Iraqi Journal for Computers and Informatics
Available online: https://ijci.uoitc.edu.iq/index.php/ijci
Volume 52, issue 1. 2026

Print ISSN: 2313-190X, Online ISSN: 2520-4912

Ensuring data protection within multi-tenant databases built on cloud infrastructure

Suadad Layth Akram AL- Yawer
Al-iraqia University Baghdad, Iraq
suadad.l.akram@aliraqia.edu.iq

ARTICLEINFO

Keywords:

Cloud Computing; Multi-Tenant Databases; Cloud-DBMS (CDBMS); Database Security; Data Protection; Software as a Service (SaaS);

ABSTRACT

Cloud computing has swiftly become a prominent trend in the technology industry, owing to its various benefits such as scalability, high availability, and cost-effectiveness. Consequently, an increasing number of organizations are shifting their operations to cloud-based environments. It is anticipated that cloud technologies will soon form a foundational component of nearly every enterprise.

This transformation has significantly influenced sectors like Software as a Service (SaaS), where traditional Database Management Systems (DBMSs) have transitioned into Cloud-based DBMSs (CDBMSs). Alongside this evolution, there has been a movement from conventional single-tenant database structures to multi-tenant architectures, which allow for more efficient utilization of infrastructure and resources.

However, despite these benefits, many organizations are still hesitant to adopt multi-tenant database systems due to persistent concerns over data security. The practice of storing data from multiple tenants on the same server—or even within shared tables—introduces considerable risks related to unauthorized access. In response to these concerns, the current research centers on database security in cloud settings, with a specific focus on the unique challenges associated with multi-tenant systems.

1. Introduction

Cloud computing which is now a required part of 90 percent of organizations and projected to be a 1.25 trillion market by 2028 provides scalability, cost-efficiency, and low infrastructure requirements [1–5]. Cloud Database Management Systems (CDBMSs) has become a major ingredient, with more than half of the entire database revenue being created and a scalable data service being made possible through outsourcing [6,7]. SaaS applications Multi-tenancy enables sharing of infrastructure with logical data segregation, each with a security implication, with methods such as separate databases, schema or even shared tables [810].

Multi-tenancy is however, very dangerous to security. Only tenants should be able to access their data, but 76% of organizations are worried about data security, separation between tenants and transparency of CSP [11,12]. The dynamic, distributed world of cloud systems further complicates the issue of security, requiring powerful isolation and access policies [12]. One of the problems is that there are no automated mechanisms to consider security in the early design stages; developers usually postpone with security plan until After attacks [13,14].

This study is based on the application layer of multi-tenant SaaS databases, which lacks automated tools that incorporate any safety concerns at the beginning of the design stage is viewed as a significant gap. This supposition states that by using an automated CASE tool, including the concept of multi-tenant security in the initial stages, a safer and more efficient protection

E-mail address: suadad.l.akram@aliraqia.edu.iq Corresponding* : *Suadad Layth Akram AL- Yawer*
Received 22th September 2025, Accepted 25th November 2025

DOI: 10.25195/ijci.v52i1660.

can be provided. The goals are as follows: to find vulnerabilities in CDBMS, to define security requirements, to use structured diagrams, to create CRUD matrices, to apply a role-based and partitioned access, and to automate SQL to create tenant-specific configurations. The research is focused on application-layer data-at-rest and tenant-level security, without considering transmission, inter-application flows, and high-level regulatory issues, such as GDPR.

2. Literature Review

This section defines the conceptual and theoretical background of the research by looking at the main issues of the research, including cloud computing, multi-tenancy, and database systems, as applied to security. It contains definitions, contextual reflections, and a critical literature review to back the methodology and analysis used in the study, putting the research into the current academic discussion.

2.1. Cloud Computing

According to the definitions of National Institute of Standards and Technology, cloud computing has revolutionized IT using the idea of scalability, flexibility and cost efficiency [1518], which enable quick access to a resource that is shared by many people with minimum management.

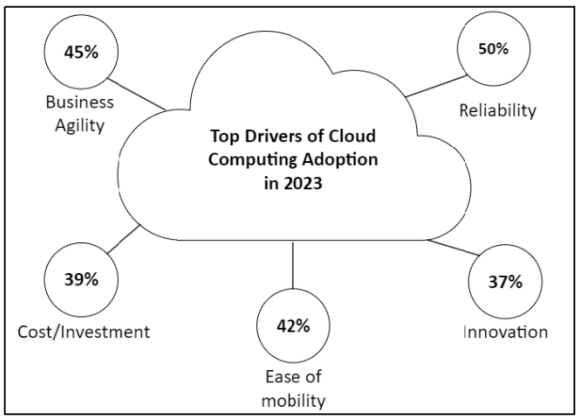

Fig. 1. Top Drivers for Cloud Adoption [21].

[19]. The services provided by such high-profile providers as Amazon, Oracle, IBM, and Microsoft are available worldwide [20]. They are also key features such as on-demand self-service, wide-access, resources pooling, elasticity and billing by usage, which facilitate dynamic multi-tenant environments [19,21]. Nevertheless, there is still a worry on the security, downtime, unknown costs and regulation compliance because of offsite data storage over common infrastructure [2,22]. Models such as StaaS and DBaaS do have data-specific functionality [25-27], whereas the SaaS, PaaS and IaaS models differ in the extent of user control [23-24]. With the shift of organizations to cloud based systems, instead of the traditional DBMS, this migration necessitates reconsideration of the data protection strategy, particularly in the context of the multi-tenant architecture.

2.2. DBMS and PostgreSQL

Modern information systems rely on databases, which are efficient in storing and manipulating of massive data to assist the decision making process [28]. The majority of databases have tabular structure and manipulation by the use of SQL [28,29]. Database Management System (DBMS) is a system that serves as an interface between users or programs and the underlying databases and it carries out activities that include storage and retrieval of data as well as data modification. It also controls such important issues as data integrity, schema organization, indexing, and recovery procedures [2931]. The leading DBMSs, such as Oracle, PostgreSQL, MySQL, and Microsoft SQL server are compatible in the cloud environment [29].

PostgreSQL is a strong open source DBMS that supports both relational and object model with a client server architecture compatible with other systems [32]. It supports simultaneous connectivity to clients and has superior security options including views, trigger, and customization options. PostgreSQL has mandatory, discretionary and role-based access control and supports Row-Level Security (RLS) controls over fine-grained access control [32–34], and is compatible with clouds.

2.3. CDBMS and Multi-Tenancy

Cloud Database Management Systems (CDBMS), which is commonly made available as a service through SaaS, is displacing the traditional DBMSs because of its scalability, cost efficiency, availability and fault-tolerance [30,35,36]. A CDBMS has storage (to encrypt and back up data/objects), database (to store data/Objects), and application layers (to authenticate/report data)

[37]. Multi-tenant setups improve performance and sharing through load balancers and connection pools and guarantee data redundancy using remote backups [35,37].

User Management and Access Control
Users are assigned roles with permissions managed through GRANT and REVOKE commands [36,38–40]. Authentication uses username-password methods [33,41].
Database Operations and Views
Operations (SELECT, INSERT, UPDATE, DELETE) depend on role-based permissions. Data is organized in schemas and tables [42–44]. Views manage access to specific data subsets and enforce policies, though not all are updateable [32,45].

2.4. Stored Procedures and Backup

Stored procedures/functions encapsulate logic, enhancing security and preventing SQL injection [34,46–48]. Backup mechanisms support recovery and rollback to stable system states [49].

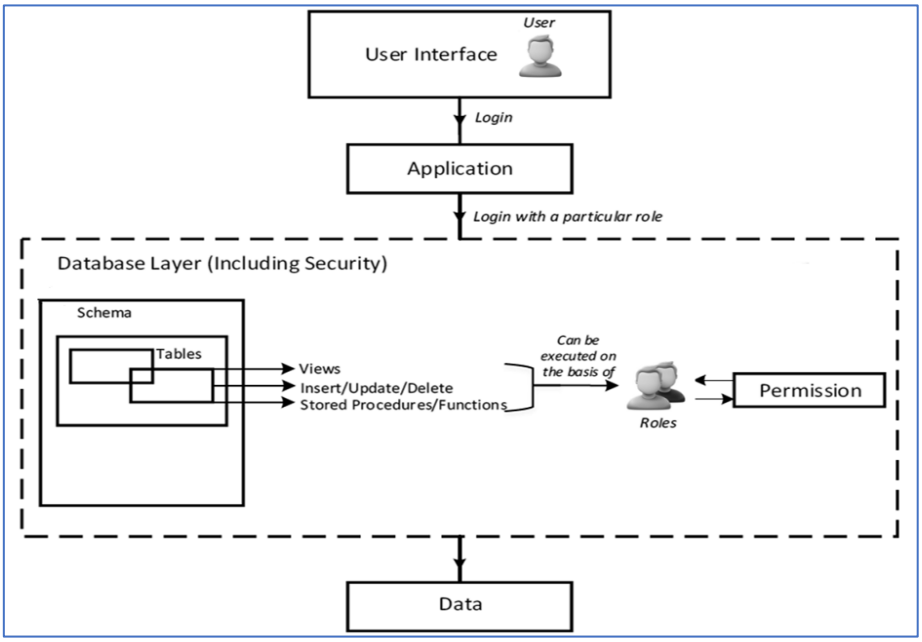

Fig. 2. Architecture of Cloud-Based Database Management System (CDBMS). Adapted from [37].

