

PROGRAMMING TRIGGERS FOR CRITICAL DATA SURVEILLANCE

Monday Eze^{1*}, Charles Okunbor², Ruth Amanze³, Deborah Aleburu⁴

^{1,3} Department of Computer Science, Babcock University, Ogun State, Nigeria

² Department of Computer Science, Admiralty University, Delta State, Nigeria

⁴ Department of Computer Science, Caleb University, Imota, Lagos State, Nigeria

* ezem@babcock.edu.ng

Abstract - The ability of a Database Management System (DBMS) to detect fraudulent activities at the earliest possible time cannot be overemphasized. The concept of critical data surveillance is the ability of system to automatically watch over some selected database objects, and take cognizance of ensuing events. This work achieves this solution through the implementation of database triggers, which keeps track of the major database events. The focus of this work is on surveillance and protection against updates to critical database tables. The work implements real life audit-based surveillance of automated product sales enterprises database tables. The trigger was created from the scratch, and programmed to monitor the critical database object - the product pricing table, which keeps record of the price tagged on each the products in the supermarket. Thus, this work ensures that the trigger is fired accordingly, and that changes to the contents are reported to an audit table for further review by only authorized persons. This research was implemented using PostgreSQL.

Keywords: DBMS; Database Trigger; System Workflow; Audit Table; Multi-Event; Evaluation.

1. Introduction

A database trigger is defined as a function that is invoked (triggered) automatically when a database event occurs on a database object [1]. One of such database objects is a table. The action verb that defines the automatic invocation of a trigger is 'fire'. Thus, a trigger gets fired [2] the moment a target event takes place on a target object. A number of database events could be tracked by a trigger, some of which are Updates, Deletes, and Inserts. A trigger could be programmed to track just one or a number of such events. When it tracks a number of such events, the trigger is said to be a multiple event trigger [3]. A trigger or group of triggers programmed to keep surveillance on database tables will automatically cease to exist the moment the associated database objects get dropped [4]. Two operators determine the areas of influence of a trigger. A trigger marked with the FOR EACH ROW operator will be fired once for each row modified by the operation, while a trigger defined as FOR EACH STATEMENT gets fired once for a specific operation [5]. A number of operations can be performed on a trigger. For instance, triggers could be deleted from a table using the DROP TRIGGER statement [6]. Furthermore, a trigger could be renamed using the ALTER TRIGGER statement [7], disabled using the ALTER TABLE DISABLE TRIGGER statement, and enabled using the ALTER TABLE ENABLE TRIGGER statement [8].

2. Tools and Industrial Case Study

This section presents the technological tools and real life industrial application of this research. A number of modern database management systems (DBMS) have the capacity for data surveillance and protection through the implementation of triggers. Some of the DBMSs are MySQL [9], Oracle [10], PostgreSQL [11], and SQLite [12] among others. This work was implemented using PostgreSQL Ver. 12.0, an object-relational database management system [13] which offers multiple trigger programmability. The key deliverable of this research is therefore to implement a practical and replicable case of critical database surveillance and protection, by programming of a trigger from the scratch. This research is tailored for application in a typical automated Super Store. In a typical Super Store, one of the common but critical database tables is the Product Price Configuration Table (PPCT). The PPCT is a very critical table that houses all the products on display for sale in the different shelves, as well as their configuration information. As shown in Fig. 1, some of product configuration information of interest incorporated in this work are: Product ID, Product Price and Product Name, and some others included as part of the post-trigger audit trail buildup.

