BioCycle-DB: Data-Tier Enforcement of Spatio-Temporal Logistics and Compliance Auditing in Poultry Waste Recovery

Heng Huey Jin[1], Muhammad Sharilazlan Bin Salleh[2], Ts. Muhammad Faheem Bin Mohd Ezani[3], Ts. Dr. Mailasan A/L Jayakrishnan[4], Muhammad Huzaifah Ismail[5], Dr. Mohamad Raziff bin Ramli[6]

[1,2,3,4]Fakulti Teknologi Maklumat dan Komunikasi Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, 76100, Malaysia

[5]School of Computing, Asia Pacific University of Technology and Innovation, No. 11, Jalan Teknologi 5, Taman Teknologi Malaysia, Bukit Jalil, 57000 Kuala Lumpur, Wilayah Persekutuan Kuala Lumpur

[6]UUM College of Arts and Sciences, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah Darul Aman, Malaysia

DOI: https://doi.org/10.47772/IJRISS.2026.100701145

Received: 10 August 2026; Accepted: 16 August 2026; Published: 22 August 2026

ABSTRACT

Modern circular agricultural supply chains require high-integrity digital platforms to coordinate the recovery and disposal of highly perishable organic waste. While the web-based application tier is always used to perform transaction processing and implement business logic and auditing processes, implementing all of this through the application tier often results in system performance issues, code duplication, and significant security threats due to the risks involved in accessing the database directly. The paper proposes the design of high-integrity database architecture which uses the Database Life Cycle Model and runs on Microsoft SQL Server Express 2022 platform.

A key database-tier design pattern utilized is 1:1 Extension/Association (One-to-One Delegation). The central USER table submits a role request (ROLE_REQUEST) when an external stakeholder registers. Upon administrative approval, the system enforces a strict 1:1 extension by creating a record in either the Slaughterhouse or Processor entity, each with its own independent primary key and a foreign key referencing the corresponding ROLE_REQUEST. To facilitate logistics management based on spatio-temporal parameters independent of application tier, we have designed and implemented: (1) an automated volume-based biohazard disposal routing engine using a database cursor inside a stored procedure (sp_ProcessExpiredReports) which automatically runs a 24-hour storage expiry routine and disposes of the waste either for Incineration, Effluent Treatment (ETP), or Anaerobic Digestion; (2) a filtered unique index constraint (idx_one_pending_per_processor_report) that eliminates transaction conflicts mathematically and duplicate active states; and (3) a non-repudiable audit-trail mechanism through DML triggers (trg_ProcessBatch_Audit) that captures global SESSION_CONTEXT variables and logs all field level changes. Overall, this architecture provides complete and tamper-evident traceability of waste from its creation to disposal/recovery and fulfils the strict reporting requirements for compliance under the Environmental Quality Act (EQA) 1974.

Keywords- Database Life Cycle (DBLC), Microsoft SQL Server, Filtered Unique Indexes, Database Triggers, Session Context Auditing, Circular Agriculture Traceability.

INTRODUCTION

The transition toward sustainable industrial and agricultural practices has intensified the need for efficient agricultural by-product management. In Malaysia, the chicken processing industry generates significant quantities of chicken blood waste, driven by halal slaughter protocols that mandate the complete drainage of blood from the chicken carcass. [1]. Over the years, this perishable waste product has been disposed of using both biological and chemical wastewater treatment facilities (Effluent Treatment Plants - ETPs) to minimize

the extremely high COD before disposal [2]. Although ETPs satisfy regulatory demands, such a disposal practice is costly, energy-intensive, and results in the destruction of valuable nitrogen and proteins.

Collecting and processing raw chicken blood waste immediately before biodegradation occurs allows it to be converted into high-value agricultural products such as organic fertilizer, blood meal and biogas. [3]. Nevertheless, there is an important "precision gap." There is no synchronized system for slaughterhouses and waste processors to collect information about their waste's quality, degradation possibilities, based on such factors as Chemical Oxygen Demand (COD), pH, storage temperature, and time of storage.

The majority of agricultural tracking systems implement core business logic (such as calculating distances between points geographically, setting 24-hour expiration period for storage, auditing logs) using application-tier code (such as PHP, Java, JavaScript, etc.) [4]. Such an approach poses considerable technical challenges:

- Direct Database Bypass: External reporting tools or direct database connections can easily insert or manipulate records, bypassing application-tier validation and violating environmental constraints.
- Transactional Latency: Running iterative sweeps of perishable inventories using application-tier cron-jobs creates massive processing overheads.
- Audit Vulnerability: Application-logged audits are prone to tampering and fail to track direct database actions executed by system administrators.

To resolve these architectural limitations, this paper presents BioCycle-DB, a high-integrity, data-tier solution implemented in Microsoft SQL Server Express 2022. BioCycle-DB enforces all spatio-temporal logistics rules (distance validation, 24-hour expiry, and volume-dependent biohazard disposal routing) and supply chain audit trails directly within the database engine via stored procedures, triggers, and filtered unique constraintsThis architecture enforces database-level referential integrity and reduces backend application overhead, providing a robust model for digital traceability in sustainable circular agriculture [5]. The integration of digital traceability within circular agriculture is essential for optimizing sustainable supply chains and preventing environmental pollution [6][7]. Validated data-tier tracking not only ensures rigorous compliance with environmental mandates such as Malaysia's Environmental Quality Act (EQA) 1974 [8] but also accelerates the recovery of perishable agricultural by-products. By enforcing strict spatio-temporal logistics directly at the database tier, BioCycle-DB provides a tamper-evident foundation that directly supports the transition towards zero-waste economies [9].

Related Works

Traditional poultry waste management relies heavily on physical infrastructure or localized digital logbooks, which fail to support automated, data-driven circular economy workflows. To map the current research gaps, standard technologies in this domain were systematically analysed:

- Effluent Treatment Plants (ETPs): These systems apply physical, chemical, and biological processes to reduce high organic pollutant loads, such as Chemical Oxygen Demand (COD) and Biochemical Oxygen Demand (BOD), before discharging the processed effluent into local waterways. However, as reviewed by Bustillo-Lecompte and Mehrvar (2015), while ETPs are crucial for environmental regulatory compliance, they focus exclusively on waste destruction rather than nutrient recovery. Moreover, ETPs incur heavy chemical and energy operational costs, and offer zero data-tracking, predictive quality screening, or stakeholder coordination capabilities [10].
- Mechanical Waste Handling Systems: Automated mechanical systems are typically deployed to transport, dewater, and physically manage solid slaughterhouse by-products within processing facilities. As highlighted by Mozhiarasi and Natarajan (2025), these setups significantly improve physical waste conveyance and volume reduction. Nonetheless, they remain strictly physical handling tools. They lack database-tier recording, automated quality classification, or digital synchronization interfaces between waste generators and processors [11].
- Rendering and Biogas Conversion Plants: Industrial rendering and anaerobic digestion plants are widely utilized to convert poultry by-products into industrial fats, high-protein blood meal, or renewable methane gas. According to Bułkowska and Zielińska (2024), although these conversion technologies support circular

recovery, they operate strictly at the post-collection phase. Consequently, they lack pre-collection logistics coordination, data-driven raw input quality screening, and spatio-temporal tracking mechanisms required to prevent degraded or bio-hazardous inputs from entering the recovery supply chain [12].

To highlight the technical advancement of the proposed database-tier framework, Table I compares these traditional technologies against BioCycle-DB.

Table I Feature Matrix of Existing Systems and BioCycle-DB

Features	ETP Systems	Mechanical Conveyors	Rendering / Biogas	BioCycle-DB (Proposed)
Primary Function	Waste Treatment	Waste Handling	Waste Conversion	Traceability & Recovery
Data Recording	None	None	None	Centralized Digital Tier
Quality Screening	None	None	None	Data-Tier ML Classification
Stakeholder Coordination	None	None	None	Integrated Client-Server Map
Spatio-Temporal Logic	No	No	No	Yes (DBMS Enforced)
Automated Expiry Tracking	No	No	No	Yes (24-Hour SP Cursor)
Data Recording	None	None	None	Centralized Digital Tier
Reporting & BI	Manual Logs	None	Static Sheets	Real-Time DirectQuery (Power BI)
Spatio-Temporal Logic	No	No	No	Yes (DBMS Enforced)
Tamper-Evident Auditing	No	No	No	Yes (DML Audit Triggers)

METHODOLOGY

We applied the Database Life Cycle (DBLC) framework to systematically construct and optimize the BioCycle database schema as shown in Fig. 1. The execution of the six DBLC phases is detailed below. The structural robustness of BioCycle-DB relies on the Database Life Cycle (DBLC), a framework widely recognized for ensuring data consistency and scalable performance across enterprise systems [13] [14]. Translating conceptual models directly into physical relational schemas guarantees that all referential dependencies are mathematically validated before deployment [15]. This rigorous normalization process mitigates common data anomalies typically found in fragmented agricultural tracking applications [16].

