method to improve the data integrity within a relational database using Distinct Function.

Specifically, it aims to;

1. Perform distinct function on each table in the database;
2. Identify tables with duplicate records;
3. Profile the unique records to a Temp Table;
4. Find affected tables and related fields;
5. Apply the correct record and perform deletion, and;
6. Evaluate the improvement of the database cleaning method to Restaurant Datasets (ZAGAT and FODORS)

Manuscript received September 25, 2019; revised October 29, 2019.

Eugene S. Valeriano and Rosanna A. Esquivel are with the Tarlac Agricultural University, Graduate School, Angeles University Foundation, Philippines (e-mail: esvaleriano@tau.edu.ph).

doi: 10.18178/ijke.2019.5.2.122

III. Conceptual Framework

Fig. 1 shows the conceptual framework on the improvement of the data integrity within a relational databases using distinct function. First, the Relational Database Management System database browse all the table within that database to perform distinct function and find out which table has duplicate record. Next, it will profile the unique records to a temporary table so that it can perform comparison to the original table to find records to be deleted and retained. Finally, before deleting records, the framework will make sure that all the affected tables and fields will be updated accordingly until such time the database will be clean and free from unnecessary records and improve the consistency and integrity of data.

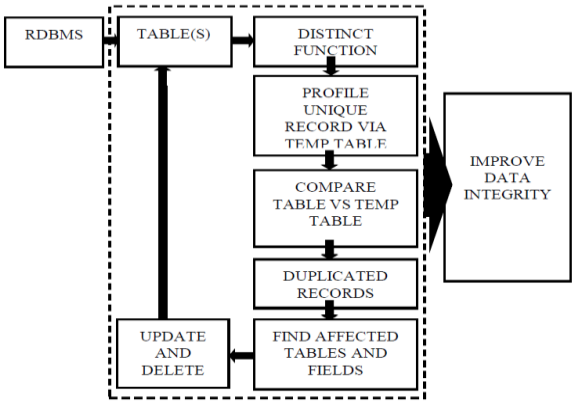

Fig. 1. Conceptual framework of improving data integrity within a relational database using distinct function.

IV. Methodology

A. Diagram for Executing Distinct Function

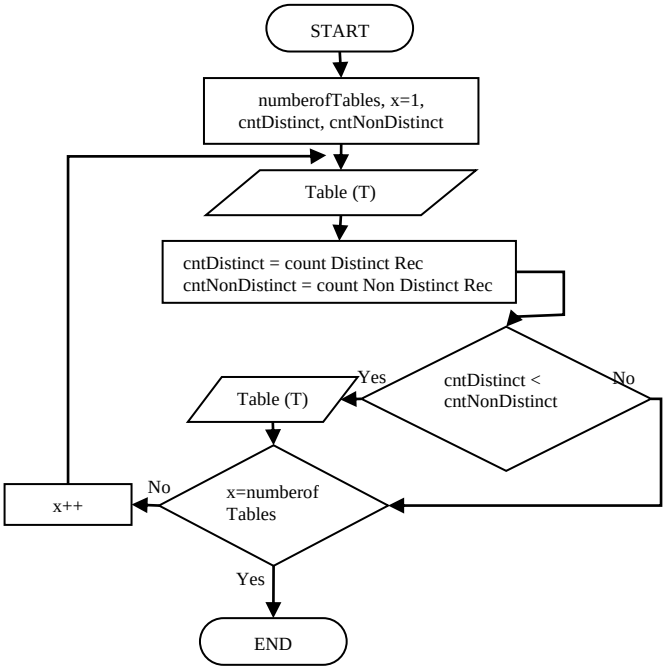

Fig. 2. Flowchart for performing distinct function and identify Table for cleaning in the relational database.

Fig. 2 shows the flow chart for performing distinct function in a database. Once the method started, it will initialize how many tables are there in the database, it represents the variable *numberofTables*, the *x* represents the counter of each loop into the number of tables. *cntDistinct*, represents the number of records perform by the query statements and *cntNonDistinct* represents the number of records of the original table. *(T)* Represents the current table being analyzed. The flowchart has a condition that if the *cntDistinct* is less than the number of records in the *cntNonDistinct* variable, it means that there is a duplicate record in the table. In that case the table will be profiled within a temporary table with unique records, and if *cntDistinct* is equal, the process will move to the next table and repeat the process until all the tables inside the database has been read.