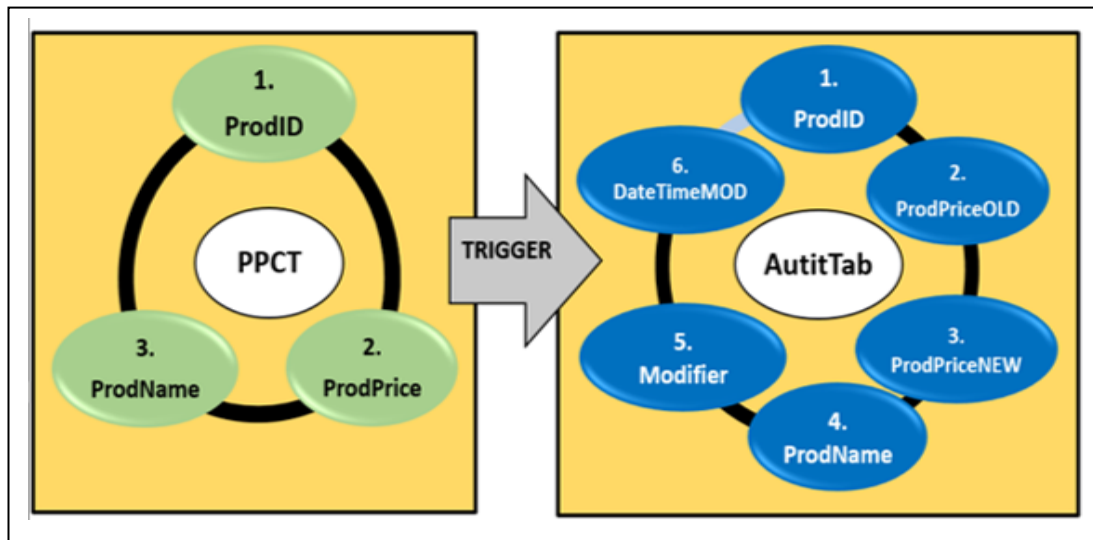


Fig. 1. Structure of Product Price Configuration Table (PPCT)

The importance of the PPCT cannot be overemphasized, and thus the need for this research on its surveillance and protection using database triggers. The trigger could be programmed to keep track of update, deletion, and insertion (UDI) activities [14] in the PPCT, though this work focuses on tracking update events only. On firing, the trigger updates the audit trail table (AuditTab) with the contents of the configuration table. Apart from the fields inherited from the PPCT, three other important information are also added to the audit trail [15]. These are ProdPriceNew which keeps track of the new product price after a modification, Modifier which keeps track of the user who did the modification, and DateTimeMOD which keeps track of the exact time and date when the modification took place. Since the daily sales in the Super Store uses the PPCT to determine the product prices, it is therefore of utmost importance to secure the table. Without taking such a step, huge fraud could be perpetrated through unauthorized changes [16] to product prices, thus the utmost necessity for this research.

3. General System Workflow

This section presents a general workflow [17] for this research. There are four major stages of the system implementation, each of which has been labelled in a chronological fashion 1 to 4 as shown in the workflow diagram in Fig. 2. Sub Section 1 deals with the system requirements [18] for this work, 2 deals with creating the necessary objects, 3 deals with building the trigger logic, while 4 deals with testing the resulting system.