2.5. Security Factors in Multi-Tenant

The problem of security is a significant obstacle on the way to the adoption of multi-tenant cloud-based DBMSs, regardless of the advantages that it has [49]. Security issues, especially those related to data distributed on different hardware and tenants [37], [70] are that the local DBMS is still preferred by many users. Contrary to on-premise systems where the data and security are handled internally, cloud environments are dependent on external Cloud Service Providers.

for security enforcement [71].

Multi-tenancy will make security more complex, as it will have more users and will have different access requirements. The balancing and co-existence of such differing needs on a common infrastructure creates serious difficulties in the confidentiality of data, integrity, and compliance.

2.6. Access Controls

The main approach to reducing the security risks of multi-tenant cloud systems is access controls, which control user permissions and privileges [95]. They provide granular access based on vertical and horizontal partitioning and limits users to particular access.

data, respectively. Another widespread way to demonstrate the access permissions is the CRUD matrix, where the access control rights are given to the users or classes, assigned to every system object, through Create, Read, Update, and Delete [96].

The traditional models of access control systems are the Discretionary Access Control (DAC), Mandatory Access Control (MAC) and the advanced models of Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) [95].

DAC grants the access on the basis of user identity and is frequently used with the Access Control Lists though not devoid of vulnerabilities such as Trojan horse attacks and necessitating constant maintenance [97]-[100].

MAC, conversely, applies access control via system-defined security labels and adheres to the write up, read down model. Although it provides greater security, it is not flexible and granular [95], [101], [102].

RBAC resolves multi-tenancy issues by granting roles rather than users permissions, which is endorsed by the principle of least privilege. It makes access control management simplified and particularly is apt to dynamic settings such as cloud systems [103]- [105].

ABAC provides access to attributes such as user-related, resource-related, and environment-related attributes. These attributes are used to define policies, which are very flexible in complicated situations [106].

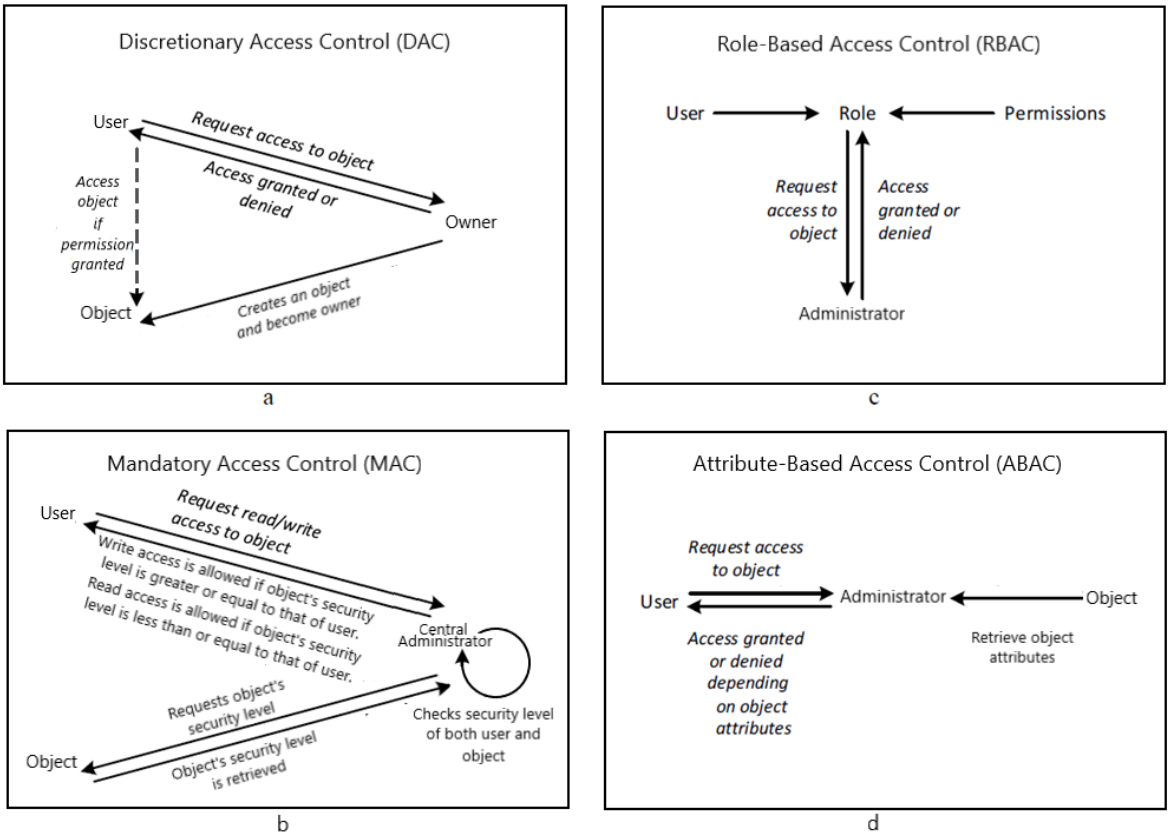

Fig. 3. illustrates the differences among assesad [107].

The models of security are compared by Pernul [108] and Hasani and Modiri [110] in terms of the complexity of administration, constraint support, delegation and implementation, authentication and granularity of authorization [109]-[111]. The models to be used depend on the needs and threat environments in the organization.

2.7. Security Models

Access controls are used as part of the security models to create protection frameworks. According to Younis et al. [112], threat context is another feature that is important when selecting models, particularly in multi-tenant clouds. The vulnerability of the systems to effective model choice is also emphasized by Meghanathan [104]. There are four essential models which are outlined:

Bell–La Padula Model:

- Focused on confidentiality using hierarchical labels (e.g., "Top Secret"), with rules:
- Simple Security Property – read at or below level.
- Star Property – write at or above level.
- Discretionary Security Property – uses access control matrix [110], [113–115].
- Tied to MAC, it prevents unauthorized access but lacks integrity assurance [110].

BIBA Integrity Model:

- Aims to preserve data integrity with inverse rules to Bell–La Padula:
- No Read Down – read from equal/higher integrity.
- No Write Up – write to equal/lower integrity.
- Invocation Rule – restricts calls to higher integrity [116,117].

Clark–Wilson Model:

- Maintains system integrity through:
- Well-formed Transactions.
- Access Control Policies.
- Separation of Duties [110,118].
- Includes audits and authentication; used in banking systems.

Chinese Wall Model:

- Combines MAC and DAC to manage confidentiality and integrity, preventing conflicts of interest, particularly in consulting and accounting [119,120].

3. Requirements and Analysis

The requirements analysis was to identify and categorize as well as validate security requirements of secure multi-tenant cloud database systems. Through the methodology of Kitchenham [126], the requirements were elicited by reviewing the literature, analysis of features of cloud products, and threat assessment.

The requirements were grouped into base, functional, and non-functional requirements. Base requirements were design diagrams (ERM, ELH, DFD) usage, and access controls were consistent, as well as code reviews, open standards, and configuration extensions [75,137,151]. The functional requirements were dedicated to confidentiality, integrity, and availability, as well as non-repudiation, mitigation of threats, auditing, and accountabilities [80,137,147,151,152,155]. Non-functional requirements dealt with performance [153,156], tenant isolation [80,137,147] and coping with the complexity of security-systems [80,137].

The choice of PostgreSQL v15 was predetermined by the fact that this database is cloud-compatible, open-source, and has inherent security [159]. It addresses such critical requirements as parameterized querying to prevent SQL injection [166], other authentication [167], role-based authorization [159], encryption (pgcrypto) [170], backup support [172], and auditing support audit tables/triggers. PostgreSQL was closer to the specified requirements, specifically in authentication, backup and auditing compared to AWS [160] and Oracle [161].

3.1. Access Control Design

Access control was found to be a key requirement. The framework uses Role-Based Access Control (RBAC), which is the most ubiquitous within cloud systems [182] and assigns permissions to roles rather than users as a method of access control.

RBAC was favored as opposed to DAC and MAC because it has a balance between control and flexibility. RBAC is more flexible as it allows scalability, separation of duties, and the least privilege, unlike DAC, which is prone to misconfiguration, and MAC, which is not flexible [52]. Multi-tenant environments In PostgreSQL, users can be granted specific permissions, which ensures the security of access by roles.

The report is concluded by stating that PostgreSQL, which boasts of good security and the availability of RBAC, happens to be the most appropriate DBMS to use in the proposed secure multi-tenant cloud framework.

4. System Design Overview and Visual Representations

4.1 Introduction

A design phase is essential in transforming the theoretical ideas into practical application to give a flawless framework to achieve the objectives of the research. This section defines the main aspects of building the proposed CASE tool that is supposed to contribute to the creation of the automated, secure, and multi-tenant databases, which has to be situated on the cloud. Of special concern is how system design diagrams are used in defining and validating tenant-specific security assertions. Some of this content has been published before in the 4th International Conference on Information Systems and management as

"Incorporating Security Features in System Design Documents used with Cloud-Based Databases" [179].

4.2 General Design of the Proposed CASE Tool

The general architecture represented in Figure 4 helps developers to build secure multi-tenant databases, as the system automates the process of managing the access permissions, sustaining consistency, and providing continuous feedback on the design process in order to improve. It begins with the input three system diagrams are considered (ERM, DFD, and multiple ELHs) formatted into JSON and analyzed to determine any missing or weak areas.

Access control mechanisms are then defined using an automated creation of a CRUD matrix, which is extended to a three dimensional access control model. This model has row and column views, and permissions where only authorised transactions take place through validation checks.