B. Diagram for Selecting Correct Records

Temporary (T)	
Field(s)	Query
A	Q1
B	Q2
C	Q3	

Q	Original (T)		Test	Remarks
	Id	Field(s)		
Q1	1	A	TRUE	Retain
Q1	2	A	FALSE	Remove
Q1	3	A	FALSE	Remove
Q2	4	B	TRUE	Retain
Q3	5	C	TRUE	Retain

Fig. 3. Diagram performing selecting retain records and remove records.

Fig. 3 shows the selection of non-eliminating primary keys of records and removing entire row for data cleaning and improving distinct function in the database. A temporary table contains unique records. This records will be the retained records in the original table. To achieve the selecting of correct records, every record of the temporary table will perform query statements selecting all the records to the original table, the Q1, Q2, Q3 represents the query to be executed. Once the query is executed it will compare the fields associated to it, if the condition is true, the first record will stay and be prioritized, the rest will perform finding of related tables and fields in the database, then will replace all the foreign keys with the correct primary key and afterwards the tuple will be deleted.

The pseudocode below, explains the process on how to find affected tables and fields based on the given unique id.

C. Pseudocode for Finding Affected Related Tables and Fields

- Start the Program
- Identify the retained primary key
- Get the unique id for removing and replace with retained primary
- Loop while unique id finds related tables and fields affected
    - Check if unique id found > update with retained PK
- Repeat until unique id found in the database
- Delete the Record in the main table
- Drop Temporary Table
- End the Program

V. Results and Discussion

A. Dataset

A dataset used for evaluating the data integrity within a

relational database using distinct function is a Restaurant Dataset which came from ZAGAT and FODOR'S company provided by [6].

There are researchers that evaluated their algorithms and methods to eliminate data redundancy in the relational databases [7]-[10].

The composition of combined records of 864 records with the following fields (id, name, address, phone, city and type) according to the provider of the dataset, the records had been marked and identified with a total of 222 duplicates.

B. Set Up

The developed method is working in 1TB of hard Disk, Processor of Intel(R) Core (TM) i7-4510U CPU @ 2.60 GHz, Installed Memory of 8.00 GB, its running Windows Operating System with the MS SQL Server installed.

C. Problem Definition

RESTAURANT

Rid (pk)	*name*	*addr*	*city*	*phone*	*type*	*class*
24	21 club	21 w. 52nd st.	new york	212/582 -7200	american	24
775	21 club	21 w. 52nd st.	new york city	212-582-7200	america (new)	23
95	alain rondelli	126 clement st.	san francisco	415/387 -0408	french	95
846	alain rondelli	126 clement st.	san francisco	415-387-0408	french (new)	94
96	aqua	252 california st.	san francisco	415/956 -9662	seafood	96
847	aqua	252 california st.	san francisco	415-956-9662	american (new)	95
…	…	…	…	…	…	…

A

A

ORDERS

OId(pk)	*RId*	*ProductId*	*Quantity*	*Amount*	*CustomerId*
1	24	1	2	100	1
2	775	2	4	200	1
3	95	3	6	300	2
4	846	4	8	400	2
5	96	5	10	500	3
6	847	6	12	600	3

CUSTOMER

CustomerId	*CustomerName*
1	A
2	B
3	C

Fig. 4. Restaurant database.

Temp Table and Identity Column are the methods proposed [5] to remove redundant records. It was deemed that there are some duplicate records when insert to table of the relational database management SQL Server. The reason is that the Primary Key or cluster-index for the table is not established. In this case, the integrity of data is compromised and it may cause inefficient data result, accuracy and unnecessary input of redundant records which leads to consuming large amount of storage space.

According to [11] many researchers had proposed framework and methods for removing duplicate records within a single relational databases and multiple relations. Moreover, all the methods focused only on the accuracy representation of the records not on the accuracy representation of relationship and quality in the relationship of the tables.

Consider a relational database composed of three relations, the RESTAURANT, ORDERS and CUSTOMER. The relationships of the stated tables as shown above.

Fig. 4 shows that the Restaurant database schema consist of three table relationships. As noticed, there are many duplicate records inside the RESTAURANT table.