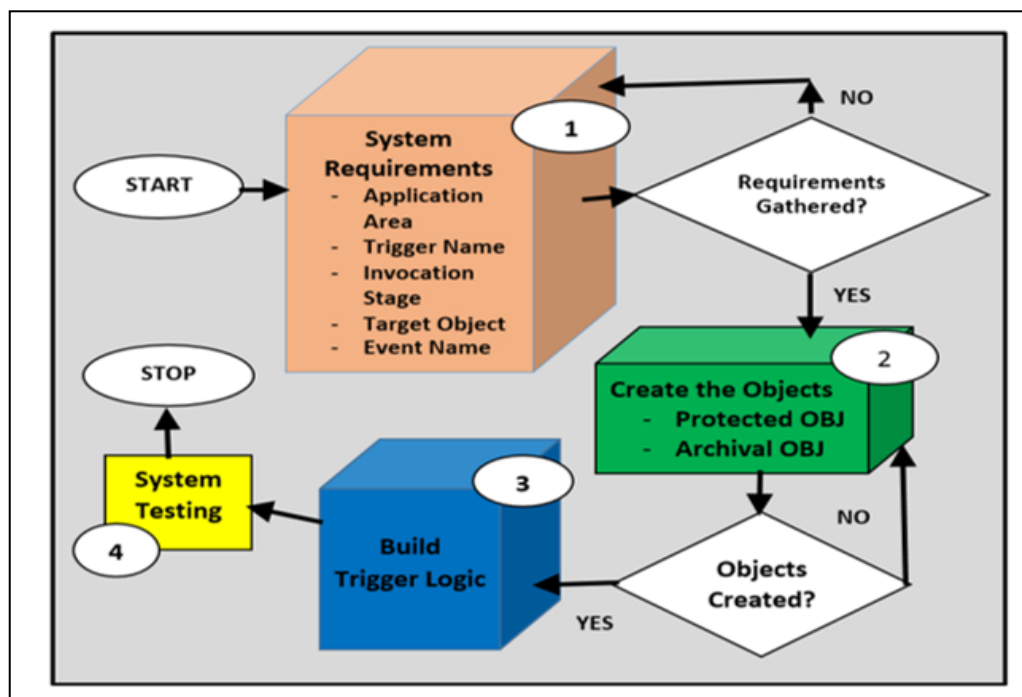


Fig. 2. General System Workflow

3.1. System Requirements

A number of requirements are to be considered in the process of programmatically implementing a trigger for data surveillance and protection. First and foremost is to make a decision on which industry case the trigger-based system is being applied. As earlier stated, this research focuses on the application of triggers in enterprise data [19]. Specifically, this work focuses on the protection of Product Price Configuration Table (PPCT) of a typical Super Store. It is important to mention that the technique demonstrated in this work could be used in similar ways in diverse industrial sectors. For instance, triggers could be applied in securing database tables that relate to academic records in higher institutions [20], account transactions in banking [21], investigation processes in military intelligence [22], patient data in hospitals [23], among others. The next two important requirements at the design stage of the trigger-based system are the name of the trigger and the invocation stage. The trigger invocation stages are determined by three standard keywords BEFORE, AFTER and INSTEAD OF [24]. The fourth requirement is the target (or protected) object, which is the exact database object being protected by the trigger. In this research, the target object is the Product Price Configuration Table of a Super Store. Finally, the fifth requirement is the event name. The major events that could lead to the firing of the trigger are the UDI activities. These are UPDATE, DELETE and INSERT activities in the database. The focus of this research is on surveillance against UPDATE event. Thus, the trigger construction syntax for this research is shown in Fig.3.

```
CREATE TRIGGER trigger-name AFTER UPDATE ON column-name  
ON table-name  
  
[  
-----Trigger Logic  
];
```

Fig. 3. Trigger Construction Syntax

3.2. Objects Creation

The two objects that should pre-exist or should be created before the trigger are the Product Price Configuration Table and the audit table. The structures of the two tables have already been shown in Fig. 2. The CREATE TABLE SQL commands are used to create the tables PPConfigTable and PPAuditTable respectively as shown in Fig. 4 and Fig. 5 respectively.

```
postgres=#  
postgres=# create table PPConfigTable (  
postgres(#      ProdID int PRIMARY KEY,  
postgres(#      ProdPrice real,  
postgres(#      ProdName varchar(100)  
postgres(# );  
CREATE TABLE  
postgres=#  
postgres=# insert into PPConfigTable values (1, 600, 'Tin Tomato');  
INSERT 0 1  
postgres=# insert into PPConfigTable values (2, 1700, 'Bournvita Size 20');  
INSERT 0 1  
postgres=# insert into PPConfigTable values (3, 2130, 'Dangote Red Noodle');  
INSERT 0 1  
postgres=# insert into PPConfigTable values (4, 2130, 'Oriental Match Box');  
INSERT 0 1  
postgres=#
```

Fig. 4. Creation and Initial Population of PPConfigTable

In preparation for a real life demonstration of trigger operations, a total of four product price data ProdID = {1,2,3,4} were inserted into the configuration table PPConfigTable, while an SQL “Select * Statement” [25] was also issued on the PPAuditTable to confirm that it has zero content.

```
postgres=#
postgres=#
postgres=# create table PPAuditTable (
postgres(#      ProdID  int not null,
postgres(#      ProdPriceOLD  real not null,
postgres(#      ProdPriceNEW  real not null,
postgres(#      ProdName  varchar(100) not null,
postgres(#      Modifier  varchar(20) not null,
postgres(#      DateTimeMOD   varchar(50) not null
postgres(# );
CREATE TABLE
postgres=# select * from PPAuditTable;
 prodid | prodpriceold | prodpricenew | prodname | modifier | datetimemod
-----+-----+-----+-----+-----+-----
(0 rows)

postgres=#
```

**NOT NULL
Constraints**

Fig. 5. Creation and Initial Inquiry on PPAuditTable

3.3. Building the Trigger Logic

The choice of trigger logic used in this research aims at reducing system complexity to barest minimum. This method is based on the creation and execution of a stored function. A stored function is defined as a special variant of stored program that returns a single value. Stored functions are used to encapsulate business rules that are reusable among SQL statements [26]. The pictorial presentation of the systematic handshake [27] between the trigger and the stored function is shown in Fig. 6.