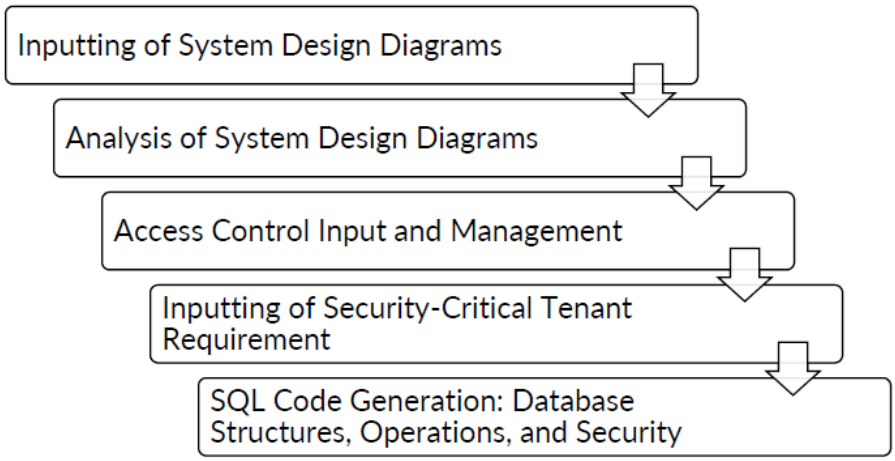

Fig. 4. Proposed CASE Tool Workflow

The users are able to create tenant preferences like encryption. The secure SQL code is produced after feedback addressing is repeated, which is in accordance with the security requirements such as SQL injection protection. The process of the tool minimizes the human error by automating the provision of tenant-level security assertion. Section 7 illustrates a practical example of banking system, and the system (Scott) [180].

4.3 System Design Diagrams

As illustrated in Figure 3, the system's foundation lies in three primary diagrams:

1. **Entity Relationship Management (ERM):** Illustrates the database tables along with their relationships.
2. **Entity Life History (ELH):** Describes all potential states and transitions of entities.
3. **Data Flow Diagram (DFD):** Depicts the flow of information among system components.

These diagrams are instrumental for incorporating security at the design stage. They support the direct mapping of privileges and access rights, requiring a balance between generic and specific annotations for usability and security.

Subsequent sections explain the construction, required components, and format for each diagram to ensure proper implementation.

4.4 Assumptions

Key assumptions in the design of the CASE tool are as follows:

- Only binary and ternary relationships are considered in the ERM [181].
- Each system is represented by a single DFD, one ERM, and multiple ELHs.
- ELH processes are arranged in sequential order.
- Partial keys of weak entities in the ERM act as primary keys.
- ERM primary keys are simple, non-derived, and single-valued [182].
- For entities without an ELH, a basic version includes insert, update, and delete operations.

4.5 Encoding and Representation of Design Diagrams

The syntax for ERM, ELH, and DFD diagrams is defined by three sets of EBNF rules. JSON format is used for encoding (.erm,

.elh, .dfd) due to its compatibility with cloud environments [183][184].

4.5.1 Encoding of ERM

Thirty-eight EBNF rules regulate the encoding of ERM. The structure starts with entity names and attributes, followed by weak entities, and then relationships, which are categorized into five types. Each relationship contains names, attributes, involved

participation levels, and cardinalities. A partial JSON example of the 'Scott' ERM is shown in Figure 5.

```
{
 "name": "Scott",
 "strong entities": [{
   "strong entity name": "Department",
   "attributes": [{
     "attribute name": "departmentNum",
     "attribute type": "integer",
     "required": "yes",
     "attribute mode":["non-derived", "single", "primary key simple"]
   }, ......
   ]}],
 "SE-SR-SE relationships": [{
   "relationship name": "Works in",
   "strong participating entity 1": {
     "strong participating entity name": "Employee",
     "participation in relationship": "total",
     "cardinality": "M"
   },
   "strong participating entity 2": {
     "strong participating entity name": "Department",
     "participation in relationship": "partial",
     "cardinality": "1"
   }}, ......
 ], ......
}
```

Figure 5. Partial JSON Encoding of the 'Scott' ERM.

```
{"name": "Scott",
"entity": "Department",
"processes": [
 {"processname": "newDepartment",
  "option": "none"
 },
 {"processname": "modifyDepartmentInformation",
  "option": "none"
 },
 {"processname": "closeDepartment",
  "option": "none"
 }
]}
```

Figure 6. JSON Encoding of the 'Scott' ELH for the 'Department' Table

4.5.2 Encoding of an ELH

Each entity in the ERM has a corresponding ELH that lists the sequential processes impacting the entity's states. Five EBNF rules define the entity names, processes, and whether these processes are iterative or selective. Figure 6 illustrates an ELH for the 'Department' entity, where all processes are designated as non-iterative.

4.5.3 Encoding of a DFD

The encoding of a DFD consists of eleven rules that describe entities, data stores, processes, and data flows. Data flows

indicate their direction as well as the types of connected components. Figure 7 presents an example of a DFD JSON encoding for the 'Scott' system.

```
{
 "name": "Scott",
 "external entities": [
  {"external entity name": "Manager"}, ......
  ],
 "data stores": [
  {"data store name": "Department"}, ......
  ],
 "processes": [
  {"process name": "closeDepartment"}, ......
  }
 ],
 "dataflows": [
  {"dataflow name": "Department Information",
   "inward connected entity": "closeDepartment",
   "inward entity type": "process",
   "outward connected entity": "Manager",
   "outward entity type": "external entity" }, ......
 ]}
```

Fig. 7. JSON encoding for 'Scott' DFD

4.6 Designing the Conversion from System Diagrams to Data Dictionary

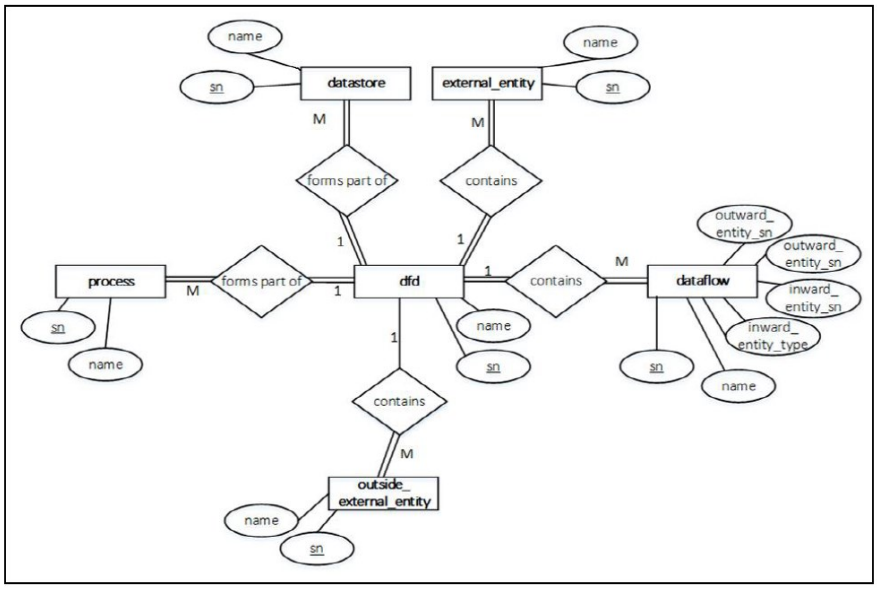

Fig. 8. Relations Employed for the Relational Storage of ELH Diagrams

To enable secure SQL generation, diagrams are transformed into relational structures. Initially, JSON files are stored in the 'import_elh', 'import_erm', and 'import_dfd' tables. Following validation using the 'is_json' function, the data is moved into JSON

columns and then broken down into relational tuples.

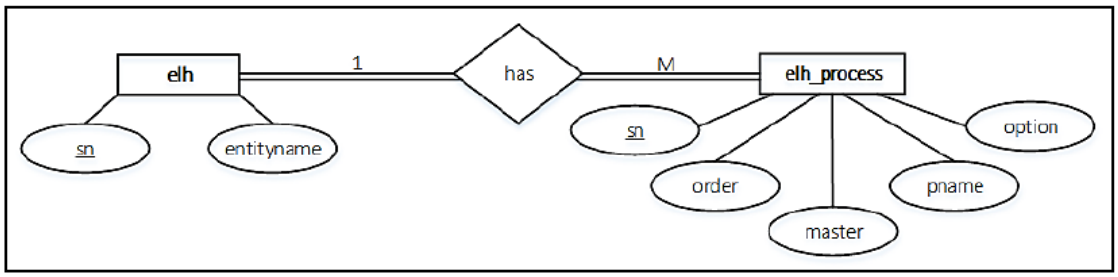

Fig. 9. Relations Utilized for the Relational Storage of DFD Diagrams

Figures 7 to 9 illustrate relations for storing ELH, DFD, and ERM components, respectively. Each table captures components with unique attributes. Pseudocode 1 and 2 detail the transformation of ELH JSON into relational tuples, including recursive processing of sub-processes.