Assumed that the Restaurant links to ORDERS table together with the CUSTOMER table.

RESTAURANT

Rid (pk)	*name*	*addr*	*city*	*phone*	*type*	*class*
24	21 club	21 w. 52nd st.	new york	212/582 -7200	american	24
775	21 club	21 w. 52nd st.	new york city	212-582 -7200	american (new)	23
95	alain rondelli	126 clement st.	san francisco	415/387 -0408	french	95
846	alain rondelli	126 clement st.	san francisco	415-387 -0408	french (new)	94
96	aqua	252 californi a st.	san francisco	415/956 -9662	seafood	96
847	aqua	252 californi a st.	san francisco	415-956 -9662	american (new)	95

TEMP_RESTAURANT

Rid (pk)	*name*	*addr*	*city*	*phone*	*type*	*class*
	21 club	21 w. 52nd st.	new york	212/58 2 -7200	america n	24
	alain rondel li	126 clement st.	san francis co	415/38 7 -0408	french	95
	aqua	252 california st.	san francis co	415/95 6 -9662	seafood	96

Fig. 5. Result in performing distinct function.

Fig. 5 shows the performing of distinct function result on RESTAURANT relation, the following collected records based on the distinct query with the id (24, 95, and 96).

TEMP_RESTAURANT						
RId(pk)	***name***	***addr***	***city***	***phone***	***type***	***class***
24	21 club	21 w. 52nd st.	new york	212/582 -7200	american	24
95	alain rondelli	126 clement st.	san francisco	415/387 -0408	french	95
96	aqua	252 california st.	san francisco	415/956 -9662	seafood	96

Fig. 6. Collected correct records.

Fig. 6 shows the correct records to be retained, followed by the diagram stated in Fig. 3, methods to loop through the temp tables and compare it to the original table.

As the method recommends that if the test result of the condition is TRUE for the first time, that record will be the main primary key. If it is TRUE in the next step but still contains duplicate records in the test, it will be removed soon as the method already updated the records in the different relation accordingly.

```
Declare @sql nvarchar(max);
Declare @uniqueId nvarchar(max);
Set @sql = ('SELECT ROW_NUMBER() OVER(ORDER BY
sys.columns.name ASC) AS Row#,
sys.columns.name AS ColumnName,
sys.types.name as DataType, tables.name AS TableName
FROM sys.columns JOIN sys.tables ON
sys.columns.object_id = tables.object_id
join sys.types on
sys.types.user_type_id = sys.columns.user_type_id
where  sys.types.name not in
(''uniqueidentifier'',''image'',''binary'',''bit'',
''ntext'',''datetime'')
order by tables.name,sys.types.name'
);

Declare @T Table(Row# varchar(max),ColumnName
varchar(max), ColumnType varchar(max), TableName
varchar(max))
Insert @T
EXEC(@sql)

Select ' SELECT  '''+ColumnName+''' ColumnName,
['+ColumnName+'] ColumnValue, '''+TableName+'''
TableName FROM [' + TableName +']
where cast( [' + ColumnName + '] as varchar) =
cast(@uniqueId as varchar)  union all'
as r,Row#,ColumnName,ColumnType,TableName from @T
```

Fig. 7. Main source query statements for finding affected tables and fields inside the relational database.

Fig. 7 shows the main source query for finding the affected tables and related fields based on the unique id given. The query has two variables named *@sql*, which represents the methods collecting all the table fields in the database and formulating select query filtering to a specified unique id for every column found in the table. While *@uniqueId*, represents the unique value found in the table that will soon be deleted and updated in the foreign keys with the correct primary keys.