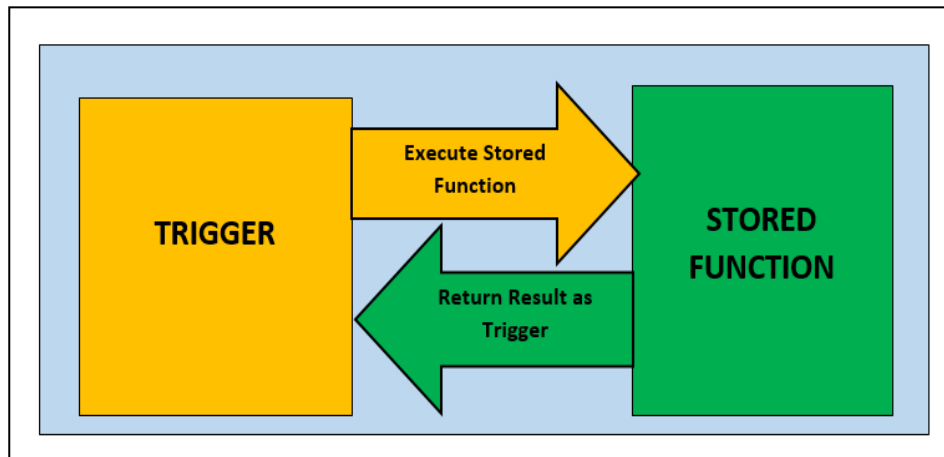


Fig. 6. Handshake between Trigger and Stored Function

The source codes for the stored function is shown in Fig. 7. As shown, the code creates a stored function known as AuditAlarmFunc() which returns a trigger designated as a system variable \$my_table\$. The program actions are subsumed in between the BEGIN .. END keywords [28]. When the trigger is fired, it inserts the six audit values ProdID, ProdPriceOLD, ProdPriceNEW, ProdName, Modifier, and DateTimeMOD. Finally, the last statement is an invocation of the system variable \$my_table\$, then an indication of the programming language used for building the stored function, which in this case, is “PLPGSQL”.

```
CREATE OR REPLACE FUNCTION AuditAlarmFunc()
RETURNS TRIGGER AS $my_table$
BEGIN
    INSERT INTO PPAuditTable(
        ProdID, ProdPriceOLD,
        ProdPriceNEW, ProdName,
        Modifier, DateTimeMOD)
    VALUES
        (old.ProdID, old.ProdPrice,
        new.ProdPrice, old.ProdName,
        user, current_timestamp);
    RETURN NEW;
END;
$my_table$ LANGUAGE 'plpgsql' ;
```

Fig. 7. Source Code for AuditAlarmFunc Stored Function

The source code for creating the trigger is shown in Fig. 8. As shown, the trigger name is price_trigger, and is fired after every update of the PPConfigTable. The trigger executes the AuditAlarmFunc() stored function every moment a row of the configuration table is modified. This is achieved using the “FOR EACH ROW” statement [29]. Apart from the row-by-row activation, triggers could also be activated for every designated operation. This is achieved using the “FOR EACH STATEMENT” statement [30].

```
postgres=#
postgres=# CREATE TRIGGER price_trigger
postgres=# AFTER UPDATE ON PPConfigTable
postgres=# FOR EACH ROW EXECUTE PROCEDURE AuditAlarmFunc();

CREATE TRIGGER
postgres=#
```

Fig. 8. Programming Code for price_trigger

It is important to watch out for a system response that confirms if the creation of a trigger was successful or otherwise. The system response CREATE TRIGGER [31] is an indication that the trigger was successfully created, as shown in Fig. 8.

4. Trigger Evaluation

The evaluation [32] of the trigger involved two major steps. First, the pre-evaluation contents of the configuration and audit tables are checked. This is achieved by issuing SELECT statements on PPConfigTable and PPAuditTable respectively. As shown in Fig 9, the audit table has a zero content, while the configuration table has a total of four records.