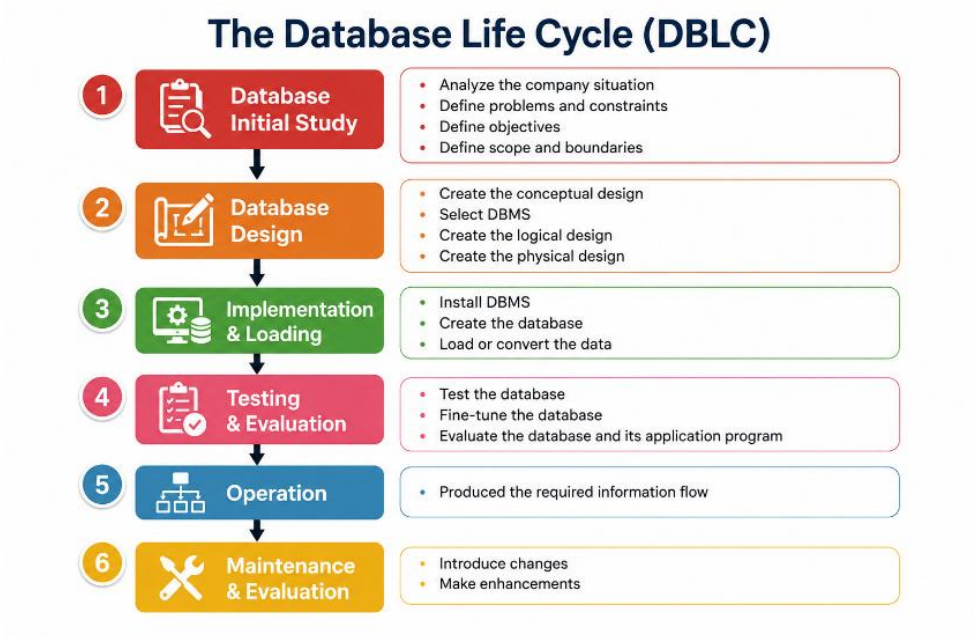

Fig 1. Application of the Database Life Cycle (DBLC) Framework.

The DBLC phases are applied in the development of the BioCycle database as follows:

- Database Initial Study: Requirements were gathered from environmental compliance guidelines and local poultry operations. Four distinct stakeholder roles were established: SuperAdmins, Administrators, Slaughterhouse Operators, and waste Processors.
- Database Design: Conceptual data models were mapped, resulting in a normalized Entity-Relationship Diagram (ERD). Relational normalization was carried out to the Third Normal Form (3NF) to construct a comprehensive Data Dictionary with structural integrity constraints (CHECK, FOREIGN KEY, and UNIQUE). The database structure is designed by identifying key entities such as USER, BLOOD_WASTE_REPORT, COLLECTION_REQUEST, PROCESS_BATCH, CLASSIFICATION and DISPOSAL_RECORD. Relationships between these entities are defined to ensure data integrity and proper system flow.
- Implementation and Loading: The physical database schema was compiled using Microsoft SQL Server Express 2022. Structural tables were generated using SQL DDL, and sample transactional records were seeded via SQL Server Management Studio (SSMS) to validate the referential connections.
- Testing and Evaluation: The system is tested to ensure all database operations function correctly. This includes testing data insertion, updates, queries, classification results and collection request processing to ensure data accuracy and consistency.
- Operation: The physical engine was integrated with the application layer. The PHP application communicates via the SQLSRV PDO extension, while the Python-based Random Forest classifier interacts via a Flask REST API cURL call to classify batch quality degradation parameters.
- Maintenance: Database-tier optimization strategies, including index restructuring for audit logs and automated transaction backups, were scheduled to ensure system survivability.

Proposed High-Integrity Database Architecture

3-Tier Client-Server Architecture & Normalized Schema

BioCycle-DB operates as the foundational Data Layer for a three-tier client-server web architecture. To maintain data consistency, we designed a normalized 9-entity schema to model the entire poultry waste supply chain as shown in Fig. 2.

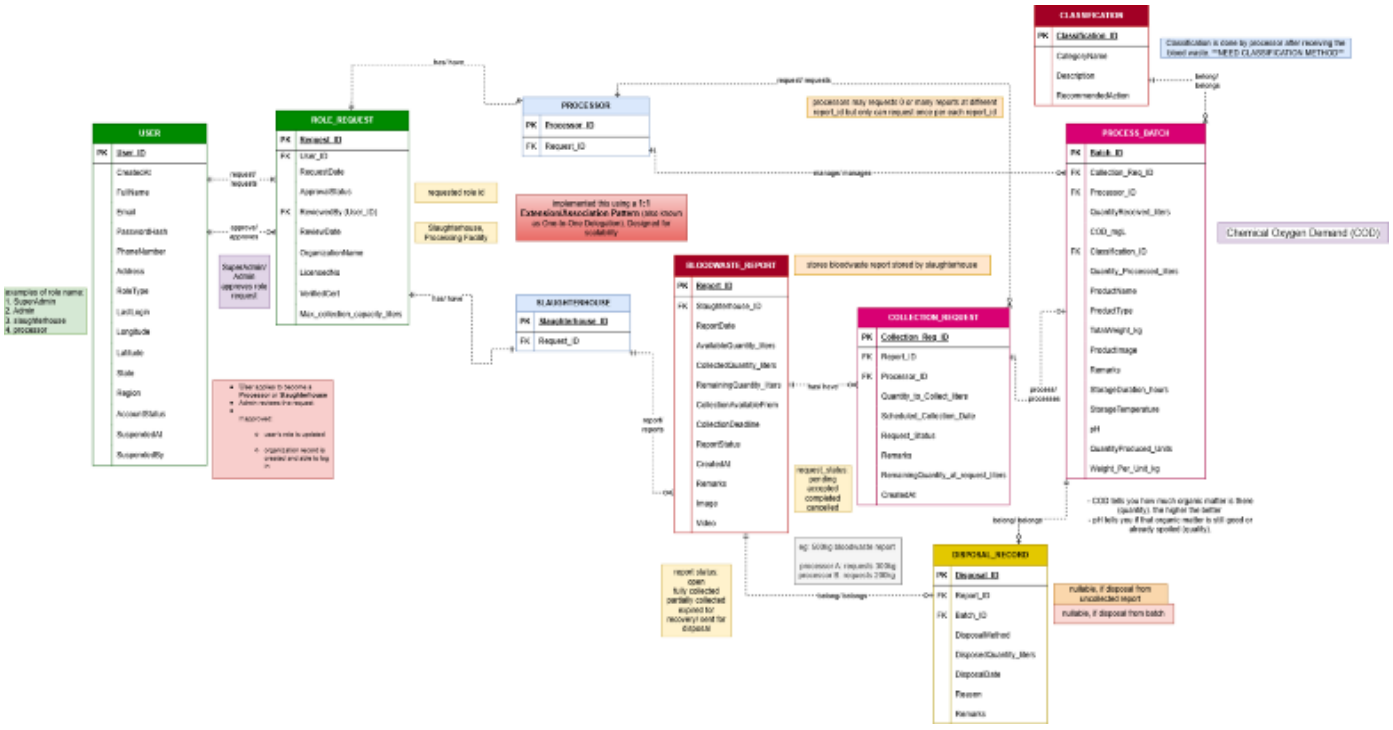

Fig 2. 9-Entity Relational Data Model with 1:1 Specialization-Generalization.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

A key database-tier design pattern utilized is 1:1 Extension/Association (One-to-One Delegation). The central USER table submits a role request (ROLE_REQUEST) when an external stakeholder registers. Upon administrative approval, the system enforces a strict 1:1 extension using specialization-generalization.

This creates a record in either the SLAUGHTERHOUSE or PROCESSOR entity, linking their primary keys back to the corresponding ROLE_REQUEST. This prevents user-role duplication and guarantees that a single login identity cannot act as both a waste generator and a waste collector.

Spatio-Temporal Expiry and Volume-Based Disposal Logic

To prevent environmental contamination, blood waste must be recovered within 24 hours of reporting. If the deadline passes without collection, the database must transition the report's status to Expired and route it to a DISPOSAL_RECORD table to alert regulatory authorities.

