TRIGGERS IN ORACLE SQL: AUTOMATION AND INTEGRITY IN ECONOMIC DATABASE MANAGEMENT

Ionel IACOB[1]
Cezar MIHĂLCESCU[2]

Abstract
In the economic sphere, maintaining operational efficacy and decision-making accuracy requires process automation and data consistency. Enforcing business rules, automating time-consuming tasks, and preserving data integrity all depend on Oracle SQL triggers. This article examines the principles and applications of triggers in economic database management systems, with a focus on their use in financial transactions, audit trails, and compliance enforcement. Real-world examples show how triggers enhance database functionality and dependability, underscoring their significance in economic environments. Because the economic sector relies largely on transactional data and has high accuracy and transparency requirements, it is uniquely positioned to benefit from database triggers. Triggers serve as the basis for automating crucial procedures including real-time reporting, tax compliance, and fraud detection. In both banking and retail, triggers serve as the basis for automating critical procedures including tax compliance, fraud detection, and real-time financial reporting. This article explores the principles, architecture, and real-world applications of Oracle SQL triggers to demonstrate how they can maximize database performance while meeting industry requirements.

Keywords: Oracle SQL, database automation, economic systems, triggers, data integrity, financial transactions, compliance, audit trails

JEL Classification: O33, M15

1. Introduction

Data is a vital resource that drives decision-making and operational efficiency across industries in the contemporary economic environment. Organizations in a variety of industries, including banking, insurance, retail, and manufacturing, depend on reliable database management systems (DBMS) to handle enormous volumes of transactional data, implement corporate policies, and guarantee regulatory compliance. The ability to build triggers, which are essential for automating database processes and preserving data

[1] Lecturer Ph. D., Romanian-American University, Romania, ionel.iacob@rau.ro
[2] Professor Ph. D., Romanian-American University, Romania, cezar.mihalcescu@rau.ro, corresponding author

integrity, is one of Oracle SQL's many sophisticated features that set it apart from other DBMS platforms.

Procedural code objects called triggers in Oracle SQL run automatically when certain events on a database table or view are triggered. Operations like INSERT, UPDATE, and DELETE as well as schema-level updates like user logins or privilege changes might be examples of these events.

Triggers are essential in the economic domain, where process automation and data consistency are critical. They simplify processes that are essential to industries like banking, retail, and finance, like tax calculations, transaction validation, and audit trail maintenance. For example, a trigger can prevent unauthorized overdrafts in a banking application, ensuring customer accounts remain accurate and compliant with internal policies. Similarly, in e-commerce platforms, triggers can automatically adjust inventory levels after each transaction, reducing the likelihood of overselling.

Triggers are important for reasons other than operational effectiveness. Triggers enforce data security and integrity in a smooth and consistent way by integrating logic directly into the database layer. This reduces the possibility of human error and guarantees adherence to industry rules, including those controlling tax collection and financial reporting. However, in order to prevent performance snags and guarantee scalability in high-volume economic systems, trigger implementation calls for meticulous planning and optimization.

The concepts, varieties, and uses of triggers in Oracle SQL are examined in this essay, with an emphasis on how they contribute to cost-effective database administration. With the help of real-world examples and case studies, it explains how triggers may improve data integrity, automate important procedures, and ease regulatory compliance.

2. Key Concepts and Types of Triggers

Triggers can be configured to execute either before or after specific events, such as INSERT, UPDATE, or DELETE operations. Oracle SQL supports a wide variety of trigger types, including:

• Triggers at the row level: These triggers operate for each impacted row, making them ideal for scenarios requiring fine-grained management.

• Statement-Level Triggers: These triggers only fire once for the entire statement, regardless of how many rows are impacted.

• Permit DML operations on views instead of triggers, as these are essential for complex financial reporting.

• System triggers respond to schema-level events, such as database logins or privilege changes, and are commonly used for security purposes.

These trigger kinds provide customized solutions to meet certain business needs in economic systems. For example, statement-level triggers can automate batch procedures in financial reporting, whereas row-level triggers are essential for guaranteeing transactional accuracy in banking systems.