The second group of the query is the creation of the table dynamically, to store formulated statements, row number, column name, column type and table name. And the last group of the query represents the collection of query statements as shown the Fig. 8.

	r	Row#	ColumnName	ColumnType	TableName
1	SELECT 'CustomerId' ColumnName, [CustomerId] Colu...	6	CustomerId	int	CUSTOMER
2	SELECT 'CustomerName' ColumnName, [CustomerNam...	7	CustomerName	nvarchar	CUSTOMER
3	SELECT 'Amount' ColumnName, [Amount] ColumnValue...	2	Amount	float	ORDERS
4	SELECT 'ProductId' ColumnName, [ProductId] ColumnV...	12	ProductId	int	ORDERS
5	SELECT 'Quantity' ColumnName, [Quantity] ColumnValu...	13	Quantity	int	ORDERS
6	SELECT 'RId' ColumnName, [RId] ColumnValue, 'ORD...	14	RId	int	ORDERS
7	SELECT 'OrderId' ColumnName, [OrderId] ColumnValue...	10	OrderId	int	ORDERS
8	SELECT 'CustomerId' ColumnName, [CustomerId] Colu...	5	CustomerId	int	ORDERS
9	SELECT 'class' ColumnName, [class] ColumnValue, 'RE...	4	class	float	RESTAURANT
10	SELECT 'id' ColumnName, [id] ColumnValue, 'RESTAU...	8	id	float	RESTAURANT
11	SELECT 'name' ColumnName, [name] ColumnValue, 'R...	9	name	nvarchar	RESTAURANT
12	SELECT 'addr' ColumnName, [addr] ColumnValue, 'RES...	1	addr	nvarchar	RESTAURANT
13	SELECT 'city' ColumnName, [city] ColumnValue, 'REST...	3	city	nvarchar	RESTAURANT
14	SELECT 'phone' ColumnName, [phone] ColumnValue, '...	11	phone	nvarchar	RESTAURANT
15	SELECT 'type' ColumnName, [type] ColumnValue, 'RES...	15	type	nvarchar	RESTAURANT

Fig. 8. Snapshot of the main query result for finding of affected tables and fields.

```
SELECT  'CustomerId' ColumnName, [CustomerId]
ColumnValue, 'CUSTOMER' TableName FROM [CUSTOMER]
  where cast( [CustomerId] as varchar) =  cast('775'
as varchar)  union all
 SELECT  'CustomerName' ColumnName, [CustomerName]
ColumnValue, 'CUSTOMER' TableName FROM [CUSTOMER]
  where cast( [CustomerName] as varchar) = cast('775'
as varchar)  union all
 SELECT  'Amount' ColumnName, [Amount] ColumnValue,
'ORDERS' TableName FROM [ORDERS]
  where cast( [Amount] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'ProductId' ColumnName, [ProductId]
ColumnValue, 'ORDERS' TableName FROM [ORDERS]
  where cast( [ProductId] as varchar) = cast('775' as
varchar)  union all
 SELECT  'Quantity' ColumnName, [Quantity]
ColumnValue, 'ORDERS' TableName FROM [ORDERS]
  where cast( [Quantity] as varchar) =  cast('775' as
varchar)  union all
 SELECT 'RId' ColumnName, [RId] ColumnValue, 'ORDERS'
TableName FROM [ORDERS]
  where cast( [RId] as varchar) =  cast('775' as
varchar)  union all
 SELECT 'OrderId' ColumnName, [OrderId] ColumnValue,
'ORDERS' TableName FROM [ORDERS]
  where cast( [OrderId] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'CustomerId' ColumnName, [CustomerId]
ColumnValue, 'ORDERS' TableName FROM [ORDERS]
  where cast( [CustomerId] as varchar) =  cast('775'
as varchar)  union all
 SELECT  'class' ColumnName, [class] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [class] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'id' ColumnName, [id] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [id] as varchar) = cast('775' as varchar)
union all
 SELECT  'name' ColumnName, [name] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [name] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'addr' ColumnName, [addr] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [addr] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'city' ColumnName, [city] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [city] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'phone' ColumnName, [phone] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [phone] as varchar) =  cast('775' as
varchar)  union all
 SELECT  'type' ColumnName, [type] ColumnValue,
'RESTAURANT' TableName FROM [RESTAURANT]
  where cast( [type] as varchar) =  cast('775' as
varchar)
```

Fig. 9. SQL statements produce by the main query for finding affected tables and fields.

Fig. 8 represents the script collections of all the tables and fields inside the database. The scripts will produce a series of select statements with each columns comparing it to the intended primary key soon to be deleted. The query result will produce the SQL statements as shown below.

Fig. 9 shows the list of SQL statements to be executed to finally find what the specific tables and fields affected based on the uniqueId to be searched. The value *775* is the *@uniqueId* to be search which causes a duplicate tuples in the *Restaurant* table and soon to be removed as soon as all affected tables and fields into it will be updated to the corrected primary key *24*.