Pseudocode 1. Transformation of ELH Diagrams into Relational Data Structures

```
1:   Input: sysdiag_name text
2:   Output: void
3:
4:   Algorithm:
5:   If sysdiag_name is NOT NULL then
6:     sysDiag ←SELECT sysdiag_sn
7:                 FROM systemdiagrams
8:                 WHERE sysdiag_name = sysdiag_nm
9:     If sysDiag == 0 then
10:    For elh_curs in
11:      SELECT ie_serno, ie_json → entity
12:      FROM import_elh
13:      WHERE ie_sysdiag_serno = sysDiag
14:    Do
15:      elh ← SELECT ie_json
16:     FROM import_elh
17:      WHERE ie_serno = elh_curs.ie_serno
18:                                          >get ELH JSON to convert into predicates
19:      INSERT INTO elh
20:      VALUES elh_curs.iename, elh_curs.ie_sysdiag_serno
21:      RETURNING elh_e_sn INTO current_e_sn
22:
23:      idx ← json_array_length(elh → processes)      >extract process subtree, if any
24:      If idx == 0 then
25:      elh proc ← function json_extract(elh, processes)
26:      p_rtn ← function elh_import2predicates_processes(current_e_sn, null, proc)
27:      End if
28:    End for
29:    End if  End if
```

Pseudocode 2. Transformation of ELH Processes into Relational Data Structures

```
Input: elh_sn integer, elh_master integer, json_fragment JSON
Output: void

Algorithm:
For rec_curs in
 SELECT t.proc → processname, t.proc → option, t.proc → processes,
    json_array_length(t.proc→ processes)
 FROM json_fragment → processes AS t(proc)
 Do

  INSERT INTO elh_process
  VALUES (rec_curs.key, elh_sn, elh_master, rec_curs.pn, rec_curs.opt)
  RETURNING elh_p_sn INTO current_p_sn

  If rec_curs.nsubp → 0 then
   p_trn←function elh_import2predicates_processes(elh_sn, current_p_sn,
   processlist)                                    >Recursive calling of function
  End if
  End for
```

DFD transformation involves more tables, with Pseudocode 3 describing the mapping of dataflows. Each component is converted into tuples with appropriate entity-type associations.

Pseudocode 3. Transformation of DFD to Relational Data Structures (Dataflow Segment)

```
Input: sysdiag_name text
Output: void

Algorithm:                                   >extract the dataflows fragment
 dfcounter ← json_array_length(dfd → dataflows)
  If dfcounter == 0 then
   dfd_extent_df ← function json_extract(dfd, dataflows)
   For rec_curs in
  t.rel -» dataflow_name, t.rel -» inward_entity_type,
   CASE t.rel -» inward_entity_type
    external_entity then
   SELECT dfd_ee_sn
    FROM external_entity
    WHERE dfd_ee_name = t.rel-»'inward connected entity'
  AND dfd_ee_h_sn = current_d_sn)
   data_store then
        SELECT dfd_d_sn
   FROM data_store
   WHERE dfd_ee_name = t.rel -» 'inward connected entity'
   AND dfd_d_h_sn = current_d_sn
       process then
        SELECT dfd_p_sn
   FROM process
   WHERE dfd_p_name = t.rel -» 'inward connected entity'
   AND dfd_p_h_sn = current_d_sn)
      END CASE,
      t.rel -» 'outward entity type' ,
      CASE t.rel -» outward entity type ...
      END CASE,
      FROM dfd_extent_df→'dataflows' AS t(rel)

   Do                                        >insert each dataflow as a tuple
   INSERT INTO dfd_dataflow
   VALUES (current _d_sn, rec_curs.dfn, rec_curs.iet, rec_curs.iesn,
    rec_curs.oet, rec_curs.oesn)
   RETURNING dfd_df_sn INTO current_df_sn
  End for
   End if
```

ERM transformation follows a similar approach, requiring the most tables. Entities and attributes are processed, and boolean

values are converted appropriately. Relationship definitions are transformed as shown in Pseudocode 4.

Pseudocode 4. Transformation of 'SE-SR-SE' Relationships into Relational Data Structures

```
1:  Algorithm:
2:   rc ← json_array_length(json_fragment → SE-SR-SE rel)
3:   If rc > 0 then
4:   erm_extent_sss ← function json_extract(json_fragment, SE-SR-SE rel)
5:   For rec_curs_sss in
6:   (erm_extent_sss → SE-SR-SE rel)
7:   Do
8:   INSERT INTO erm_relationship
9:   VALUES (erm_sn, rec_curs_sss.rn, 'strong')
10:  RETURNING erm_r_sn INTO current_r_sn
11:
12:  INSERT INTO erm_part_entity
13:  VALUES (current_r_sn, rec_curs_sss.particEnt, rec_curs_sss.cardEnt,
14:   rec_curs_sss.partEnt)
15:  RETURNING erm_pe_sn INTO current_pe_sn
16:  For rec_curs_rel_att in
17:   SELECT rel→'relationship attribute' -» 'attribute name', rel →
18:   'relationship attribute' -» 'attribute type'
19:   FROM erm_extent_sss → SE-SR-SE rel
20:   WHERE rel → 'relationship attribute' -» 'attribute name' IS NOT NULL
21:   AND rel -» 'relationship name' = current_r_sn.name
22:  Do
23:       INSERT INTO erm_attribute_relationship
24:   VALUES (current_r_sn, rec_curs_rel_att.attName, rec_curs_rel_att.attType)
25:   RETURNING erm_ra_sn INTO current_ra_sn
26:  End for
27:  End for
28:  End if
```

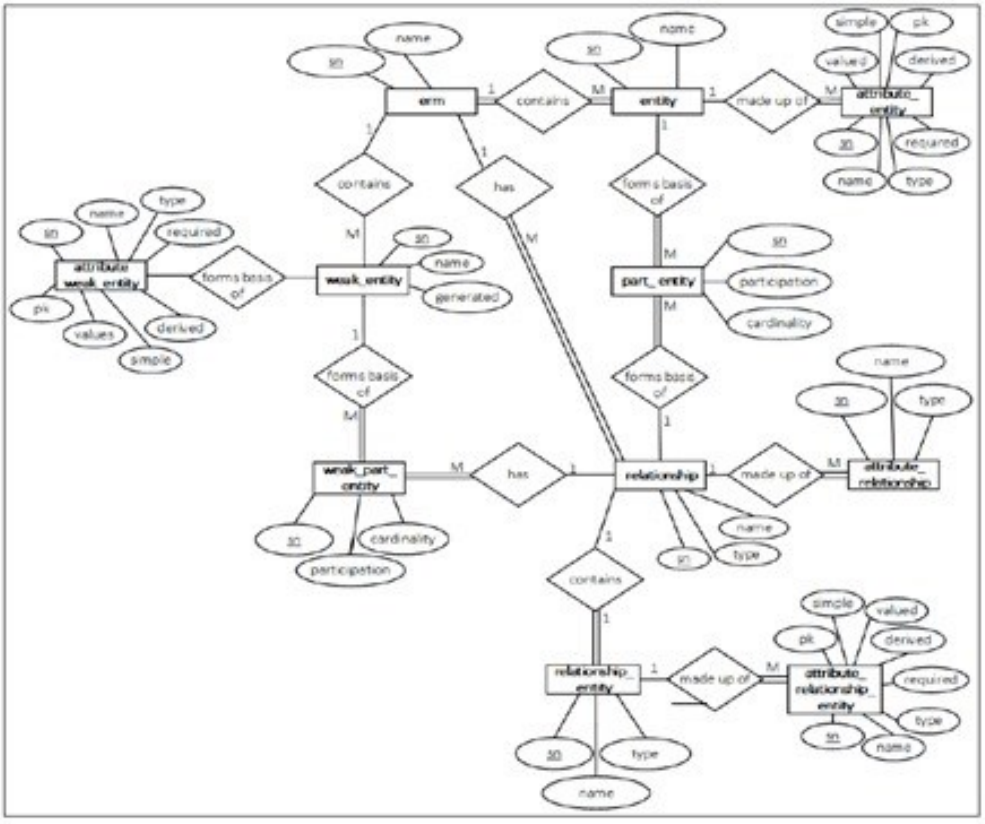

Fig. 10. Relations Used for the Relational Storage of ERM Diagrams

4.7 System Diagrams Validation Suite

A set of checks was created to assess the diagrams submitted to the CASE tool, functioning as an initial evaluation of

system adequacy prior to implementing security features. These checks are documented in PreAndPostChecks.txt, with the outcomes presented in a table that includes the check name, data violating the check, and the status labeled as 'Good', 'Bad', or 'Warning'.

4.7.1 Preliminary Checks on Individual System Diagrams

4.7.1.1 ELH Checks

- *Minimum Number of Processes:* ELH must have ≥2 processes; fewer triggers 'Warning'.
- *No Duplicate Processes:* Duplicates return 'Bad' with names.
- *Single ELH per Entity:* Entities must have only one ELH; otherwise, 'Bad'.

4.7.1.2 ERM Checks

These six points represent the **Logical Consistency Rules** or an **Automated Checking Mechanism** typically applied by database modeling software (Tools) to a designed **Entity-Relationship Model (ERM)**.

- *Entity Consistency in Relationship Section:* Entities in relationships must exist in the entity section; else 'Bad'.
- *Recursive Relationship Participation:* Must include one partial participation; if not, 'Warning'.
- *Ternary Relationships and Strong Entities:* Only strong entities may participate.
- *Entity Connected to a Relationship:* Unconnected entities return 'Warning'.
- *Weak Entity Linked to Strong Entity:* Required link; failure results in 'Bad' with entity name.
- *Unique Entity and Attribute Names:* Enforced via syntactic constraints.

4.7.1.3 DFD Checks

DFD Checks are a set of validation rules applied to Data Flow Diagrams to ensure the model is **structurally and logically correct**, and free from errors that might hinder system development or understanding.