3. Applications of Economic Systems

Generating financial transaction automation, economic databases are commonly used to manage repetitive processes, including applying discounts, managing stock portfolios, and calculating interest. Triggers increase efficiency and reduce human error by automating these procedures. For example, a trigger can automatically calculate and apply value-added tax (VAT) to sales transactions to ensure compliance with tax regulations.

As regards the ensuring of data integrity, financial systems must provide data consistency to prevent errors that could lead to significant financial losses. Triggers uphold corporate regulations, such as prohibiting withdrawals that exceed an account's available balance or ensuring that invoices are not deleted without administrative approval.

Another relevant aspect concerns the maintenance of audit trails. In sectors like banking and insurance, regulators demand meticulous record-keeping. The timestamp, the type of change, and the user who did it are all automatically recorded by triggers when sensitive data is modified. This ensures transparency and facilitates compliance audits.

4. Economic database triggers offer multiple benefits

The main advantage is related to increased efficiency and automation. Triggers reduce the amount of manual work required to complete repetitive tasks, such as recalculating account balances or preparing reports. By automating these processes, companies save time and money, allowing employees to focus on higher-value tasks.

At the same time, increased data security and integrity is an important advantage. Triggers act as a safety net, ensuring data accuracy, blocking unwanted changes, and enforcing business rules at the database level. To avoid overselling and customer dissatisfaction, triggers can, for example, check inventory levels in an e-commerce platform before processing orders.

Triggers facilitate compliance to the industry rules, by automating regulatory processes such as maintaining audit trails or calculating legal taxes. This ensures that companies comply with the law without having to put in additional physical effort.

5. Implementation Challenges

Triggers have benefits, but they must be used cautiously to prevent performance problems. Database processes may be slowed down by poorly designed triggers, such as those involving intricate queries. Furthermore, excessive trigger usage can make database systems challenging to maintain and troubleshoot, particularly in settings where schema changes occur often.

6. Scientific Research

The rationale behind the selection of the research applications, automatic stock updates, VAT computation, and overdraft prevention etc. was their economic significance, practical relevance, and ability to demonstrate the effectiveness of database triggers in resolving real-world issues. Every example tackles a basic issue in its field and demonstrates how Oracle SQL's automated features may improve data integrity, operational effectiveness, and regulatory compliance.

6.1. Dealing with Real-World Issues

Accuracy and consistency of data are critical in economic systems. Errors in inventory management, tax compliance, or financial management can result in large financial losses or harm to one's reputation. The selected examples center on situations in which automation using triggers immediately lowers risks, minimizes mistakes, and guarantees that business standards are followed.

6.2. Improving Efficiency in Operations

A key component of contemporary database administration is automation. Triggers make it possible to execute business logic and enforce rules in real time without depending on third-party apps. This database-level integration guarantees consistency across all access points, streamlines processes, and lowers overhead.

For instance:

- The banking example protects clients and organizations from any financial irregularities by dynamically preventing overdrafts.
- Retail inventory triggers guarantee real-time stock level updates, increasing consumer satisfaction and lowering holding costs.

6.3. Guaranteeing Adherence to Regulations

Economic systems are highly regulated, especially when it comes to financial reporting and taxes. The example of VAT calculation shows how database triggers may reliably enforce tax laws, bringing operations into compliance with the law.

6.4. Contribution to Science

These examples demonstrate the flexibility of database triggers as instruments for carrying out intricate business logic from a research standpoint. They are used as case studies to investigate: the combination of declarative and procedural database programming. the effect of real-time automation on accuracy and system performance. Trigger scalability in high-transaction settings.

6.5. Wider Consequences

In addition to proving technical viability, these examples offer guidance on creating reliable systems in industries including banking, retail, and taxation. They open the door for more research into the ways database triggers can help with issues in other economic domains like e-commerce, insurance, and supply chain management.

In summary, the selection of these instances for scientific study stems from their applicability, their capacity to demonstrate the power of Oracle SQL triggers, and their potential to spur additional advancements in database-driven automation for cost-effective uses. These examples demonstrate how Oracle SQL triggers can be used to incorporate business logic directly into the database from a research standpoint. By lowering reliance on application layers, this method improves the scalability, resilience, and effectiveness of systems. The examples also shed light on how triggers may be modified to address similar problems in other fields, like healthcare, logistics, or educational institutions.