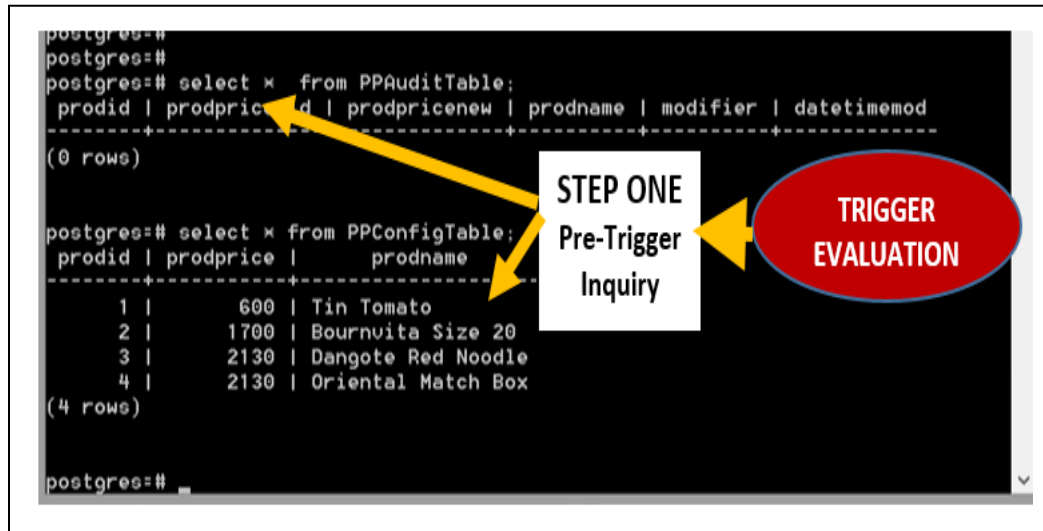


Fig. 9. Pre-Trigger Inquiry Screen 1

The second stage of the evaluation is to perform an SQL update on the product price table. As shown in Fig. 10, the product price for Product ID number 4 was changed from the original value of 2130 to a new value of 4000. This action is expected to fire the trigger, thus the need for the next evaluation step, which is the post-trigger inquiry.

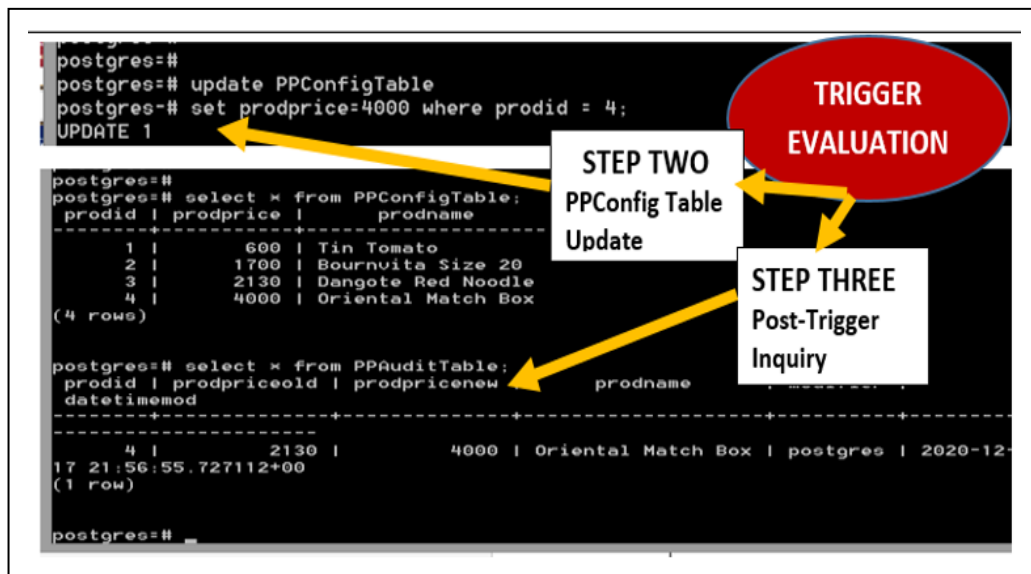


Fig. 10. Trigger Evaluation Screen 2

As shown in the final evaluation screen, the exact record that was updated in the PPConfigTable was successfully inserted into the PPAuditTable after the trigger was fired. This is a confirmation that the trigger is working as expected.