The scripts will execute until all selected primary key to be deleted will be updated to the corrected primary key. The snap result based in the above statements is shown in Fig. 10.

	ColumnName	ColumnValue	TableName
1	RId	775	ORDERS
2	id	775	RESTAURANT

Fig. 10. Snapshot of the query result produced by finding affected tables and fields in the database.

Fig. 10 shows the snap figure of the final result on what tables and fields to be updated. The results contains the *ColumnName*, *ColumnValue* and *TableName*. The *@uniqueId* is *775* were given, the proposed method is successfully find where *775* is exactly located in a specific column names and table names and ready for execution to update corrected primary key *24*.

D. Results

1) Updated restaurant database

RESTAURANT

RId(PK)	*name*	*addr*	*city*	*phone*	*type*	*class*
24	21 club	21 w. 52nd st.	new york	212/582 -7200	american	24
95	alain rondelli	126 clement st.	san francisco	415/387 -0408	french	95
96	aqua	252 california st.	san francisco	415/956 -9662	seafood	96
…	…	…	…	…	…	…

ORDERS

OId(PK)	*RId*	*ProductId*	*Quantity*	*Amount*	*CustomerId*
1	24	1	2	100	1
2	24	2	4	200	1
3	95	3	6	300	2
4	95	4	8	400	2
5	96	5	10	500	3
6	96	6	12	600	3
…	…	…	….	…	…

CUSTOMER

CustomerId	*CustomerName*
1	A
2	B
3	C

Fig. 11. Updated restaurant database.

Fig. 11 shows the updated restaurant database with the application of the proposed method to improve data integrity within a relational database. In the relation shown in RESTAURANT, the unique records which undergo data cleaning process and the table ORDERS updated the corrected primary keys (24, 95, 96). The updated Restaurant Database is free from referential integrity issues and able to removed duplicate tuples found by the SQL distinct function.

2) Summary

TABLE I: THE RESULT OF THE IMPROVEMENT OF THE DATA INTEGRITY

Table (T)	Deleted Records(s)	Updated Record(s)	Total Record
Restaurant	23	0	864
Orders	0	46	500
Customer	0	0	100

Table I conveys that out of 864 restaurant records from ZAGAT and FODOR'S company dataset, the SQL distinct function only detected 841 unique records, it seems that this SQL function is quite low in detecting duplicate records in the relational database, the researcher was able to remove 23 records or 2.7% of the records and able to apply referential integrity, consistency in the ORDERS table with 46 records updated accordingly without losing data. In the CUSTOMER table does not have any detected duplicate records, therefore no records had been updated and deleted.

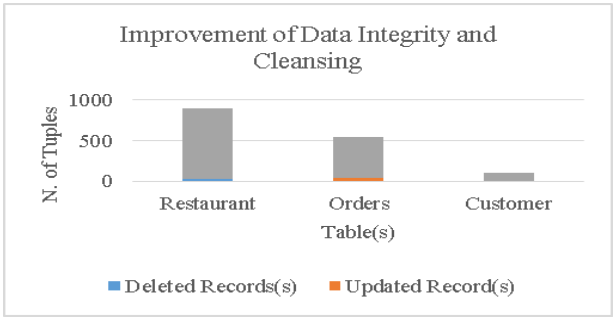

Fig. 12. Representation of the improvement of data integrity and cleaning in SQL database.

Fig. 12, shows the result of the improvement of referential integrity and cleaning method in relation to the data integrity.

As the result of the representation chart, it clearly shows that the more records the database has, the more tuples can be duplicated, can have more referential integrity issues and discrepancy of table affected that causes inefficient data integrity in the relational database. By the use of the proposed method as shown in the result, it can clean duplicate records and retain the correct tuple and update the reference key in the different table affected to the relational database.

VI. CONCLUSION

The major challenge for this study is to use distinct function, find duplicate records, retain one correct tuples, find affected tables and update with the correct primary key before deleting the duplicate tuples in the relational database. With the help of finding duplicate records using distinct function and by using the power of temporary table technique to separate unique tuples in the table, the researcher was able to clean unnecessary tuples and improve the data integrity of

SQL database without losing referential integrity.