7. Practical Examples

7.1. Automating Tax Calculation in a Retail Database

Domain: Retail and Supply Chain Management Use: Accurate inventory control is essential for operational effectiveness in the retail industry. When a sale is registered, the stock update example automatically lowers inventory levels, guaranteeing that the system displays current stock availability. Given that millions of transactions take place every day in e-commerce platforms and huge retail chains, this example is very relevant. Businesses can accomplish the following by integrating stock management logic into the database.

Operational Efficiency: Less manual intervention is required because inventory adjustments happen automatically.

Error Reduction: Avoids underselling or overselling because of erroneous stock information.

Better Customer Experience: Better service is a direct result of real-time stock visibility.

Scenario: A retail company operates in a region where a 20% VAT (Value-Added Tax) must be applied to every sale. To ensure compliance and streamline operations, a trigger is used to calculate and record the VAT for each transaction automatically.

Implementation: Every time a new sales record is added to the sales table, the following trigger automatically determines the VAT amount:

```
CREATE OR REPLACE TRIGGER calculate_vat
BEFORE INSERT ON sales
FOR EACH ROW
BEGIN
    :NEW.vat_amount := :NEW.sale_amount * 0.20;
    :NEW.total_amount := :NEW.sale_amount + :NEW.vat_amount;
END;
/
```

Figure 1. VAT automating calculator

Explanation of the code:

- The trigger is executed before a new row is inserted into the Sales table.
- It calculates the VAT (20% of the sale amount) and updates the vat_amount field.
- The total amount, including VAT, is also computed and updated automatically.

Impact: This trigger ensures all sales transactions include VAT, reducing the risk of non-compliance and eliminating manual calculation errors.

7.2. Maintaining an Audit Trail in Financial Systems

Scenario: A financial institution needs to track all modifications to its Loans table for compliance purposes. A trigger is used to log every insert, update, or delete operation into an Audit_Log table.

```
CREATE OR REPLACE TRIGGER audit_loans
AFTER INSERT OR UPDATE OR DELETE ON loans
FOR EACH ROW
BEGIN
    INSERT INTO audit_log (action_type, user_id, timestamp, old_data, new_data)
    VALUES (
        CASE
            WHEN INSERTING THEN 'INSERT'
            WHEN UPDATING THEN 'UPDATE'
            WHEN DELETING THEN 'DELETE'
        END,
        USER,
        SYSDATE,
        CASE
            WHEN DELETING OR UPDATING THEN :OLD.loan_details
            ELSE NULL
        END,
        CASE
            WHEN INSERTING OR UPDATING THEN :NEW.loan_details
            ELSE NULL
        END
    );
END;
/
```

Figure 2. Implementation ex.

The code is explained as follows:

• The trigger records the timestamp, the user who executed the operation, and the type of operation (INSERT, UPDATE, or DELETE).

• Additionally, it keeps track of both new and old data for any updates or removals.

Impact: This ensures compliance with financial regulations by maintaining a complete record of changes, which is crucial for audits and investigations.

7.3. Banking System - Preventing Overdrafts

Domain: Banking and Financial Systems

Application: This example addresses a common problem in banking—preventing customers from withdrawing more money than they have in their accounts. Overdraft prevention ensures that no transaction results in a negative account balance, protecting both customers and financial institutions.

Triggers play a critical role here by enforcing account balance checks dynamically at the database level. This eliminates reliance on application-layer validations that might vary across platforms (e.g., online banking, mobile apps, ATMs).

The implementation guarantees the data integrity, by ensuring accurate account balances at all times, a better risk management, by reducing financial exposure due to overdraft errors, and enhancing customers' confidence in the banking system.

Theoretical Foundation: Data integrity requirements that are applied at the database level are used to prevent overdrafts. In database theory, triggers serve as a dynamic extension of constraints, which guarantee that data complies with predetermined guidelines. Triggers provide the ability to implement intricate business logic, in contrast to static constraints like NOT NULL or CHECK.