Rather than running vulnerable application-level loops, we implemented sp_ProcessExpiredReports directly within the SQL Server engine. It implements a database-level cursor that calculates the expired remaining waste quantity and applies a volume-dependent logic to automatically route waste to the safest disposal method:

```
USE [BioCycleDB]

GO

/****** Object:  StoredProcedure [dbo].[sp_ProcessExpiredReports] ******/

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

DROP PROCEDURE IF EXISTS [dbo].[sp_ProcessExpiredReports]

GO

CREATE PROCEDURE [dbo].[sp_ProcessExpiredReports]

AS

BEGIN

   SET NOCOUNT ON;

   -- STEP 1: Update status to Expired first

   UPDATE BLOODWASTE_REPORT

   SET ReportStatus = 'Expired'

   WHERE CollectionDeadline < GETDATE()

   AND ReportStatus IN ('Open', 'Partial');
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

```
-- STEP 2: Move expired reports to DISPOSAL_RECORD

DECLARE @Report_ID VARCHAR(20);

DECLARE @RemainingQty DECIMAL(10,2);

DECLARE @CollectedQty DECIMAL(10,2);

DECLARE @AvailableQty DECIMAL(10,2);

DECLARE @Disposal_ID VARCHAR(20);

DECLARE @NextNum INT;

DECLARE @DisposalQty DECIMAL(10,2);

DECLARE @DisposalMethod VARCHAR(100);

DECLARE @DisposalReason VARCHAR(500);

DECLARE expired_cursor CURSOR FOR

  SELECT

    b.Report_ID,

    b.RemainingQuantity_liters,

    b.CollectedQuantity_liters,

    b.AvailableQuantity_liters

  FROM BLOODWASTE_REPORT b

  LEFT JOIN DISPOSAL_RECORD d ON b.Report_ID = d.Report_ID

  WHERE b.ReportStatus = 'Expired'

  AND d.Report_ID IS NULL

  AND b.RemainingQuantity_liters >= 0;

OPEN expired_cursor;

FETCH NEXT FROM expired_cursor INTO @Report_ID, @RemainingQty, @CollectedQty, @AvailableQty;

WHILE @@FETCH_STATUS = 0

BEGIN

  -- Get next Disposal ID - ONLY from DSL### format

  SELECT @NextNum = ISNULL(MAX(CAST(SUBSTRING(Disposal_ID, 4, LEN(Disposal_ID)-3) AS INT)), 0) + 1
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

```
    FROM DISPOSAL_RECORD
    WHERE Disposal_ID LIKE 'DSL%';  -- Only get IDs that start with 'DSL'
    -- If no DSL records exist, start from 1
    IF @NextNum IS NULL OR @NextNum = 0
       SET @NextNum = 1;
    SET @Disposal_ID = 'DSL' + RIGHT('000' + CAST(@NextNum AS VARCHAR), 3)
    -- Determine disposal quantity
    IF @RemainingQty > 0
       SET @DisposalQty = @RemainingQty;
    ELSE
       SET @DisposalQty = @AvailableQty;
============================================================
    -- DISPOSAL METHOD RECOMMENDATION BASED ON QUANTITY (LITERS)
    -- ============================================================
IF @DisposalQty > 5000
    BEGIN
       SET @DisposalMethod = 'Anaerobic Digestion / Biogas';
       SET @DisposalReason = 'High volume (' + CAST(@DisposalQty AS VARCHAR) + ' liters) - Recommended for biogas recovery. Research shows 70-80% of slaughterhouse biogas potential comes from blood (Edström et al., 2003). Converts waste to energy instead of landfill.';
    END
    ELSE IF @DisposalQty >= 500
    BEGIN
       SET @DisposalMethod = 'ETP with Pre-treatment';
       SET @DisposalReason = 'Moderate volume (' + CAST(@DisposalQty AS VARCHAR) + ' liters) - Requires ETP with pre-treatment. Must be diluted/equalized before treatment.';
    END
    ELSE
    BEGIN
       SET @DisposalMethod = 'Incineration';
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

```
  SET @DisposalReason = 'Low volume (' + CAST(@DisposalQty AS VARCHAR) + ' liters) - Below economic processing threshold. Safe biohazard disposal required per clinical waste standards.';

      END

      -- Insert into DISPOSAL_RECORD with calculated method and reason

      INSERT INTO DISPOSAL_RECORD (

         Disposal_ID, Report_ID, Batch_ID, DisposalMethod,

         DisposedQuantity_liters, DisposalDate, Reason, Remarks)

      VALUES (

         @Disposal_ID, @Report_ID, NULL, @DisposalMethod,

         @DisposalQty, GETDATE(),

         @DisposalReason,

         CASE

            WHEN @RemainingQty > 0 AND @CollectedQty > 0

               THEN 'Partial collection - ' + CAST(@CollectedQty AS VARCHAR) + ' liters collected by processor, ' + CAST(@RemainingQty AS VARCHAR) + ' liters uncollected after 24 hours'

            WHEN @RemainingQty = 0 AND @CollectedQty > 0

               THEN 'Fully collected (' + CAST(@CollectedQty AS VARCHAR) + ' liters) - compliance record (no disposal needed)'

            ELSE

  'No collection - ' + CAST(@AvailableQty AS VARCHAR) + ' liters uncollected after 24 hours. Auto-disposal per slaughterhouse waste management policy.'

  END     );

      FETCH NEXT FROM expired_cursor INTO @Report_ID, @RemainingQty, @CollectedQty, @AvailableQty;

   END

   CLOSE expired_cursor;

   DEALLOCATE expired_cursor;

END;

GO
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

Tamper-Evident Auditing Utilizing Session Context

To achieve regulatory compliance with Malaysia's Environmental Quality Act (EQA) 1974, the database must prevent bad actors (such as unauthorized handlers) from modifying the quality degradation parameters of process batches. We implemented a robust database trigger, trg_ProcessBatch_Audit, which runs after every DML operation (INSERT, UPDATE, DELETE) on the PROCESS_BATCH table [17]. Captures the performing user's identity from session context. For updates, records old and new values side by side and determines the image change action (UPLOADED / REPLACED / REMOVED / UNCHANGED). Only fires when product fields are non-null or have meaningfully changed. It captures user context dynamically from SQL Server's global SESSION_CONTEXT:

```
CREATE TRIGGER trg_ProcessBatch_Audit
ON PROCESS_BATCH
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   SET NOCOUNT ON;
   IF EXISTS (SELECT * FROM inserted)
     AND NOT EXISTS (SELECT * FROM deleted)
   BEGIN
      INSERT INTO product_audit_log (
         Batch_ID, Collection_Req_ID, Processor_ID,
         PerformedBy_UserID, PerformedBy_Name,
         PerformedBy_Role, ActionType, ActionTime,
         IPAddress, ProductName, ProductType,
         QuantityProduced_Units, HasProductImage,
         ProductImage, Remarks, Classification_ID,
         ChangeSummary)
      SELECT i.Batch_ID, i.Collection_Req_ID,
         i.Processor_ID,
         ISNULL(CAST(SESSION_CONTEXT(N'UserID')
            AS VARCHAR(20)),'SYSTEM'),
         ISNULL(CAST(SESSION_CONTEXT(N'UserName')
            AS NVARCHAR(200)),'SYSTEM'),
         ISNULL(CAST(SESSION_CONTEXT(N'UserRole')
            AS VARCHAR(50)),'SYSTEM'),
         'CREATE', GETDATE(),
         ISNULL(CAST(SESSION_CONTEXT(N'IPAddress')
            AS VARCHAR(45)),NULL),
         i.ProductName, i.ProductType,
```

```
        i.QuantityProduced_Units,
        CASE WHEN i.ProductImage IS NOT NULL
            AND i.ProductImage!=''
            THEN 1 ELSE 0 END,
        i.ProductImage,
        CAST(i.Remarks AS NVARCHAR(MAX)),
        i.Classification_ID,'New product created'
    FROM inserted i
    WHERE i.ProductName IS NOT NULL;
  END
  ELSE IF EXISTS (SELECT * FROM deleted)
     AND NOT EXISTS (SELECT * FROM inserted)
  BEGIN
    INSERT INTO product_audit_log (
      Batch_ID, Collection_Req_ID, Processor_ID,
      PerformedBy_UserID, PerformedBy_Name,
      PerformedBy_Role, ActionType, ActionTime,
      IPAddress, ProductName, ProductType,
      QuantityProduced_Units, HasProductImage,
      ProductImage, Remarks, Classification_ID,
      ChangeSummary)
    SELECT d.Batch_ID, d.Collection_Req_ID,
      d.Processor_ID,
ISNULL(CAST(SESSION_CONTEXT(N'UserID')
        AS VARCHAR(20)),'SYSTEM'),
      ISNULL(CAST(SESSION_CONTEXT(N'UserName')
        AS NVARCHAR(200)),'SYSTEM'),
      ISNULL(CAST(SESSION_CONTEXT(N'UserRole')
        AS VARCHAR(50)),'SYSTEM'),
      'DELETE', GETDATE(),
ISNULL(CAST(SESSION_CONTEXT(N'IPAddress')
        AS VARCHAR(45)),NULL),
      d.ProductName, d.ProductType,
      d.QuantityProduced_Units,
      CASE WHEN d.ProductImage IS NOT NULL
         AND d.ProductImage!=''
         THEN 1 ELSE 0 END,
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

```
        d.ProductImage,
        CAST(d.Remarks AS NVARCHAR(MAX)),
        d.Classification_ID, 'Product deleted'
    FROM deleted d
    WHERE d.ProductName IS NOT NULL;
  END
  ELSE IF EXISTS (SELECT * FROM inserted)
     AND EXISTS (SELECT * FROM deleted)
  BEGIN
    INSERT INTO product_audit_log (
        Batch_ID, Collection_Req_ID, Processor_ID,
        PerformedBy_UserID, PerformedBy_Name,
        PerformedBy_Role, ActionType, ActionTime,
        IPAddress, ProductName, ProductType,
        QuantityProduced_Units, HasProductImage,
        ProductImage, Remarks, Classification_ID,
        OldProductName, OldProductType,
        OldQuantityProduced_Units,
OldHasProductImage,
        OldRemarks, ImageAction,
        OldImagePath, NewImagePath,
        ChangeSummary, ChangeDetails)
    SELECT i.Batch_ID, i.Collection_Req_ID,
        i.Processor_ID,
        ISNULL(CAST(SESSION_CONTEXT(N'UserID')
          AS VARCHAR(20)),'SYSTEM'),
        ISNULL(CAST(SESSION_CONTEXT(N'UserName')
          AS NVARCHAR(200)),'SYSTEM'),
        ISNULL(CAST(SESSION_CONTEXT(N'UserRole')
          AS VARCHAR(50)),'SYSTEM'),
        'UPDATE', GETDATE(),
ISNULL(CAST(SESSION_CONTEXT(N'IPAddress')
          AS VARCHAR(45)),NULL),
        i.ProductName, i.ProductType,
        i.QuantityProduced_Units,
        CASE WHEN i.ProductImage IS NOT NULL
           AND i.ProductImage!=''
```

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VII July 2026