Distinct function is limited in finding duplicate records. It cannot detect same words consist of upper case and lower case. It was not able to identify duplicated words that are abbreviated and non- abbreviated. Misspelled words and correct spelled words cannot also be identified as duplicate words. In relations to this limitations and issues of distinct functions, in the future work, a record duplication detections algorithm and knowledge-based algorithm that could be incorporated into this study to improve the method for more efficient data cleaning of duplicate tuples in SQL database, referential integrity which is a result to the improvement of data integrity within relational databases.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

A, B planned the research topic, the main conceptual ideas and proof outline. A, B developed the formulation of decomposition technique in the propose method used and apply the nature inspired crow search algorithm. A, B also verified the method used. B supervised the findings of the study. A, B carried out the different experiments and test cases. A, wrote the manuscript in consultation with B. All authors discussed the results and commented on the manuscript. Finally, all authors had approved the final version of the study.

ACKNOWLEDGMENT

The researcher would like to express his gratitude to all the people who in one way or another contributed to the success of the study. To the CHED K-12 Program for making the researchers goal into reality. To the researcher's adviser and members who shared their expertise that leads to the refinement of the study. To the researcher's colleagues for the support, to the researcher's family for their undying motivation to work with this study and above all to our Almighty God for giving the researcher the knowledge and wisdom to finish the study.

REFERENCES

[1] Z. Bousalem, I. Cherti, and G. Zhao, "Migration from relational databases to HBase: A feasibility assessment," *Lect. Notes Networks Syst.*, vol. 25, pp. 383-395, 2018.

[2] V. P. Parmar and C. Kumbharana, "Comparing linear search and binary search algorithms to search an element from a linear list implemented through static array, dynamic array and linked list," *Int. J. Comput. Appl.*, vol. 121, no. 3, pp. 13-17, 2015.

[3] M. Fotache and C. Strimbei, "SQL and data analysis. some implications for data analysits and higher education," *Procedia Econ. Financ.*, vol. 20, no. 15, pp. 243-251, 2015.

[4] B. Calabrese, "Data cleaning," *Encycl. Bioinforma. Comput. Biol.*, pp. 472-476, 2019.

[5] L. Yue, "The comparison of storage space of the two methods used in SQL server to remove duplicate records," *Procedia Comput. Sci.*, vol. 131, pp. 691-698, 2018.

[6] Sheila Tejada, *Restaurant Dataset.*

[7] S. Tejada, C. A. Knoblock, and S. Minton, "Learning object identification rules for information integration name street phone name street," vol. 26, no. 8, pp. 607-633, 2001.

[8] C. E. and C. F. T. D. Camacho, *A Hybrid Algorithm for Matching Arabic Names Tarek el-Shishtawy Benha University*, Faculty of Computers and Informatics, , pp. 1-15, 2003.

[9] M. Rehman, *Duplicate Record Detection for Database Cleansing*, 2009.

[10] A. Skandar, M. Rehman, and M. Anjum, "An efficient duplication record detection algorithm for data cleansing," *Int. J. Comput. Appl.*, vol. 127, no. 6, pp. 28-37, 2015.

[11] S. A. Babu, *Duplicate Record Detection and Replacement Within A Relational Database*, vol. 10, no. 6, pp. 1893-1901, 2017.

Eugene S. Valeriano is a senior computer programmer and instructor in Tarlac Agricultural University. He obtained his B.S. degree in information technology in March 2008, from Tarlac College of Agriculture and his M.S. degree in information technology in June 2016, from Tarlac State University, Tarlac, Tarlac. He is currently taking his doctorate degree at Angeles University Foundation.

His research interests in the areas of information systems, database systems, and query optimization.

Rosanna A. Esquivel is currently the B.S. information technology program chair of the College of Computer Studies, Angeles University Foundation. She earned her master's degree in computer science from De La Salle University, Manila under a Faculty Development Scholarship program. Dr. Esquivel finished her doctorate degree in information technology from Angeles University Foundation where she was a University scholar. Her research interests include cloud computing, databases, human-computer interaction and algorithms and complexity. She is currently a professor in the Graduate School of Angeles University Foundation teaching graduate level courses and serves as dissertation adviser to graduate students.