Economic Consequences: Preventing overdrafts guarantees that financial institutions continue to operate with credibility. Particularly in economies that depend on digital banking, mistakes in this area could result in systemic financial concerns. The organization reduces risk exposure and guarantees adherence to internal and regulatory financial controls by incorporating overdraft logic into the database layer.

```
CREATE TABLE accounts (
    accountId NUMBER PRIMARY KEY,
    account_holder VARCHAR2(100),
    balance NUMBER CHECK (balance >= 0)
);

INSERT INTO accounts VALUES (1, 'Andrei Ionescu', 5000);
INSERT INTO accounts VALUES (2, 'Grigore Popescu', 10000);
INSERT INTO accounts VALUES (2, 'Mihai Ilie', 10000);
COMMIT;

CREATE OR REPLACE TRIGGER prevent_overdraft
BEFORE UPDATE OF balance ON accounts
FOR EACH ROW
BEGIN
    IF :NEW.balance < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds: Overdraft not allowed.');
    END IF;
END;
/

UPDATE accounts SET balance = 4000 WHERE account_id = 1;
UPDATE accounts SET balance = -100 WHERE account_id = 1;
```

Figure 3. Implementation ex.

Explanation of the code:

The accounts table structure includes a balance column to represent the account balance.

The BEFORE UPDATE trigger checks if the balance is less than zero before committing the transaction. If the condition is met, the trigger raises an application error.

Outcome: Transactions that would result in a negative balance are blocked, ensuring account integrity.

7.4. Retail System - Automatic Stock Updates

Theoretical Foundation: The event-driven programming paradigm, in which changes in one area of the system cause equivalent updates in other areas, is utilized in this example.

Referential integrity across related tables is crucial in relational database systems, particularly in high-volume transactional settings like retail.

Economic Consequences: Sustaining operational effectiveness and customer happiness depends heavily on accurate stock management. Inventory tracking mistakes can lead to either understocking, which reduces sales and erodes consumer trust, or overstocking, which raises holding costs.

Real-Time Synchronization: Stock level updates are instantly disseminated, guaranteeing that all system parts run on the same information.

Error Prevention: The trigger avoids operational problems like overselling, which could damage a brand's reputation and customer connections, by preventing negative stock levels.

Scalability: The trigger logic ensures data consistency in multi-user scenarios by supporting distributed systems with concurrent transactions.

```
CREATE TABLE inventory (
    product_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(100),
    quantity NUMBER CHECK (quantity >= 0)
);

CREATE TABLE sales (
    sale_id NUMBER PRIMARY KEY,
    product_id NUMBER,
    quantity_sold NUMBER,
    sale_date DATE DEFAULT SYSDATE,
    FOREIGN KEY (product_id) REFERENCES inventory(product_id)
);

INSERT INTO inventory VALUES (1, 'Laptop Lenovo', 50);
INSERT INTO inventory VALUES (2, 'Smartphone Allview', 100);
COMMIT;

CREATE OR REPLACE TRIGGER update_stock
AFTER INSERT ON sales
FOR EACH ROW
BEGIN
    UPDATE inventory
    SET quantity = quantity - :NEW.quantity_sold
    WHERE product_id = :NEW.product_id;
    IF (SELECT quantity FROM inventory WHERE product_id = :NEW.product_id) < 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Stock level cannot be negative.');
    END IF;
END;
/

INSERT INTO sales (sale_id, product_id, quantity_sold) VALUES (1, 1, 5);
INSERT INTO sales (sale_id, product_id, quantity_sold) VALUES (2, 1, 100);
```

Figure 4. Implementation ex.

Explanation of the code:

The inventory table holds product stock levels, and the sales table records transactions.

Trigger Logic: The AFTER INSERT trigger updates the inventory levels immediately after a sale is recorded. If the new inventory quantity drops below zero, the trigger raises an error.

Outcome: Inventory levels are maintained in real time, preventing overselling and ensuring stock integrity.

7.5. Tax Compliance - Automatic VAT Calculation

Domain: Financial Reporting and Tax Compliance Use: In many businesses, calculating taxes is a regulation necessity. In order to ensure consistent and accurate tax application, this example shows how database triggers can automatically compute Value Added Tax (VAT) on each transaction. Businesses with significant transaction volumes, like retail, hospitality, or online marketplaces, will find this extremely helpful.

Compliance: By using an automated method to calculate VAT, companies adhere to tax laws without making mistakes by hand.