```
    THEN 1 ELSE 0 END,
i.ProductImage,
CAST(i.Remarks AS NVARCHAR(MAX)),
i.Classification_ID,
d.ProductName, d.ProductType,
d.QuantityProduced_Units,
CASE WHEN d.ProductImage IS NOT NULL
    AND d.ProductImage!=''
    THEN 1 ELSE 0 END,
CAST(d.Remarks AS NVARCHAR(MAX)),
CASE
  WHEN (d.ProductImage IS NULL
       OR d.ProductImage='')
      AND (i.ProductImage IS NOT NULL
          AND i.ProductImage!='')
      THEN 'UPLOADED'
  WHEN (d.ProductImage IS NOT NULL
       AND d.ProductImage!='')
      AND (i.ProductImage IS NULL
          OR i.ProductImage='')
      THEN 'REMOVED'
  WHEN d.ProductImage!=i.ProductImage
      AND i.ProductImage IS NOT NULL
      THEN 'REPLACED'
  ELSE 'UNCHANGED' END,
d.ProductImage, i.ProductImage,
'Product updated',
CONCAT(
  CASE WHEN ISNULL(d.ProductName,'')
          !=ISNULL(i.ProductName,'')
    THEN CONCAT('ProductName: ',
     ISNULL(d.ProductName,'NULL'),
     ' -> ',ISNULL(i.ProductName,'NULL'),
     '; ') ELSE '' END,
  CASE WHEN ISNULL(d.ProductType,'')
          !=ISNULL(i.ProductType,'')
    THEN CONCAT('ProductType: ',
```

```
            ISNULL(d.ProductType,'NULL'),
            ' -> ',ISNULL(i.ProductType,'NULL'),
            '; ') ELSE '' END,
        CASE WHEN ISNULL(d.QuantityProduced_Units,0)
              !=ISNULL(i.QuantityProduced_Units,0)
          THEN CONCAT('Quantity: ',
            CAST(ISNULL(d.QuantityProduced_Units,0)
              AS VARCHAR),' -> ',
            CAST(ISNULL(i.QuantityProduced_Units,0)
              AS VARCHAR),'; ') ELSE '' END,
        CASE WHEN ISNULL(d.Classification_ID,0)
                !=ISNULL(i.Classification_ID,0)
          THEN 'Classification changed; '
          ELSE '' END)
    FROM inserted i
    INNER JOIN deleted d ON i.Batch_ID=d.Batch_ID
    WHERE
      ISNULL(d.ProductName,'')
        !=ISNULL(i.ProductName,'')
      OR ISNULL(d.ProductType,'')
        !=ISNULL(i.ProductType,'')
      OR ISNULL(d.QuantityProduced_Units,0)
        !=ISNULL(i.QuantityProduced_Units,0)
      OR ISNULL(d.ProductImage,'')
        !=ISNULL(i.ProductImage,'')
      OR ISNULL(d.Classification_ID,0)
        !=ISNULL(i.Classification_ID,0);
  END
END;
```

However, it is essential to state that despite the enforcement of continuous audit logging for all transactional DML actions (INSERT, UPDATE, DELETE) through trg_ProcessBatch_Audit, it functions as a tamper-evident system rather than an absolute tamper-proof barrier. It is possible for the database administrator (DBA) with high privileges (e.g., sysadmin or db_owner permissions) to turn off database triggers or to modify the table directly – product_audit_log. In order to mitigate this vulnerability of an internal threat, it is a highly auditable approach with future plans of integration of ledger sealing via Blockchain nodes (Section VII).

Spatial-Temporal Constraints Enforced via Filtered Unique Index

In circular agricultural logistics, a common transactional conflict occurs when multiple Processors submit redundant Pending collection requests for a single blood waste report simultaneously. This creates a race

condition that can crash backend application scripts. To solve this, BioCycle-DB enforces transactional isolation directly at the database engine level by compiling a Filtered Unique Index [18]. The filtered unique index enforces the business rule that one processor may only hold one active pending request per blood waste report at a time, while still allowing multiple processors to request the same report simultaneously. The composite index speeds up queries that filter collection requests by report and status.