- *Input/Output Consistency:* I/O must match entities, processes, or data stores.
- *Process Participation in Dataflows:* Each dataflow must involve a process; invalid ones return 'Bad'.
- *I/O Requirement for Processes and Data Stores:* Each must have at least one input and output.
- *External Entity Linked to a Dataflow:* Must participate in at least one dataflow.
- *Leaf and Non-leaf Node Connections:* Leaf nodes connect to data stores; non-leaf must not.
- *External Entities Linked to Parent Processes Only:* Only allowed with top-level processes.
- *Intermediate Processes Connected to Processes Only:* Must connect exclusively to other processes.
- *Uniqueness of Components and Dataflows:* All components and identifiers must be unique.

4.7.2 Validation Checks on Combined System Diagrams (Post-checks)

Due to element overlap among ELH, ERM, and DFD, post-checks ensure cross-diagram consistency (Figure 11):

- *ELH Processes Must Exist in DFD:* Required for traceability; exceptions allowed for non-altering processes.
- *Entity Presence in Both ERM and DFD:* All entities must appear as data stores in DFD and vice versa.
- *Each ERM Entity Requires an ELH:* Missing ELHs are auto-created using *elh_any* with Insert, Update, and Delete processes.

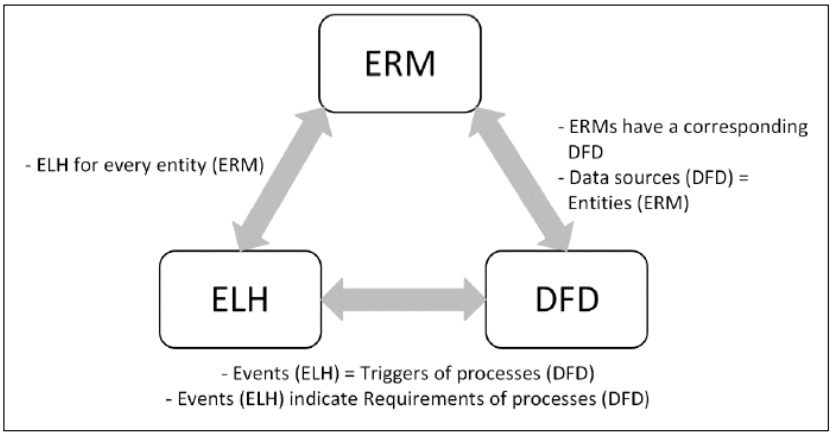

Fig. 11. Inter-relationships between ERM, ELH, DFD

4.8 Setup Design Concerning Other Key Users

Even though CASE tool primarily concentrates on tenant-specific arrangements, the fundamental setups of other functions were also established and they included Cloud Service Provider (CSP) as well as the software provider as described in.

The CSP environment has a privileged access that has been granted authorization to a separate database, and all other permissions have been removed to provide maximum security. This database has extensive contents which are the subject of this

dissertation.

The software provider setup includes a dedicated database that has three schemas namely softwareproviderdba, softwareproviderdeveloper and softwareprovidersecurity. Only specific roles are authorized to access the database namely; DBa and developer, with authorization to the schema privileges depending on the level of role.

In order to support the operations of the tool, necessary database objects were created to support ERMs, ELHs, DFDs, user specific tables and functions, CRUD operations, and role management. These objects were well placed in the specified schemas.

5. Evaluation

5.1 Introduction

The system coverage, performance and security evaluation were carried out to ascertain how the CASE tool matched the objectives in Section 1. The evaluation of strengths, limitations, and applicability of the tools was done through a mixed qualitative and quantitative approach.

5.2 System Coverage Evaluation

The research evaluated the capability of DFD, ERM, and ELH diagrams to support the secure multi-tenant system design. Depending upon the SSADM and literature of SB by Professor Sterman [162] and Haimes [163]:

- *Functional:* DFDs model entities, dataflows, and processes.
- *Structural:* ERMs model entities and relationships.
- *Time-based:* ELHs capture lifecycle transitions.

This combination matches SSADM's holistic system definition. ERMs are well-suited but may lack advanced modeling (e.g., aggregation). ELHs and DFDs support rule validation but not time-dependent triggers. UML diagrams, including Class and Sequence Diagrams, offer alternatives, but Use Case Diagrams lack data store representation [164][165]. Table 1 compares these diagrams, concluding that DFD, ERM, and ELH provide effective, low-overhead modeling aligned with database needs.

Table 1. Comparison Between System Diagrams and UML-Based Diagrams

System Diagrams Used for Proposed Tool	***Equivalent UML Diagrams***	***Main Distinctions***
ERM Class Diagrams	Class Diagrams	Entity-Relationship Models (ERMs) are designed to illustrate entities and the associations between them within a database system, whereas class diagrams are mainly used to represent classes and the relationships among them within a software application.
ELH	Sequence Diagrams	The Entity Life History (ELH) diagram illustrates the sequence of processes that cause changes in the state of a database entity over time, while a sequence diagram primarily maps out the chronological order of interactions or events that occur within a system.
DFD	Use Case Diagrams	Use case diagrams do not incorporate data stores, which limits their ability to provide a comprehensive overview of a system. As a result, they are more appropriate for early-stage conceptualization. In contrast, Data Flow Diagrams (DFDs) present a more complete depiction of system processes and data movements, making them more suitable for formal modeling purposes.

5.3 Performance Evaluation

Under the different workloads, both in the local and the cloud environment, quantitative tests were performed. The local test was done with an 8 GB RAM, 256 GB SSD, 2.7 GHz Core i7 CPU; the cloud tests were executed on Google Cloud server of Europe-north1 and Asia-east2 with the same configurations (Table 2) and using PostgreSQL and pg-bench [176].

This variability was identified as a result of sampling bias, network conditions and cloud properties as there was benchmarking of secured and non-secured systems.

Table 2. Google Cloud Evaluation Criteria

Region	*Operating System*	*CPU*	*RAM*	*Disk Type*	*Location*
Europe	Windows Server 2016	2 vCPUs @ 2.7 GHz	8 GB	Zonal SSD	europe-north1
Asia	Windows Server 2018	2 vCPUs @ 2.7 GHz	8 GB	Regional SSD	asia-east2

5.3.1 Establishing a Benchmark Baseline

Unsecured systems (Scott, Banking System and School Management System) were all tested with 100,000 CRUDs. According to Bermbach et al. [192], local tests excluded deployment time (Figure 12). Benchmarks were run ten times at the client loads of different numbers, and average latency and throughput were measured. Result in three models, namely, sd (separate database), ss (separate schema), and st (same table), are provided in Figure 13 and Figure 14.

These results are important conclusions obtained as a result of a performance study between the various database architecture in a multi-tenant system (where a single application/infrastructure is shared amongst a number of customers, or tenants). In this study, it compares isolated and shared database models specifically.

- The separate database model consistently showed the lowest latency and highest throughput, attributed to the absence of schema filtering.
- Latency increased as the number of clients grew; throughput also rose due to parallel processing.
- The schema and table models showed improved performance relative to separate databases as the number of tenants increased.

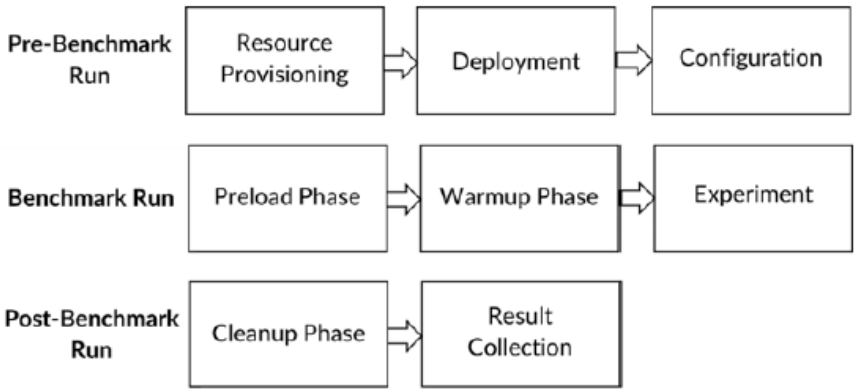

Fig.12. Benchmark Experimental Setup and Procedure [192]