5. Conclusion

This work has demonstrated in practical terms the evolution, design and implementation of triggers from the scratch. It was also applied on a practical case, being the surveillance and audit-based protection of critical enterprise database tables, such as the product price configuration table (PPCT) in a typical Super Store. The evaluation of system was successfully demonstrated. It is hoped that the trigger-based system demonstrated in this work can be replicated in a number of industrial concerns with minor modifications.

References

- [1] Wang, M. (2011). Solving Relational Database Problems with ORDBMS in an Advanced Database Course. *Information Systems Education Journal*. Vol 9, No. 4, 80-90.
- [2] Parsana, F. (2016). Trigger for Monitoring, Tracking and Database Security – From Theory to Practice. *Int. Journal of Innovative Research in Computer and Communication Eng.* Vol 4, Issue 10, 17186-17191.
- [3] Liu, S., Li, Y., Zhou, X., Yang, T. and Zhang, F. (2019). Event Detection without Triggers. In *Proceedings of NAACL-HLT*. 2019, Minnesota, June 2-7, pp 735-744.
- [4] Itai, Y., Oludele, A. and Goga, N. (2013). Trigger and Database Security. *International Journal of Computers and Technology*. Vol 4, No. 1, Feb 2013. Pp 57-62
- [5] Matulevicius, R. and Lakk, H. (2015). A Model-driven Role-based Access Control for SQL Databases. *Complex Systems Informatics and Modeling Quarterly*. Issue3, pp 35-62.
- [6] Jangra, A. Bishla, D., Komal, B. and Priyanka, B. (2010). Functionality and Security Analysis of Oracle, IMB-DB2 and SQL Server. *Global Journal of Computer Science and Technology*. Vol 10. Issue 7 pp8-12
- [7] IBM (2013). Database SQL Programming. IBM i Version 7.2 Documentation. IBM, USA.
- [8] Mok, W., Hickman, C. and Allport, C. (2013). Implementing Business Processes: A Database Trigger Approach. *International Journal of Knowledge-Based Organizations*. Vol. 3, Issue 2, pp36-55
- [9] Bell, C. (2016). MySQL for the Internet of Things. APress, New York.
- [10] Feuerstein, S. and Pribyl, B. (2009). Oracle PL/SQL Programming, 5th Edition, Published by O'Reilly Media, Inc. Sebastopol.
- [11] Juba, S. and Volkov, A. (2017). Learning PostgreSQL 10: A beginner's guide to building high-performance PostgreSQL database solutions, Second Edition, Published by Packt Publishing Ltd. , Birmingham
- [12] Kengalagutti, D. (2020). Comparing Database Management Systems: MySQL, PostgreSQL, SQLite. *International Research Journal of Engineering and Technology*. Vol. 7, Issue 6, 2238-2241.
- [13] Velicanu, M. and Botha, I. (2011). Solutions for the Object – Relational Databases Design. *Database Systems Journal*, Vol 2, No 4, p51-64
- [14] Bhardwaj, A. and Sharma, K. (2015). Types of Queries in Database System. *International Journal of Computer Applications in Technology*. Vol. 7, No. 2, 149-152
- [15] Carcary, M. (2009). The Research Audit Trial – Enhancing Trustworthiness in Qualitative Inquiry.” *The Electronic Journal of Business Research Methods* Vol. 7 Issue 1 pp.11 – 24.
- [16] Ahmad, K. (2014). Data Prevention from Unauthorized Access by Unclassified Attack in Data Warehouse. In the *Proceedings of IEEE 2014 International Conference on “Computing for Sustainable Global Development”*, 5 March, 2014 Bharati Vidyapeeth's Institute, New Delhi, pp846-850
- [17] Reijnders, H. and Van der Aalst, W. (2005). The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned. *International Journal of Information Management*. Vol. 25, No. 5, pp 458-472.
- [18] Mighetti, J. and Hadad, G (2016). A Requirements Engineering Process Adapted to Global Software Development. *CLEI Electronic Journal*, Vol. 19, No. 3, pp 1-21
- [19] Morento, C., Carrasco, R. and Herrera-Viedma, E. (2019). Data and Artificial Intelligence Strategy: A Conceptual Enterprise Big Data Cloud Architecture to Enable Market-Oriented Organizations. *Special Issue on Use-Case of Artificial Intelligence, Digital Marketing and Neuro Science*. pp7-14.
- [20] Utami, E. and Raharjo, S. (2014). Database Security Model in the Academic Information System *International Journal of Security and its Applications* 8(3):163-174
- [21] Haque, A., Kumar, A., Sabbir, M. and Raquib, A. (2009). Electronic Transaction of Internet Banking and its Perception of Malaysian online Customers. *African Journal of Bus. Management* 3(6):248-259.
- [22] Symon, P. and Tarapore, A. (2015). Defense Intelligence Analysis in the Age of Big Data. *Joint Force Quarterly* 79, National Defense University Press. Washington, D.C.
- [23] Lederman, R. (2005). Managing hospital databases: Can large hospitals really protect patient data?. *Health Informatics Journal* 11(3):201-210.
- [24] Matthew, N. and Stones, R. (2005). *Beginning Databases with PostgreSQL: From Novice to Professional*, Second Edition, Published by APress and Springer-Verlag, New York
- [25] Meiryani, A. (2019). Database Management System. *International Journal of Scientific and Technology Research*. Vol. 1, Issue 6, pp309-312.
- [26] Malik, M. and Patel, T. (2016). Database Security - Attacks and Control Methods. *International Journal of Information Sciences and Techniques*. Vol 6, No. 1. pp175-183
- [27] Bai, Y. (2011). *Practical Database Programming with Java*. John Wiley and Sons Inc. New Jersey.
- [28] Sharma, H., Biswas, R. and Shastri, A. (2011). PL/SQL and Bind Variables: the two ways to increase the efficiency of Network Databases. *Database Systems Journal*. Vol 2, No. 4, pp9-16.
- [29] Abadi, D., Boncz, P., Harizopoulos, S., Idreos, S. and Sadden, S. (2013). The Design and Implementation of Modern Column-Oriented Database Systems. *Functions and Trends in Databases*. Vol. 5, No. 2, 197-280
- [30] Hellerstein, J., Stonbraker, M. and Hamilton, J. (2007). Architecture of a Database System. *Foundation and Trends in Databases*. Vol. 1, No. 2, pp 141-259
- [31] Chauhan, C. (2015). *PostgreSQL Cookbook: Over 90 hands-on recipes to effectively manage, administer, and design solutions using PostgreSQL*. Published by Packt Publishing Ltd, Birmingham B3 2PB, UK
- [32] Valenti, S., Cucchiarelli, A. and Panti, M. (2002). Computer-Based Assessment Systems Evaluation via the ISO 9126 Quality Model. *Journal of Information Technology Education*. pp157-175