```
CREATE UNIQUE INDEX
   idx_one_pending_per_processor_report
ON COLLECTION_REQUEST (Report_ID, Processor_ID)
WHERE Request_Status = 'Pending';
CREATE INDEX idx_report_status
ON COLLECTION_REQUEST
   (Report_ID, Request_Status);
```

This filtered index mathematically restricts the index validation only to active, pending rows. It allows a processor to submit multiple historical requests to the same slaughterhouse over time but guarantees that the database engine rejects any insertion of a duplicate Pending request for a single report.

Data-Tier Enforcement of Processor Daily Capacity Constraints

In order to avoid operational overloading, logistic problems, as well as the risk of potential contamination, BioCycle-DB imposes a mandatory physical constraint on the maximum daily collection volume for each waste processor. Each of the processors is subjected to a Max_collection_capacity_liters constraint, which is stored in the ROLE_REQUEST table. Instead of using potentially flawed application-level business logic, the business rule is implemented directly on the database level through a transactional constraint in the COLLECTION_REQUEST table. Before the creation of any new record in the database, the sum of all the current collections volume is calculated for the current day. The transaction is immediately stopped and a database-specific exception (Msg 50000, etc.) is thrown if the sum of the collections exceeds the daily threshold imposed for that particular processor. Moreover, the database job, which resets the cumulative daily collections volume for all the processors to 0 every midnight (00:00), provides automatic and highly auditable temporal workload management independent of the application.

Real-Time Data-Tier Analytics via Power BI DirectQuery Integration

To facilitate real-time monitoring and compliance reporting without compromising database transactional performance, BioCycle-DB integrates with Microsoft Power BI Desktop (Version 2.156.951.0). Unlike traditional analytical models that rely on periodic data import schedules which introduce data redundancy and synchronization latencies this architecture implements the DirectQuery storage mode. Under the DirectQuery paradigm, no raw transactional data is extracted or duplicated outside the relational database engine. Instead, when a SuperAdmin or Administrator accesses the Product Analytics Dashboard, the Power BI engine dynamically translates user visual interactions into native T-SQL queries. These queries are executed directly on the Microsoft SQL Server Express 2022 physical tier.

To support these demanding analytical workloads without causing resource contention or blocking active OLTP transactions (such as concurrent slaughterhouse waste reporting or processor collection requests), the database utilizes the optimized physical indexes. Specifically, the composite index idx_report_status and individual foreign key indexes on the product_audit_log enable the SQL Server query optimizer to generate highly efficient execution plans, resolving analytical aggregates (e.g., total volume processed, yield efficiency) in sub-second response times. By utilizing Power BI's DirectQuery mode, BioCycle-DB facilitates real-time data-tier analytics without the risk of data redundancy or synchronization latencies typical of scheduled data imports.

Integrity And Performance Evaluation

To validate the robustness of the BioCycle-DB architecture in enforcing spatio-temporal logistics and compliance auditing directly at the data tier, a series of rigorous transactional and database constraint validation tests were executed. To validate the structural integrity of BioCycle-DB, this evaluation focuses on verifying logical correctness and database resilience against direct-bypass DML query injections. This confirms that the constraint engine successfully blocks unauthorized state transitions, maintaining strict data consistency independently of the application layer.

Spatio-Temporal Logistics and Volume-Based Disposal Logic Validation

The spatial-temporal stored procedure sp_ProcessExpiredReports was evaluated by seeding the database with actual expired reports extracted from the student's operational prototype data. Under Malaysian environmental regulations, highly organic slaughterhouse blood waste must be processed within a 24-hour storage window to prevent biodegradation and hydrogen sulfide (H2S) emission. Upon calling sp_ProcessExpiredReports, the database-tier engine automatically executes a bulk update on the BLOODWASTE_REPORT table, transitioning reports past their CollectionDeadline and currently in 'Open' or 'Partial' states to 'Expired'. Subsequently, a T-SQL cursor (expired_cursor) processes each expired record and executes the volume-based mathematical routing logic:

$$\text{DisposalMethod} = \begin{cases} \text{Incineration}, & \text{if } Q < 500 \text{ liters} \\ \text{ETP with Pre} - \text{treatment}, & \text{if } 500 \leq Q < 5000 \text{ liters} \\ \text{Anaerobic Digestion/Biogas}, & \text{if } Q \geq 5000 \text{ liters} \end{cases}$$