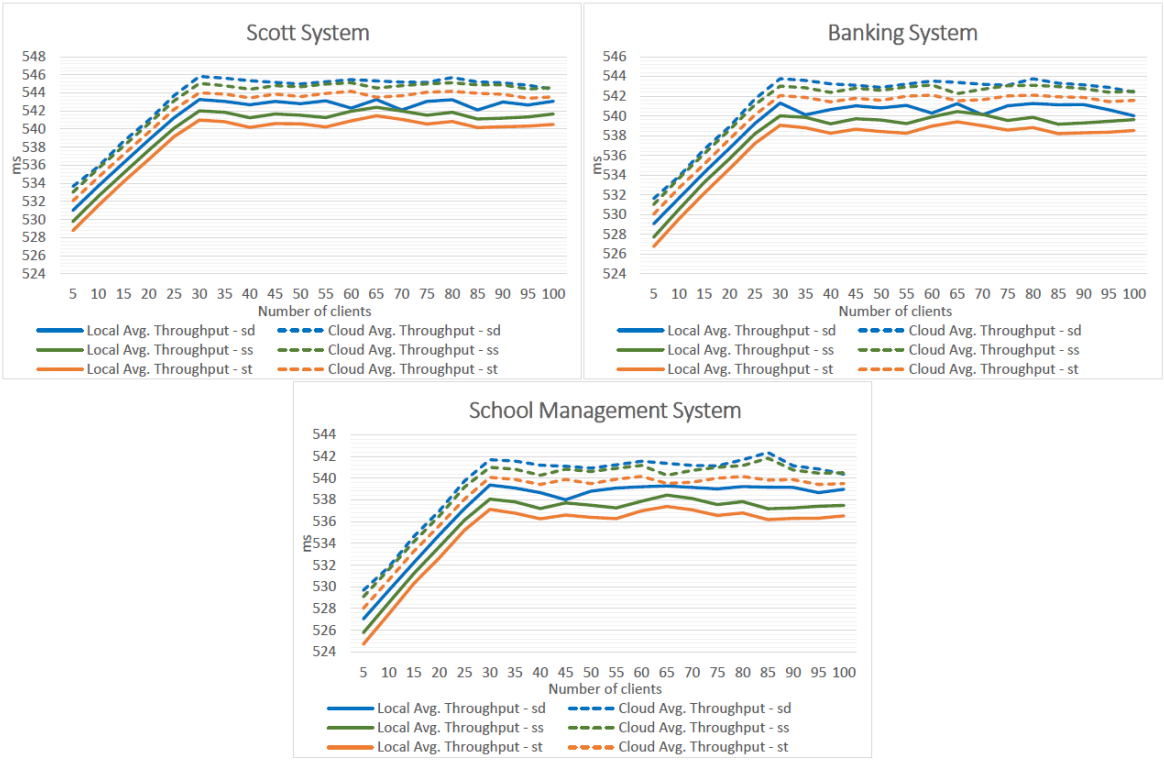

Fig. 13. Benchmark Baseline – Average Throughput

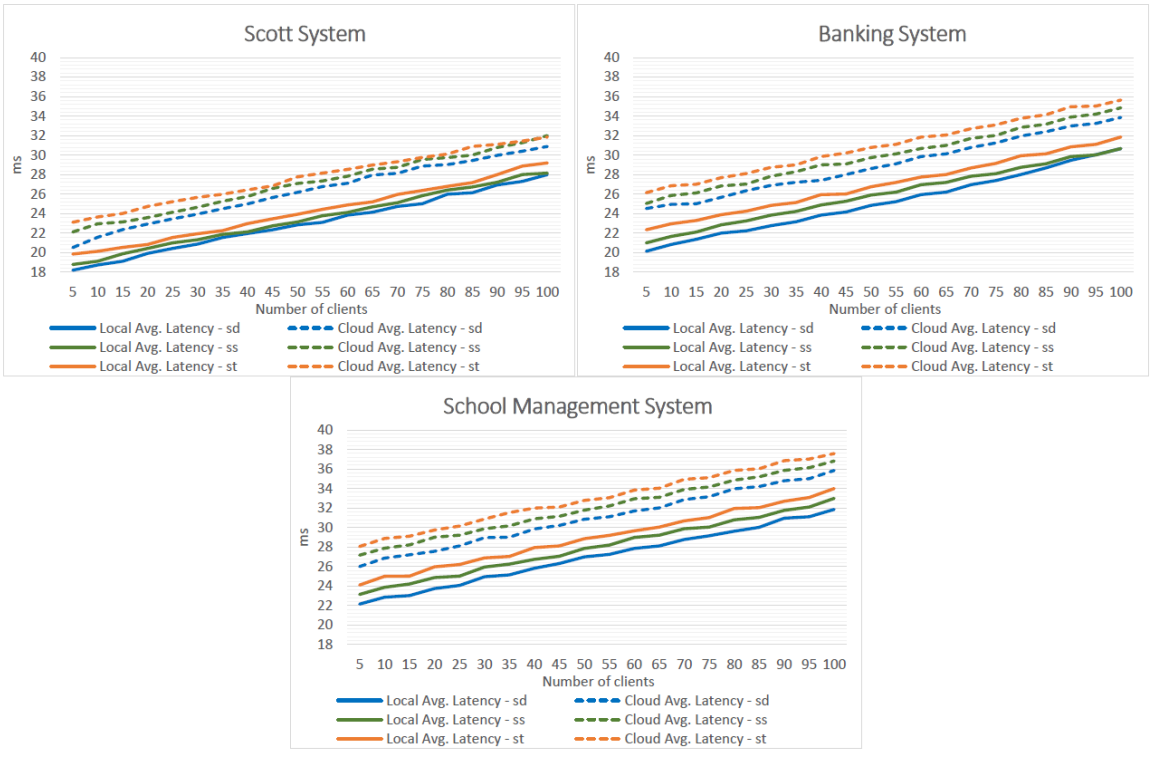

Fig. 14. Benchmark Baseline – Average Latency

5.3.2 Performance Benchmark with Security

Security features were incorporated in subsequent tests.

- The maximum latency increase observed was 3.066 ms (in the 'Scott' system with 100 clients).
- The maximum throughput decrease recorded was 2.195 requests per second (in the 'School Management System').

These few modifications show the effectiveness of the tool. The School Management System had the worst latency and worst throughput since it consisted of more roles, processes and tables which added data volume and checks.

Out of the multi-tenant models, separate databases provided the best performance since the security was not as complicated, and the schema and table models had slightly longer execution times because they took more security measures.

5.4 Security Evaluation

In this section, the ability of the proposed CASE tool to produce secure and tenant-specific code of multi tenant database is evaluated. With the framework proposed by Vieira et al. [193], where the level of database security is assessed over a scale ranging between 0 and 100, the maximum possible score in this respect is 44 through the limitations of the scope. SQL scripts were used to test the system with the help of the AppDetectivePRO tool [194].

5.4.1 Security Evaluation Framework

Vieira et al. [193] classify the level of DBMS security into the requirements that are weighted. Four mechanisms (User Authentication, User Privileges/Access Rights, Encryption, and Auditing) were the focus of the evaluation. As the requirements were met full or not, scores were assigned either as full or zero. The criteria and their weightings are shown in table 3.

5.4.1.1 User Authentication

In the case of 1.1, users used individual credentials. In the case of 1.2, the encrypted password table was only accessible to tenant DBAs, which was tested by restricted access.

5.4.1.2 User Privileges / Access Rights

The tool employs role-based access control along with an automatically generated CRUD matrix.

- Only DBAs are permitted to assign or revoke roles.
- Operations supported include insert, update, delete, and select.
- Horizontal and vertical partitioning is implemented through row- and column-level security.
- This approach failed the test because tenant data owners were granted admin rights—an intentional choice to allow greater flexibility.

5.4.1.3 Encryption

was satisfied: passwords were encrypted by default using PostgreSQL's MD5 [195].

5.4.1.4 Auditing

was met: operations were logged with all required data accessible to tenant DBAs.

Table 3. Results

Security Mechanism	*Tests Conducted*	*Test Outcome*	*Weighting*
1. User Authentication	Credential, encryption, access tests	Pass	10, 4, 6
2. User Privileges / Access	Role control, CRUD, partitioning	Pass	6, 6, 4
	Exclusive DBA rights	Fail	0
3. Encryption	Password encryption	Pass	4
4. Auditing	Logging verification	Pass	4

Final Score: 40/44.

Only 2.4 failed due to design choice favoring tenant-level control. Cloud DBA remains isolated, confirming strong overall security.

5.5 Evaluation Scripts

A suite of SQL scripts was developed to assess the security of the proposed system, with each script returning a boolean value based on expected outcomes.

5.5.1 Edge Cases

1. If a user has credentials but no tenant ID, the system denies access (Pseudocode 5).
2. If a user provides a tenant ID but is unassigned to it, access is also denied.
3. If a user is assigned to multiple tenants, access is restricted to the current tenant only.

Pseudocode 5. Edge Case Evaluation Script

```
Input: username text, pass text, tenant_id integer
Output: TABLE (obj_name text, obj_type text, priv text, has_access boolean

Algorithm:
If EXISTS                                                    > authenticate user
  SELECT *
  FROM pg_auth
 WHERE rolname = username && rolpassword = md5(pass)) then
                                                  > Set the current 'tenant ID' to NULL
 set_config(app.current_tenant_id, tenant_id, true)

Create Temp Table (access_results ← Columns (objnm, objtype, priv, hasaccess)

For object in
   SELECT table_name, table_schema, 'table/view'
    FROM information_schema.tables
   WHERE rolename = username
Do
  For each priv in ['SELECT', 'INSERT', 'UPDATE', 'DELETE']:
   hasaccess ← has_table_privilege(object.table_schema, object.table_name,
     priv, username)
   access_results ← Columns (objnm, objtype, priv, hasaccess)
   Values (object.table_name, object.object_type, priv, hasaccess)
                                                          > Check access to tables
End For

For object in Columns (p.proname, n.nspname, 'function') Table
(pg_catalog.pg_proc, pg_catalog.pg_namespace)
Do
   hasaccess ← has_function_privilege(object.specific_schema,
   object.routine_name, object.argument_types)
   access_results ← Columns (objnm, objtype, priv, hasaccess)
   Values (object.routine_name, object.object_type, 'EXECUTE', hasaccess)
                                                       > Check access to functions
End For
return access_results
```

5.5.2 Authorised Users

For valid credentials and tenant ID:

a. *Tables/Views*: A function compares actual vs. expected privileges using symmetric difference (Pseudocode 6).