Authors Profile

	<p>Monday Eze has a Bachelor's degree (BSc), Master's degree (MSc), and PhD in Computer Science. He also has an MBA in Management, and over a decade of professional experience in the Banking Industry. He is currently an Associate Professor of Computer Science at Babcock University, Nigeria. His research interests are Computational Algorithms and Data Mining. He has supervised a number of postgraduate students. He is a member of Nigerian Computer Society (NCS).</p>
	<p>Charles Okunbor has a Bachelor of Science degree (B.Sc.) in Information and Computer Systems and a Master of Science degree in Information Technology. He is an inquisitive computer science specialist and a researcher. His love for research earned him a Ph.D. in Computer Science from Babcock University, Nigeria. He has vast knowledge and experience in different areas of computing and digital technologies which he has applied to both academics and industries. He is currently an Adjunct lecturer with the Department of Computer Science, Admiralty University of Nigeria.</p>
	<p>Ruth Amanze has a Higher National Diploma (HND) in Electronics and Telecommunications from the Institute of Management and Technology (IMT), Enugu State. She also has a PGD, MSc and PhD in Computer Science from Babcock University, Ogun State. She is currently an Adjunct Lecturer in Computer Science as well as the Assistant Director, Department of Information Technology Development Services (ITDS) in Babcock. Her area of specialization is Networking and Telecommunication.</p>
	<p>Deborah Aleburu obtained her BSc in Computer Science in 2014 at Bowen University. She also obtained her MSc and PhD in Computer Science from Babcock University. Dr. Aleburu has served as a teaching and research assistant in Babcock University, and is currently a Computer Science lecturer at Caleb University, Lagos. Her areas of interest are Network and Information Security, Blockchain Technology, Internet of Things, among others.</p>