Where Q is the uncollected remaining volume of blood waste (in liters).

The operational thresholds of 500 liters and 5,000 liters are based on both biosecurity standards and economic models. Below 500 liters, the blood is sent to Incineration since the small, scattered quantities do not reach the economic viability point of transportation and active recovery; thus, thermal destruction is the safest biosecurity solution to stop pathogens from spreading. For the moderate quantities of 500 to 5,000 liters, it is possible to route the blood to Effluent Treatment Plant, although pre-treatment and equalization will be necessary to reduce the high chemical oxygen demand (COD) of blood prior to biological treatment, thus not to shock ETP filters.

Lastly, more than 5,000 liters per batch are the highly concentrated quantities of organic matter, where energy recovery becomes economically viable. According to Edström et al. (2003), between 70-80% of the potential biogas from slaughterhouses can be produced using poultry blood [19].

Table II demonstrates the actual data-tier processing outcomes using real reports from the student's database history:

Table II Execution Outcomes of Spatio-Temporal Expiry and Volume Routing

Report ID	Date	Remaining (liters)	Recommended Disposal	Reason	Action
BWR16	13 May 2026, 10:35PM	538.00 liters	ETP with Pre-treatment	Medium daily volume (500–5,000 L). Requires nutrient removal before discharge. Blood contains 3–5 g/L amino acids.	View
BWR14	28 Apr 2026, 11:06 AM	49.90 liters	Incineration	Low daily volume (<500 L). Standard biohazard disposal for small quantities.	View

BWR10	24 Apr 2026, 09:13	200.00 liters	Incineration	Low daily volume (<500 L). Standard biohazard disposal for small quantities.	View
BWR8	24 Apr 2026, 09:13	9,000.00 liters	Anaerobic Digestion / Biogas	High daily volume (>5,000 L). Economically viable for energy recovery. Blood has high methane potential.	View
BWR7	22 Apr 2026, 01:23 AM	50.00 liters	Incineration	Standard biohazard disposal for small quantities.	View
BWR6	20 Apr 2026, 05:54 PM	300.00 liters	Incineration	Low daily volume (<500 L). Standard biohazard disposal for small quantities.	View
BWR05	20 Apr 2026, 02:20 PM	250.00 liters	Incineration	Low daily volume (<500 L). Standard biohazard disposal for small quantities.	View
BWR04	17 Apr 2026, 04:57 PM	200.00 liters	Incineration	Low daily volume (<500 L). Standard biohazard disposal for small quantities.	View

Below Fig. 3 is the example web page for Expired Reports Disposal Required table, listing all expired blood waste reports in tabular format with columns for Report ID, Date, remaining (liters), Recommended Disposal method (colour-coded badge), Reason (describing the volume-based disposal justification) and a View action button same as Table II above.

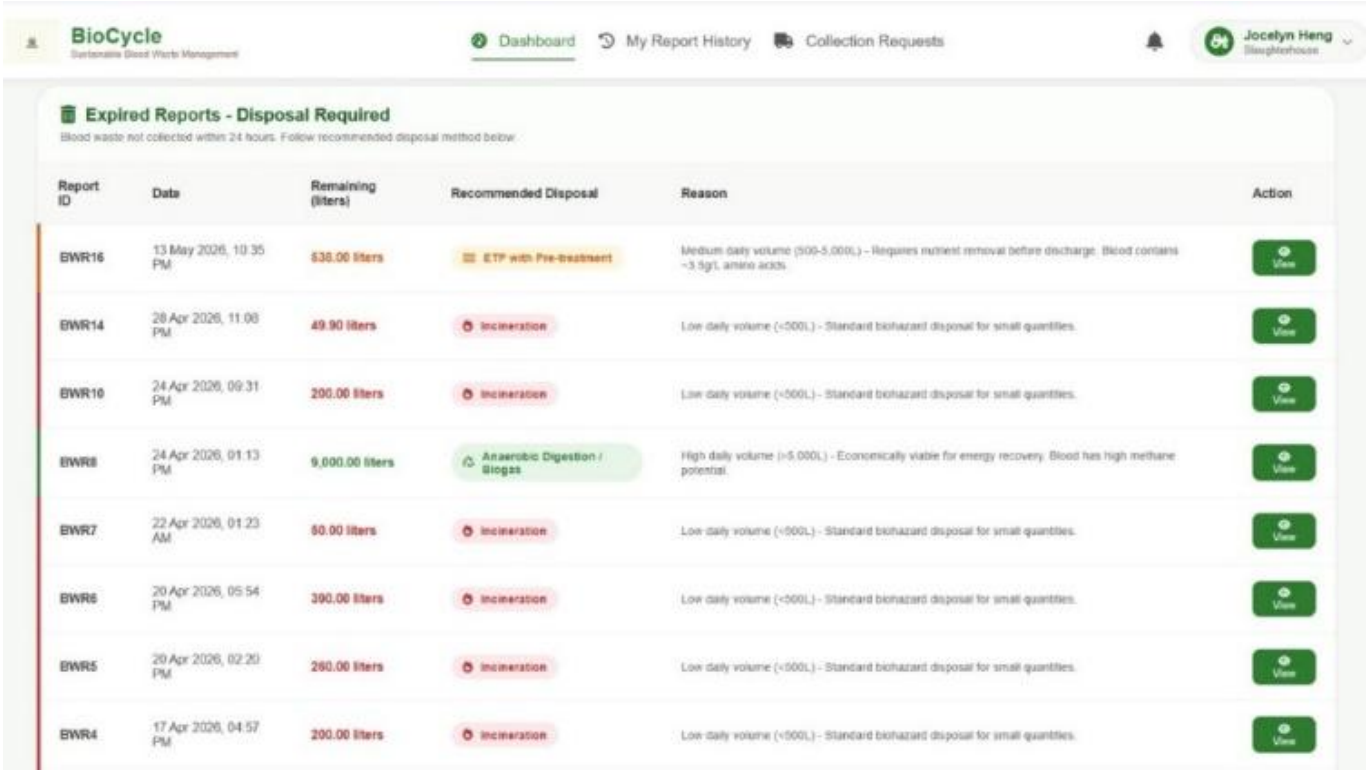

Fig 3. Expired reports execution outcomes of spatio-temporal expiry and volume routing

Database-Level Constraint and Bypass Validation

To evaluate database-tier resilience against direct bypass attempts, we simulated unauthorized DML queries directly within SQL Server Management Studio (SSMS). Table III outlines the logical constraints, the direct bypass execution statements, the native MS SQL Server engine error codes returned, and the final validation outcomes.

Table III Relational Database Integrity and Logical Constraint Validation Matrix

Test Case ID	Target Table & Enforced Rule	Direct SQL Query Execution (Bypass Simulation)	Actual Engine Response / Transaction Status	Outcome
TC-01	USER table – RoleType Domain Constraint	INSERT INTO [USER] (User_ID, FullName, Email, PasswordHash, PhoneNumber, Address, RoleType, State, Region, Latitude, Longitude, AccountStatus) VALUES ('USR099', 'Unauthorized User', 'unauth@test.com', 'hash123', '012-3456789', 'No. 1, Melaka', 'Farmer', 'Melaka', 'Durian Tunggal', 2.27, 102.27, 'Active');	Msg 547, Level 16, State 0, Line 1 The INSERT statement conflicted with the CHECK constraint "CK__USER__RoleType". The conflict occurred in database "BioCycleDB", table "dbo.USER", column 'RoleType'. The statement has been terminated. Completion time: 2026-08-06T18:03:13.7854340+08:00	PASSED
TC-02	BLOODWASTE_REPORT table – ReportStatus Domain Constraint	UPDATE BLOODWASTE_REPORT SET ReportStatus = 'Degraded' WHERE Report_ID = 'BWR16';	Msg 547, Level 16, State 0, Line 1 The UPDATE statement conflicted with the CHECK constraint "CK__BLOODWAST__Repor__6477ECF3". The conflict occurred in database "BioCycleDB", table "dbo.BLOODWASTE_REPORT", column 'ReportStatus'. The statement has been terminated. Completion time: 2026-08-06T18:07:07.2120623+08:00	PASSED