Pseudocode 6. Privileges associated with Tables/Views Evaluation Script

```
1: Input: systemid integer, tenantid integer, dbarole text
2: Output: boolean
3:
4: Algorithm:
5: Create Temp Table cruddata(rlname text, tblname text, tblVw text, perm text)
6:                                                    > retrieve privileges from CRUD
7: cruddata ← SELECT rolename, tablename, tableorview, perm
8:   FROM crud
8:   WHERE (sysdiag = systemid && tenid = tenantid)
9:  UNION                                            > retrieve RLS policies (partitions)
10:  SELECT rolename, tablename, 'RLS', columnname
11:   FROM policies
12:  WHERE (sysdiag = systemid && tenid = tenantid)
13:                                                  > retrieve actual privileges
14: actualprivileges ← SELECT grantee, table_name, table_type, privilege_type
15:   FROM role_table_grants
16:    WHERE grantee != dbarole
17:    UNION                                         > retrieve actual policies
18:   SELECT roles, tablename, 'RLS', columnname
19:   FROM role_table_grants
20:    WHERE grantee != dbarole
21: If Symmetric difference (cruddata, actualprivileges) IS NOT NULL then
23:  return false
24: Else return true
```

b. *Stored Procedures/Functions*: Privileges are similarly verified (Pseudocode 7).

Pseudocode 7. Privileges associated with Stored Procedures Evaluation Script

```
1:  Input: systemid integer, tenantid integer, dbarole text
2:  Output: boolean
3:
4:  Algorithm:
5:  Create Temp Table cruddata(rlname text, functionname text, perm text)
6:
7:  cruddata ← Columns (rolename, functionname, perm) Table (crud)
8:   Condition (sysdiag = systemid && tenid = tenantid)
9:                                                  > retrieve privileges from CRUD
10: actualprivileges ← Columns (grantee, table_name, table_type, privilege_type)
11:     Table (role_table_grants) Condition (grantee != dbarole)
12:                                                  > retrieve actual privileges
13:
14: If Symmetric difference (cruddata, actualprivileges) IS NOT NULL then
15:  return false
16: Else
17:  return true
```

c. *Data Operations*: SELECT, INSERT, DELETE, and UPDATE scripts verify CRUD outcomes (Pseudocode 8).

Pseudocode 8. SELECT Operation Evaluation Script

```
1:  Input: schemaname text, tablename text, expectedResult type[]
2:  Output: boolean
3:
4:  Algorithm:
5:  queryText ← SELECT *
6:    FROM schemaname.tablename
7:
8:  If queryText = expectedResult then
9:   return true
10: Else
11:  return false
```

Tenant DBAs are tested for extended privileges, which matched predefined DBA rights.

Software/Cloud providers are limited to designated schemas; excess access triggers warnings.

5.5.3 PUBLIC Role

The default PUBLIC role was tested on all databases to verify that it does not have any access to objects (Pseudocode 9) which was verified.

Pseudocode 9. PUBLIC Role Evaluation Script

```
Input: void
Output: TABLE (typ text, objname name, privtype text)

Algorithm:
resultSet ← SELECT 'database', databasename, privilegetype
    FROM pg_database
   WHERE rolname = PUBLIC
                                        >extract database privileges
resultSet ← SELECT 'schema', schemaname, privilegetype
    FROM pg_namespace
   WHERE rolname = PUBLIC
                                        >extract schema privileges
resultSet ← SELECT 'function', functionname, privilegetype
    FROM role_routine_grants
   WHERE grantee = PUBLIC && functionschema != pg_catalog
                                        >extract function privileges
resultSet ← SELECT 'table', tablename, privilegetype
    FROM role_table_grants
    WHERE grantee = PUBLIC && tableschema != pg_catalog
                                        >extract table privileges
return resultSet
```

5.6 AppDetectivePRO

5.6.1 Potential Security Vulnerabilities Report

AppDetectivePRO [194] found no critical risks such as SQL injections or unauthorized access and reported security to PostgreSQL patches and measures. There was categorization of risk levels and no critical findings were realized.

5.6.2 User Rights Privilege Report

The report was used to match privileges with the CRUD matrix and ensure minimal and correct access.

- *Separate database*: Tenant isolation confirmed.
- *Separate schema*: One schema per tenant enforced.
- *Same table*: RLS policies ensure isolation.
- Policy-based privileges were validated in the findings report.

6. Conclusion

It achieved its main goal, enabling the development of an automated CASE tool whereby the multi-tenant cloud security is integrated beginning with the design of the database. It considers the above sections and outlines the principal contributions and future research directions.

6.1 Summary of Sections

- **Section 1**: Introduced multi-tenancy as a cost-effective strategy, noting security as the main challenge.
- **Section 2**: Reviewed cloud systems, multi-tenancy models (separate databases/schemas/tables), access control, and CASE tools.
- **Section 3**: Identified comprehensive security requirements from literature and vendors.
- **Section 4**: Explained tool design using ERM, ELH, and DFD diagrams to create tenant-specific security assertions, covering user roles and early vulnerability checks.
- **Section 5**: Evaluated performance and security. Qualitative and quantitative benchmarks showed minimal overhead (e.g., 3.066ms latency, 2.195 r/s variance). Security evaluation included a framework (40/44 score), evaluation scripts, and AppDetectivePro results.

6.2 Objective Alignment and Contributions

The discussed the absence of security integration in multi-tenant systems in the early stages, which achieved the goals of Section 1.

Key Contributions:

1. Established a full set of base, functional and non functional security requirements.
2. Integrate security into architecture through ERM, ELH and DFD, provides architectural representation [146].
3. Generated EBNF syntax and data dictionaries of diagrams, a security checklist.
4. Supported partitioning stratagems and RBAC supported by automated security matrices (CRUD, roles-processes) extraction.
5. Created a CASE tool that produces secure, tenant-specific SQL code and was tested on three case studies (Scott, Banking, School Management) by evaluating it using a mixed-method approach, namely security, performance, and continuity.

References

[1] Pallathadka, H., et al. (2022). An investigation of various applications and related challenges in cloud computing. *Materials Today: Proceedings, 51*, 2245–2248.

[2] Beri, R., & Behal, V. (2015). Cloud computing: A survey on cloud computing. *International Journal of Computer Applications, 111*(16).

[3] Wood, L. (2023, May 15). *Global cloud security market trends, opportunities, and forecasts, 2018–2023 & 2024–2028F*. Research and Markets.

[4] Lambert Consulting. (2023). *Statistics and trends in cloud computing 2023–2024*. https://www.lambertconsulting.ch/en/statistiques-et-tendances-du-cloud-computing-2023-2024/

[5] Jamsa, K. (2022). *Cloud computing* (2nd ed.). Jones & Bartlett Learning.

[6] Shareef, T. H., Sharif, K. H., & Rashid, B. N. (2022). A survey of comparison different cloud database performance: SQL and NoSQL. *Passer Journal of Basic and Applied Sciences, 4*(1), 45–57.

[7] Kashinath, V. K. (2023). *Cloud services market research, 2031*. Allied Market Research.

[8] Chong, F., Carraro, G., & Wolter, R. (2006). Multi-tenant data architecture. *MSDN Library*, Microsoft Corporation, 14–30.

[9] Narasayya, V., & Chaudhuri, S. (2022). Multi-tenant cloud data services: State-of-the-art, challenges and opportunities. In *Proceedings of the 2022 International Conference on Management of Data*.

[10] Mallick, A., & Rathore, R. K. (2015). Survey on database design for SaaS cloud application. *International Journal of Computer Engineering & Technology (IJCET), 6*(6). https://doi.org/JCET_06_06_007

[11] Check Point Software Technologies Ltd. (2023). Cloud security threats remain rampant: Check Point survey reveals heightened concerns for 76% of organizations amid 48% increase in cloud-based network attacks. https://www.checkpoint.com/press-releases/cloud-security-threats-remain-rampant-check-point-survey-revealsheightened-concerns-for-76-of-organizations-amid-48-increase-in-cloud-basednetwork-attacks/

[12] Jangjou, M., & Sohrabi, M. K. (2022). A comprehensive survey on security challenges in different network layers in cloud computing. *Archives of Computational Methods in Engineering, 29*(6), 3587–3608.

[13] Stallings, W. (2015). *Computer security: Principles and practice*.

[14] Kousalya, A., & Baik, N. (2023). Enhance cloud security and effectiveness using improved RSA-based RBAC with XACML technique. *International Journal of Intelligent Networks, 4*, 62–67.

[15] Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: A brief review. *International Journal of Computer Sciences and Engineering, 7*(2), 421–426.

[16] Jani, K., Kumar, B., & Shah, H. (2013). Degree of multi-tenancy and its database for cloud computing. *International Journal of Engineering Development and Research, 3*, 168–171.

[17] Lee, S. (2012). Database management system as a cloud service. *International Journal of Future Generation Communication and Networking, 5*(2).

[18] Garg, T., Kumar, R., & Singh, J. (2013). A way to cloud computing basic to multitenant environment. *International Journal of Advanced Research in Computer and Communication Engineering, 2*(6), 2394–2399.

[19] Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*.

[20] Alshinwan, M., et al. (2023). Integrated cloud computing and blockchain systems: A review. *7*(2), 941–956.

[21] Jayaraman, S. (2023). 160+ fascinating cloud computing statistics for 2023. https://www.g2.com/articles/cloud-computing-statistics

[22] Hassan, J., et al. (2022). The rise of cloud computing: Data protection, privacy, and open research challenges—A systematic literature review (SLR), *8*.