Efficiency: By doing away with tedious computations, administrative expense is decreased.

Audit readiness: Produces precise, dependable documentation for audits and tax reporting.

Broader Use: Tax compliance triggers can also be used for other financial indicators, including customs duties in import/export systems or income tax deductions in payroll systems. This illustrates how triggers can handle financial regulations in a variety of ways.

Theoretical Foundation: The concept of derived attributes, in which some fields are calculated using pre-existing data rather than being stored directly, is best illustrated by tax calculations at the database level. This minimizes data redundancy while maintaining correctness, which is consistent with normalizing approaches in database architecture.

Economic consequences: For companies functioning in regulated economies, tax compliance is a basic necessity. Automating the VAT computation process lowers administrative overhead, guarantees consistent execution of tax laws, and decreases the possibility of human error. Additionally, it makes tax reporting easier, which is essential for legal compliance and audits.

Automating VAT calculation using triggers ensures that tax computations are consistent, reducing discrepancies in financial records, simplifies the transaction workflow by eliminating the need for manual tax calculations at the application layer, and aligns with regulatory frameworks, ensuring businesses can generate tax reports on demand without additional data processing.

This approach underscores the importance of embedding domain-specific logic within the database layer, particularly for computations that are invariant across multiple applications or interfaces.

```
CREATE TABLE sales (
    sale_id NUMBER PRIMARY KEY,
    product_name VARCHAR2(100),
    sale_amount NUMBER NOT NULL,
    vat_amount NUMBER,
    total_amount NUMBER,
    sale_date DATE DEFAULT SYSDATE
);

CREATE OR REPLACE TRIGGER calculate_vat
BEFORE INSERT ON sales
FOR EACH ROW
BEGIN
    :NEW.vat_amount := :NEW.sale_amount * 0.19;
    :NEW.total_amount := :NEW.sale_amount + :NEW.vat_amount;
END;
/

INSERT INTO sales (sale_id, product_name, sale_amount)
    VALUES (1, 'Laptop Lenovo', 1000);
SELECT * FROM sales;
```

Figure 5. Implementation ex.

Explanation of the code:

Table Design:

- sale_amount: The base price of the product.
- vat_amount: The computed VAT, which is dynamically calculated by the trigger.
- total_amount: The final amount, including VAT.

Trigger Type: BEFORE INSERT: Ensures VAT is calculated and the total amount is updated before the transaction is saved in the database.

Logic: VAT is calculated as 19% of the sale_amount (customizable based on tax regulations). The total_amount is computed by adding the vat_amount to the sale_amount.

Outcome: The trigger eliminates manual VAT computation, ensuring accuracy and consistency in financial records.

8. Conclusions

This study shows how important Oracle SQL triggers are for process automation, data integrity, and improving operational efficiency in economic systems. The selected examples automatic stock updates, VAT computation, and overdraft prevention - provide useful examples of how database triggers tackle actual issues in banking, retail, and taxation. Triggers offer a reliable, effective, and consistent way to enforce intricate rules and uphold regulatory compliance by integrating business logic straight into the database.

Triggers are incredibly useful despite their seeming simplicity. In addition to protecting data integrity, they enable businesses to concentrate on innovation and expansion rather than ordinary operating issues by automating basic checks and computations. These little scripts carry out enormous responsibilities, guaranteeing that every transaction, stock adjustment, and tax calculation is done perfectly. They stand for a dedication to excellence and the creation of systems that benefit, not harm, humanity.

The examples demonstrate a number of benefits of employing triggers:

• Real-Time Automation: By automating repetitive processes like inventory updates, tax computations, and balance checks, triggers remove the need for human intervention.

• Data Integrity: Triggers lower the possibility of human or system errors by enforcing rules at the database level, ensuring that data stays consistent across all activities.

• Regulatory Compliance: By applying computations or validations automatically and consistently, triggers make it easier to comply with legal obligations, such as tax laws.

• Scalability: Triggers' ability to manage activities effectively without sacrificing performance is demonstrated by their use in high-transaction scenarios.

Prospects for the Future

Triggers' adaptability creates new opportunities for application and research. Future research might concentrate on:

• the effects of employing triggers in extremely complex systems on performance.