TC-03	COLLECTION_REQUEST table – Filtered Unique Index (idx_one_pending_per_processor_report)	Step 1: Insert the first Pending collection request (Success) INSERT INTO COLLECTION_REQUEST (Collection_Req_ID, Report_ID, Processor_ID, Quantity_to_Collect_liters, Scheduled_Collection_Date, Request_Status, Remarks, RemainingQuantity_at_request_liters) VALUES ('CR1001', 'BWR18', 'PRS002', 150.00, '2026-08-10 09:00:00', 'Pending', 'Test Request 1', 150.00); Step 2: Insert a SECOND Pending request for the SAME report and processor	Msg 2601, Level 14, State 1, Line 2 Cannot insert duplicate key row in object 'dbo.COLLECTION_REQUEST' with unique index 'idx_one_pending_per_processor_report'. The duplicate key value is (BWR18, PRS002). The statement has been terminated. Completion time: 2026-08-18T14:08:16.5986315+08:00	PASSED

		with a different ID (Blocked by Filtered Index) INSERT INTO COLLECTION_REQUEST (Collection_Req_ID, Report_ID, Processor_ID, Quantity_to_Collect_liters, Scheduled_Collection_Date, Request_Status, Remarks, RemainingQuantity_at_request_liters) VALUES ('CR1002', 'BWR18', 'PRS002', 150.00, '2026-08-10 10:00:00', 'Pending', 'Test Request 2', 150.00);		
TC-04	PROCESS_BATCH table – Audit Trigger with Session Context (trg_ProcessBatch_Audit)	DECLARE @Context VARBINARY(128)=CAST('USR0007' AS VARBINARY(128)); EXEC sys.sp_set_session_context @key=N'UserID', @value=@Context; UPDATE PROCESS_BATCH SET ProductName='Adulterated Organic Fertilizer' WHERE Batch_ID='BAT007';	(1 row affected) Completion time: 2026-08-06T18:13:55.0381045+08:00	PASSED

The result for every test case for TC-01, TC-02, TC-03 and TC-04 are shown in Fig. 4, Fig. 5, Fig. 6 and Fig.7 below:

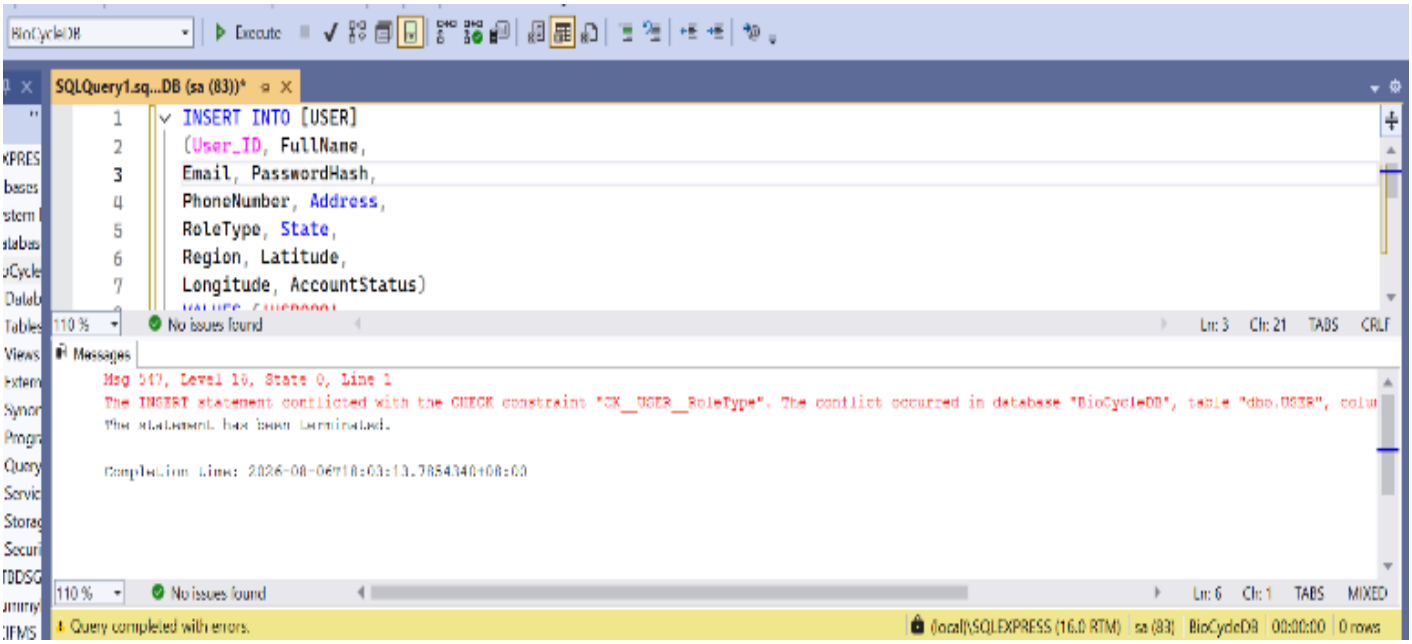

Fig 4. TC-01 SQL Validation Result for USER RoleType CHECK Constraint

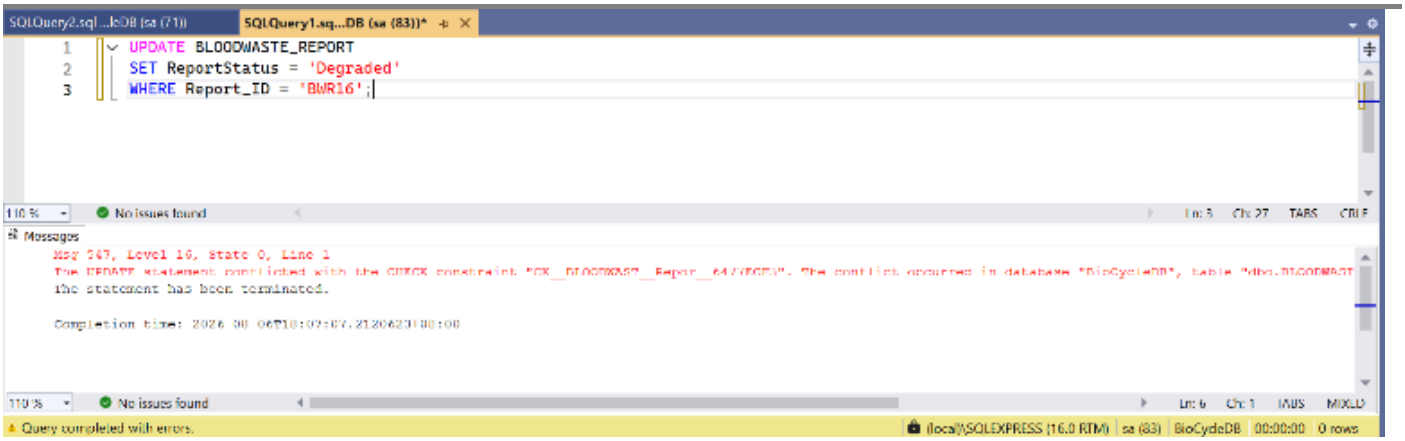

Fig 5. TC-02 SQL Validation Result for BLOOD_WASTE_REPORT ReportStatus CHECK Constraint

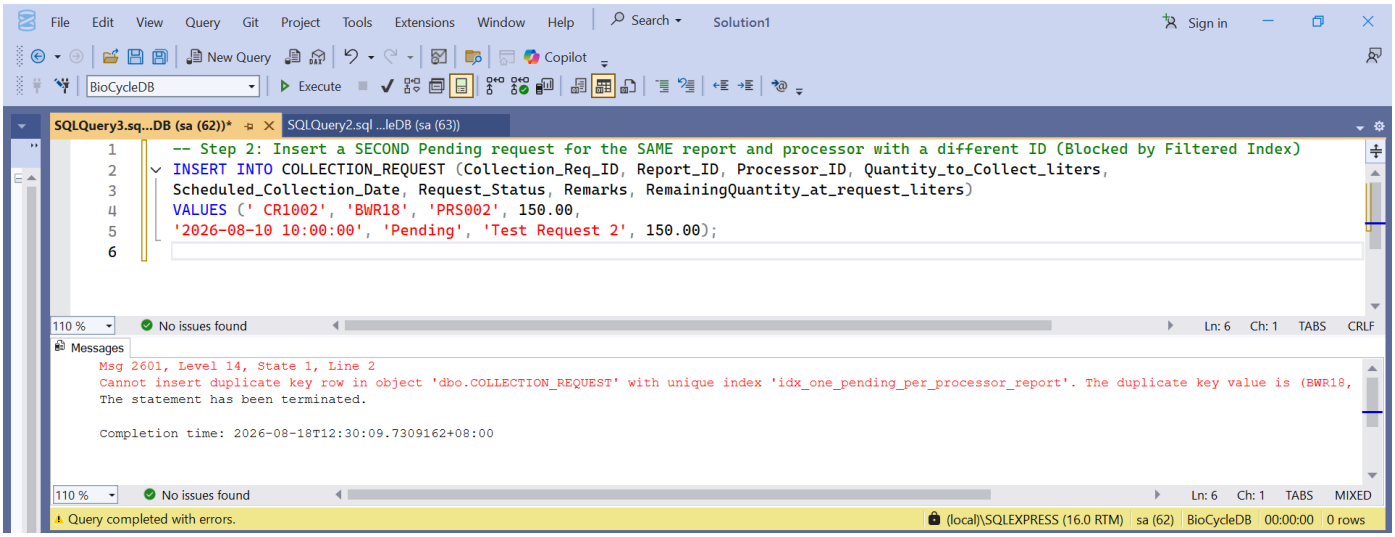

Fig 6. TC-03 SQL Validation Result for COLLECTION_REQUEST UNIQUE Constraint

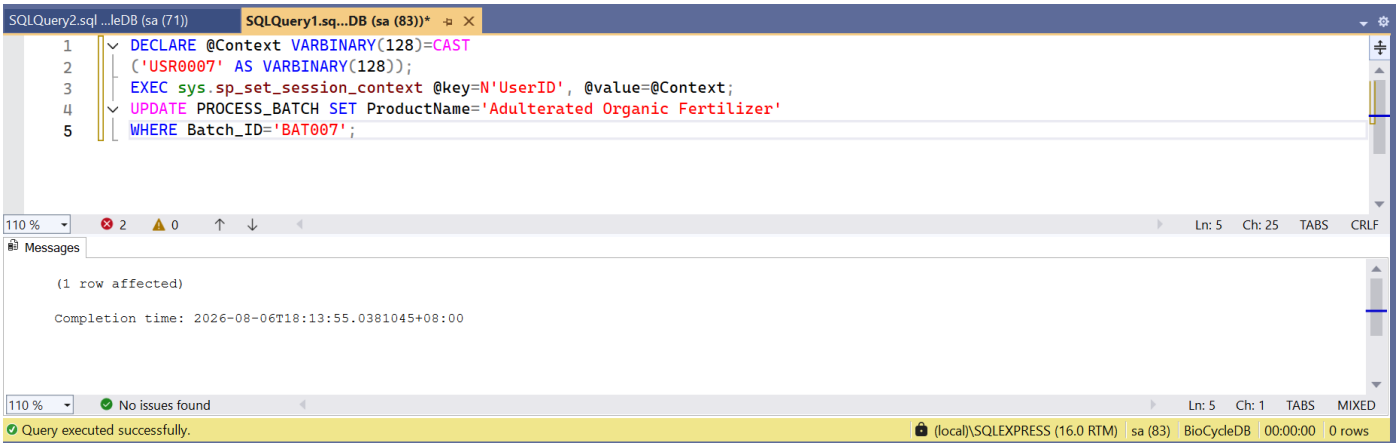

Fig 7. TC-04 SQL Validation Result for PROCESS_BATCH Audit Trigger

Tamper-Evident Auditing via Session Context

The Audit Trail module allows SuperAdmin to monitor overall activities in the system. Audit Logs include Login Audit, Blood Waste Audit, Collection Audit, Process Batch Audit and Product Audit. Each audit logs contains the repective informations performed by which user and their details on what they did to ensure security and transparency across BioCycle system. This audit trail trigger trg_ProcessBatch_Audit was

evaluated to ensure that all data manipulation actions (INSERT, UPDATE, DELETE) on the PROCESS_BATCH table are captured and logged dynamically. Below Fig. 8 and Fig. 9 are the snippet figure for process batch and Product audit trail modules.

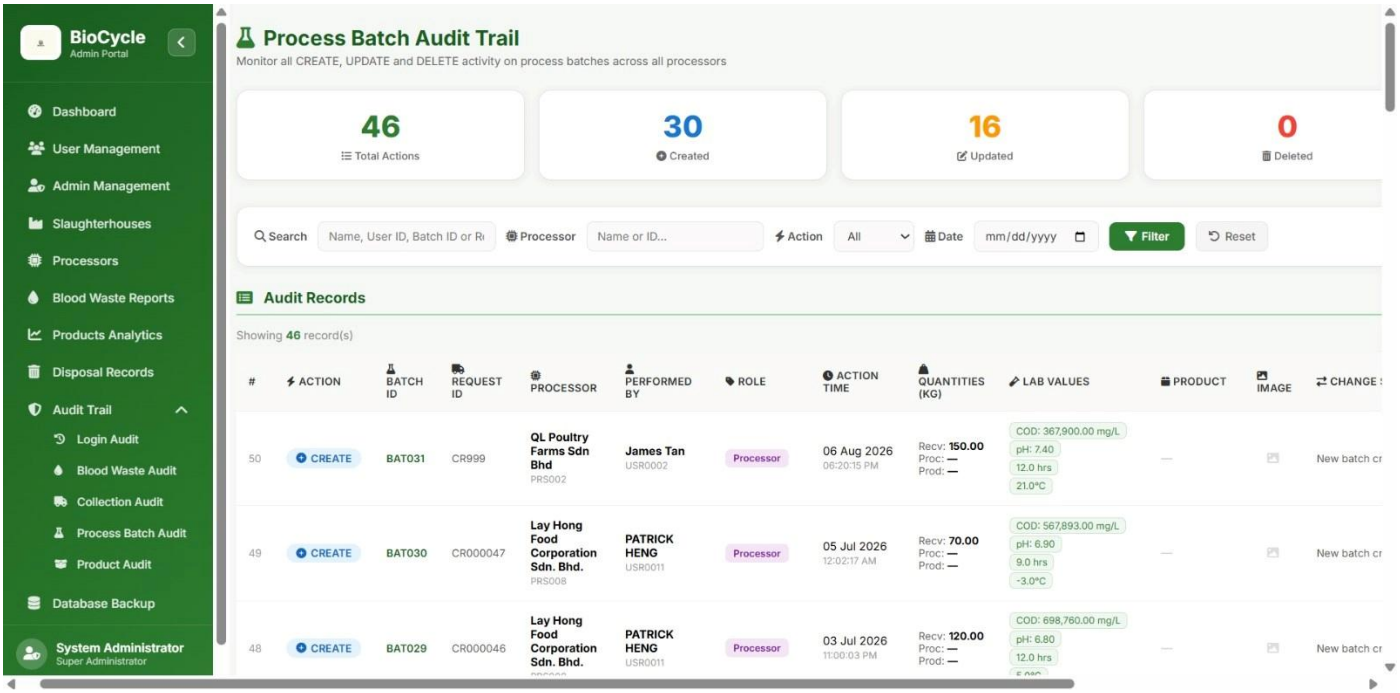

Fig 8. Process Batch Audit Trail Module (SuperAdmin)

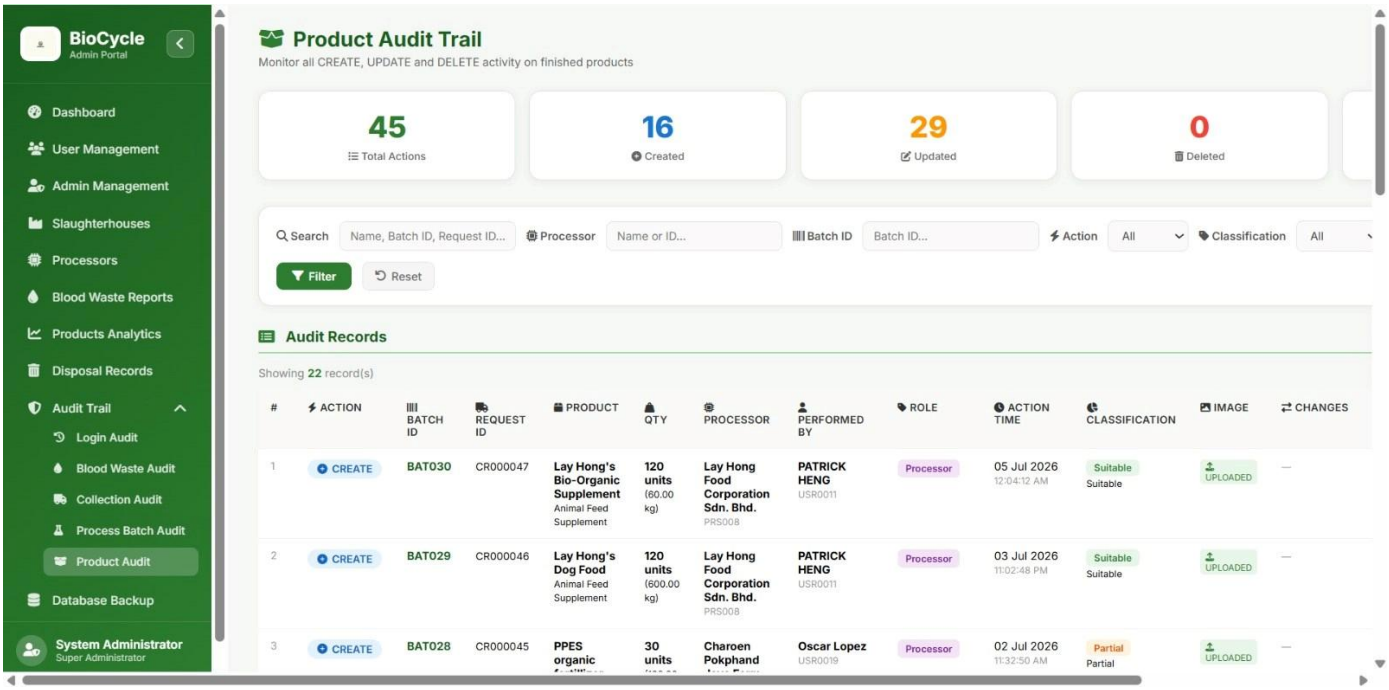

Fig 9. Product Audit Trail Module (SuperAdmin)

Database-Tier Compliance Reporting and Direct Query Validation

Non-functional testing has confirmed the BioCycle-DB schema for its performance characteristics with respect to the DirectQuery reporting workloads. The Power BI dashboard has been validated under the common web conditions with the use of Google Chrome and Microsoft Edge browsers. The tests have confirmed that the normalized 9-entity schema successfully performs complex multi-table join operations, which include PROCESS_BATCH, COLLECTION_REQUEST, BLOODWASTE_REPORT, and USER

tables without any locking, deadlocks, or timeout errors. All aggregated compliance metrics, such as total volume of blood processed (L) and yield efficiency (%), are accurately calculated (refer to the SuperAdmin Product Analytics Dashboard Interface shown in Fig. 10).

Nevertheless, these performance validations can be considered the local ones in the context of a single-instance development environment with localhost and Microsoft SQL Server Express 2022. No systematic benchmarking under multi-user load and stress-testing with huge amount of data and network latency have been quantitatively performed. Thus, scalability and sub-second performance are only theoretical design targets of architecture but not the empirically proven metrics for national production environment.

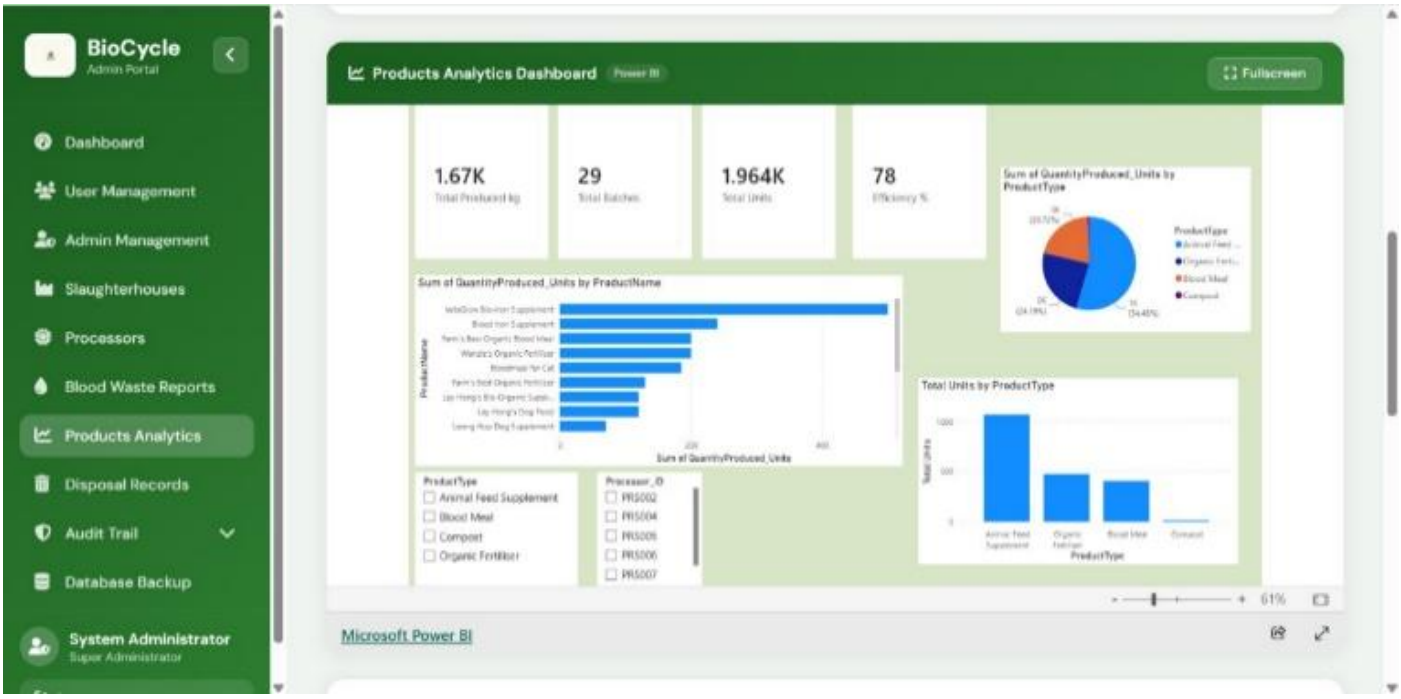

Fig 10. Products Analytics Dashboard Interface (SuperAdmin)

CONCLUSION

This paper presented BioCycle-DB, a high-integrity, normalized relational database architecture engineered utilizing the Database Life Cycle (DBLC) model to manage the complex spatio-temporal logistics and compliance auditing of perishable poultry blood waste. By offloading critical business and compliance rules from vulnerable application-tier codes to the database layer (MS SQL Server Express 2022), the system eliminates data-bypass vulnerabilities and ensures robust referential security. Conceptual specialization-generalization mapping safely isolates stakeholder user accounts, while physical stored procedures (sp_ProcessExpiredReports) successfully automate 24-hour storage expirations and execute volume-dependent biohazard disposal routing.