• extending trigger logic to help with predictive analytics, like estimating inventory requirements by looking at past patterns.

• looking into the use of triggers in cutting-edge fields where real-time data integrity is essential, such as blockchain.

To sum up, Oracle SQL triggers provide a strong way to automate business logic, guarantee compliance, and preserve data integrity in financial systems. The study illustrates how database triggers can spur creativity and productivity across a range of sectors by tackling real-world issues with these examples. The results reinforce the importance of triggers in a quickly changing digital economy and provide a basis for additional research and applications.

References

[1] IACOB, Ionel - ORACLE 10G: Designing and Implementing Applications with Databases Using SQL*PLUS. ISBN 978-9731293653, [page12-290] Pro Universitaria, oct. 2009

[2] IACOB, Ionel - ORACLE 10G: Developing IT Solutions Using PL/SQL and ORACLE DEVELOPER. ISBN 978-6065913134 [page 1-286]2nd Edition. Pro Universitaria, apr. 2009

[3] Nuijten, Alex, Barel, Patrick – Modern Oracle Database Programming: Level Up Your Skill Set to Oracle's Latest and Most Powerful Features in SQL, PL/SQL,– ISBN 978-1-4842-9166-5. [page 1-370]. Apres -feb 2023

[4] Malcher, Michelle; Kuhn, Darl – Pro Oracle Database 23c Administration: Manage and Safeguard Your Organization's Data –ISBN 978-1-4842-9898-5.[page 1-588]. Apress. Ian 2024

[5] Rosenzweig, Benjamin; Rakhimov, Elena – Oracle PL/SQL by Example. ISBN 978-0-13-8062835 [psge 1-240]] (6th Edition) – Pearson - Aug. 2023

Bibliography

BANERJEE, Abhishek - *Database Performance Monitoring with Oracle 23c.* Packt Publishing, 2023.

IACOB, Ionel - *ORACLE 10G: Designing and Implementing Applications with Databases Using SQL*PLUS.* Bucharest: Pro Universitaria, 2009

IACOB, Ionel - *ORACLE 10G: Developing IT Solutions Using PL/SQL and ORACLE DEVELOPER. 2nd Edition*. Bucharest: Pro Universitaria, 2009

FEUERSTEIN, Steven - *Resilient Oracle PL/SQL: Feature-Driven Development and Error Handling.* O'Reilly Media, 2023

HART, Matthew - *Oracle RMAN Backup & Recovery for Complex Systems*. McGraw-Hill, 2023.

MALCHER, Michelle; KUHN, Darl. - *Pro Oracle Database 23c Administration: Manage and Safeguard Your Organization's Data,* Apress, 2023.

JONES, Patrick - *SQL Trigger Optimization Techniques*. TechPress, 2023.

JOHNSON, Eric - *Cloud-Based Oracle Database and Trigger Usage*. Wiley, 2023.

SINGH, Amit - *Triggers in Oracle Cloud Database Environments*. O'Reilly Media, 2023.

GOEL, Ritesh - *Trigger Techniques for Enterprise Systems*. Springer, 2023.

https://docs.oracle.com/en/database/oracle/oracle-database/19/plsql/plsql-language-reference.html - Advanced PL/SQL Programming Techniques

https://docs.oracle.com/en/database/oracle/oracle-database/19/adfns/using-triggers.html Using Triggers in Oracle

https://docs.oracle.com/en/database/oracle/oracle-database/23/ - Oracle Corporation. Oracle Database Documentation 23c

https://dzone.com/ - DZone. Trigger-Driven Workflows in Modern Database Systems

https://www.infoworld.ro/ - InfoWorld Romania
https://www.itprotoday.com/ - IT Pro Today. Advanced SQL Techniques for Enterprise Systems
https://jitm.com - Journal of Information Technology. Trigger Optimization Techniques in Oracle Databases
https://www.packtpub.com/product/oracle-sql-recipes/ - Packt Publishing. Oracle SQL Recipes: A Problem-Solution Approach
https://www.rau.ro - RAU's official website. 20.06.2022
https://www.techhub.ro/ - Tech Hub Bucharest – IT Articles and Resources
https://www.techrepublic.com/ - TechRepublic. Optimizing Database Performance with Triggers in Oracle.