Furthermore, transactional collisions are mathematically prevented via a filtered unique index (idx_one_pending_per_processor_report), and supply chain auditing is enforced via context-aware triggers (trg_ProcessBatch_Audit) that automatically compile field-level changes. The evaluation successfully validated that BioCycle-DB provides secure database-tier, secure tracing of agricultural by-products, directly supporting national environmental mandates under Malaysia's Environmental Quality Act (EQA) 1974 while promoting sustainable resource utilization in line with Sustainable Development Goals (SDG) 12 and 15.

FUTURE WORK

While BioCycle-DB provides a highly secure and optimized transactional foundation for localized waste tracking, future developments can expand its capabilities:

Distributed Database Cluster and Replication: To support national-scale agricultural waste management, future research will explore migrating the single-server schema into a distributed, multi-tenant cloud-replicated SQL Server cluster across various states in Malaysia, ensuring high availability and geographical load balancing.

Blockchain-DBMS Ledger Integration: To completely eliminate internal threat vulnerabilities (such as malicious system administrators tampering with audit logs directly), the database trigger engine can be integrated with a decentralized, permissioned blockchain ledger, archiving the captured product_audit_log as cryptographically secured blocks to facilitate unalterable regulatory compliance audits.

REFERENCES

1. Hakim, L. I., Isa, N. M. M., & Tahir, S. M. (2020). Effect of halal and non-halal slaughtering methods on bacterial contamination of poultry meat. Sains Malaysiana, 49(8), 1947-1950.
2. Kiepper, B. H., Merka, W. C., & Fletcher, D. L. (2008). Proximate composition of poultry processing wastewater particulate matter from broiler slaughter plants. Poultry science, 87(8), 1633-1636.
3. Liang XiaoNa, L. X., Ye Qing, Y. Q., Wu Shang, W. S., Wu ShangYi, W. S., Kang ShiMo, K. S., Guan BoYuan, G. B., ... & Yue XiQing, Y. X. (2018). Fertilizer efficiency of a new type of chicken blood liquid fertilizer.
4. Currim, F., & Ram, S. (2012). Modeling spatial and temporal set-based constraints during conceptual database design. Information Systems Research, 23(1), 109-128.
5. Gupta, A. K., Fadzlillah, N. A., Sukri, S. J. M., Adediran, O. A., Rather, M. A., Naik, B., & Rustagi, S. (2024). Slaughterhouse blood: A state-of-the-art review on transforming by-products into valuable nutritional resources and the role of circular economy. Food Bioscience, 61, 104644.
6. Kamble, S. S., Gunasekaran, A., & Gawankar, S. A. (2020). Achieving sustainable performance in a data-driven agriculture supply chain: A review for research and applications. International journal of production economics, 219, 179-194.
7. Rejeb, A., Keogh, J. G., Zailani, S., Treiblmaier, H., & Rejeb, K. (2020). Blockchain technology in the food industry: A review of potentials, challenges and future research directions. Logistics, 4(4), 27.
8. Laws of Malaysia, Environmental Quality Act 1974 (Act 127). Kuala Lumpur: Percetakan Nasional Malaysia Berhad, 2006.
9. Astill, J., Dara, R. A., Campbell, M., Farber, J. M., Fraser, E. D., Sharif, S., & Yada, R. Y. (2019). Transparency in food supply chains: A review of enabling technology solutions. Trends in Food Science & Technology, 91, 240-247.
10. Bustillo-Lecompte, C. F., & Mehrvar, M. (2015). Slaughterhouse wastewater characteristics, treatment, and management in the meat processing industry: A review on trends and advances. Journal of environmental management, 161, 287-302.
11. Mozhiarasi, V., & Natarajan, T. S. (2025). Slaughterhouse and poultry wastes: management practices, feedstocks for renewable energy production, and recovery of value added products. Biomass conversion and biorefinery, 15(2), 1705-1728.
12. Bułkowska, K., & Zielińska, M. (2024). Recovery of biogas and other valuable bioproducts from livestock blood waste: a review. Energies, 17(23), 5873.
13. Coronel, C., Morris, S., & Rob, P. (2019). Database systems: design, implementation, and management. Cengage Learning.
14. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). Database system concepts (Vol. 5). New York: McGraw-Hill.
15. Hoffer, J. A., Ramesh, V., Topi, H., & Hilol, B. (2016). Modern database management (Vol. 4, No. 1). London, UK: Pearson.
16. Connolly, T. M., Begg, C. E., & Strachan, A. D. (2005). Database systems: a practical approach to design, implementation, and management (Vol. 3). Harlow: Addison-Wesley.
17. Flores, D. A., & Jhumka, A. (2017, August). Implementing chain of custody requirements in database audit records for forensic purposes. In 2017 IEEE Trustcom/BigDataSE/ICESS (pp. 675-682). IEEE.
18. Wei, Z., Leck, U., & Link, S. (2019). Discovery and ranking of embedded uniqueness constraints.

Proceedings of the VLDB Endowment, 12(13), 2339-2352.

19. Edström, M., Nordberg, Å., & Thyselius, L. (2003). Anaerobic treatment of animal byproducts from slaughterhouses at laboratory and pilot scale. Applied Biochemistry and Biotechnology, 109(1), 127-138.