[23] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *34*(1), 1–11.

[24] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*.

[25] Bagaeen, A., et al. (2019). Storage as a service (StaaS) security challenges and solutions in cloud computing environment: An evaluation review. In *2019 Sixth HCT Information Technology Trends (ITT)*.

[26] Roy, S., Pattnaik, P. K., & Mall, R. (2016). A cognitive approach for evaluating the usability of Storage as a Service in cloud computing environment. *6*(2).

[27] Kansal, M., et al. (2023). A systematic study of services and security model in cloud computing: A brief overview, 1–16.

[28] Ning, Z., et al. (2020). Distributed and dynamic service placement in pervasive edge computing networks. *IEEE Transactions on Parallel and Distributed Systems, 32*(6), 1277–1292.

[29] Amadin, F., & Obienu, A. C. (2017). Multi-tenancy approach: An emerging paradigm for database consolidation. *8*(1).

[30] Karataş, G., et al. (2017). Multi-tenant architectures in the cloud: A systematic mapping study. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*.

[31] Simitos, A. (2016). Multi-tenant databases for SaaS—Security and privacy issues, 4–9.

[32] Jacobs, D., & Aulbach, S. (2007). Ruminations on multi-tenant databases.

[33] Abdul, A. O., et al. (2017). A performance evaluation of multi-tenant data tier design patterns in a containerized environment. In *2017 International Conference on Information Society (i-Society)*.

[34] Gulati, S. S., & Gupta, S. (2012). A framework for enhancing security and performance in multi-tenant applications. *5*(2), 233–237.

[35] Heng, L., Dan, Y., & Xiaohong, Z. (2012). Survey on multi-tenant data architecture for SaaS. *9*(6), 198.

[36] Januzaj, Y., Ajdari, J., & Selimi, B. (2015). DBMS as a cloud service: Advantages and disadvantages. *195*, 1851–1859.

[37] Fernandez, E. B., Monge, R., & Hashizume, K. (2016). Building a security reference architecture for cloud systems. *21*, 225–249.

[38] Kaufman, L. M. (2009). Data security in the world of cloud computing. *7*(4), 61–64.

[39] Kandukuri, B. R., & Rakshit, A. (2009). Cloud security issues. In *2009 IEEE International Conference on Services Computing*.

[40] Ahmad, M. S., & Bamnote, G. R. (2013). Data leakage detection and data prevention using algorithm. *6*(2), 394–399.

[41] Kanade, S., & Manza, R. (2019). A comprehensive study on multi-tenancy in SaaS applications. *181*(44), 25–27.

[42] Brown, W. J., Anderson, V., & Tan, Q. (2012). Multitenancy—Security risks and countermeasures. In *2012 15th International Conference on Network-Based Information Systems*.

[43] Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences, 305*, 357–383.

[44] Rong, C., Nguyen, S. T., & Jaatun, M. G. (2013). Beyond lightning: A survey on security challenges in cloud computing. *Computers & Electrical Engineering, 39*(1), 47–54.

[45] Hashizume, K., et al. (2013). An analysis of security issues for cloud computing. *4*, 1–13.

[46] Xiao, Z., & Xiao, Y. (2012). Security and privacy in cloud computing. *15*(2), 843–859.

[47] Alouffi, B., et al. (2021). A systematic literature review on cloud computing security: Threats and mitigation strategies. *9*, 57792–57807.

[48] Hubbard, D., & Sutton, M. (2010). *Top threats to cloud computing v1.0*, 1–14.

[49] Islam, T., Manivannan, D., & Zeadally, S. (2016). A classification and characterization of security threats in cloud computing. *7*(1), 268–285.

[50] Kazim, M., & Zhu, S. Y. (2015). A survey on top security threats in cloud computing. *6*(3), 109–113.

[51] Kumar, S. V. K., & Padmapriya, S. (2014). A survey on cloud computing security threats and vulnerabilities. *2*(1), 622–625.

[52] Amalarethinam, G., & Rajakumari, S. (2019). A survey on security challenges in cloud computing. *24*(1), 133–141.

[53] Al Nafea, R., & Almaiah, M. A. (2021). Cyber security threats in cloud: Literature review. In *2021 International Conference on Information Technology (ICIT)*.

[54] Bonasera, W., Chowdhury, M. M., & Latif, S. (2021). Denial of service: A growing underrated threat. In *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*.

[55] Cloud Security Alliance. (2010). *Top threats to cloud computing v1.0* (Vol. 23).

[56] Verma, G., & Adhikari, S. (2020). Cloud computing security issues: A stakeholder's perspective. *1*(6), 329.

[57] Sood, S. K. (2012). A combined approach to ensure data security in cloud computing. *35*(6), 1831–1838.

[58] Irwin, L. (2019). List of data breaches and cyber attacks in May 2019 – 1.39 billion records leaked. https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-may-2019-1-39-billion-records-leaked

[59] Kulkarni, S., & Urolagin, S. (2012). Review of attacks on databases and database security techniques. *2*(11), 253–263.

[60] Mousa, A., Karabatak, M., & Mustafa, T. (2020). Database security threats and challenges. In *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*.

[61] Pallavi, G. B., & Jayarekha, P. (2022). Secure and efficient multi-tenant database management system for cloud computing environment. *14*(2), 703–711.

[62] Qiu, J., et al. (2020). A survey on access control in the age of internet of things. *7*(6), 4682–4696.

[63] Cavique, L., & Cavique, M. (n.d.). Axiomatic design applied to CRUD matrix in information systems. In *IOP Conference Series: Materials Science and Engineering*.

[64] El Sibai, R., et al. (2020). A survey on access control mechanisms for cloud computing. *31*(2), e3720.

[65] Abdi, A. I., et al. (2020). Blockchain platforms and access control classification for IoT systems. *12*(10), 1663.

[66] Wang, Y., et al. (2022). A survey on metaverse: Fundamentals, security, and privacy.

[67] Kashmar, N., Adda, M., & Atieh, M. (2019). From access control models to access control metamodels: A survey. In *Advances in Information and Communication: Proceedings of the 2019 Future of Information and Communication Conference (FICC)* (Vol. 2).

[68] Bertin, E., et al. (2019). Access control in the Internet of Things: A survey of existing approaches and open research questions. *74*, 375–388.

[69] Aluvalu, R., & Muddana, L. (2015). A survey on access control models in cloud computing. In *Emerging ICT for Bridging the Future: Proceedings of the 49th Annual Convention of the Computer Society of India (CSI), Volume 1*.

[70] Soni, K., & Kumar, S. (2019). Comparison of RBAC and ABAC security models for private cloud. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*.

[71] Meghanathan, N. (2013). Review of access control models for cloud computing. *3*(1), 77–85.

[72] Cai, F., et al. (2019). Survey of access control models and technologies for cloud computing. *22*, 6111–6122.

[73] Lo, N. W., Yang, T. C., & Guo, M. H. (2015). An attribute-role based access control mechanism for multi-tenancy cloud environment. *84*, 2119–2134.

[74] Indu, I., Anand, P. R., & Bhaskar, V. (2018). Identity and access management in cloud environment: Mechanisms and challenges. *21*(4), 574–588.

[75] Pernul, G. (1995). Information systems security: Scope, state-of-the-art, and evaluation of techniques. *International Journal of Information Management, 15*(3), 165–180.

[76] Almarhabi, K. (2019). Arbiter: A lightweight, secured and enhanced access control mechanism for cloud computing. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*.

[77] Hasani, S. M., & Modiri, N. (2013). Criteria specifications for the comparison and evaluation of access control models. *5*(5), 19.

[78] Zhang, N. (2005). Generating verified access control policies through model-checking.

[79] Younis, Y. A., Kifayat, K., & Merabti, M. (2014). An access control model for cloud computing. *19*(1), 45–60.

[80] Cristiá, M., & Rossi, G. (2021). Automated proof of Bell–LaPadula security properties. *65*(4), 463–478.

[81] Wurster, G. D. (2010). Security mechanisms and policy for mandatory access control in computer systems.

[82] Danturthi, R. S. (2020). *70 Tips and Tricks for Mastering the CISSP Exam*.

[83] Biba, K. J. (1977). *Integrity consideration for security computer system (Technical Report TR-3153)*.

[84] Toapanta, M., et al. (n.d.). Analysis of the appropriate security models to apply in a distributed architecture. In *IOP Conference Series: Materials Science and Engineering*.

[85] Tsegaye, T., & Flowerday, S. (2020). A Clark-Wilson and ANSI role-based access control model. *28*(3), 373–395.

[86] Brewer, D. F., & Nash, M. J. (n.d.). The Chinese Wall security policy. In *IEEE Symposium on Security and Privacy*.

[87] Parast, F. K., et al. (2022). Cloud computing security: A survey of service-based models. *Computers & Security, 114*, 102580.

[88] Da Silva, M. A., & Danziger, M. (2015). The importance of security requirements elicitation and how to do it.

[89] Lim, S., Henriksson, A., & Zdravkovic, J. (2021). Data-driven requirements elicitation: A systematic literature review. *[Journal name not provided]*, *2*, 1–35.

[90] Tsochev, G. R., & Trifonov, R. I. (n.d.). Cloud computing security requirements: A review. In *IOP Conference Series: Materials Science and Engineering*.

[91] Kumar, R., & Goyal, R. (2019). On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *[Journal name not provided]*, *33*, 1–48.

[92] Irshad, M., Petersen, K., & Poulding, S. (2018). A systematic literature review of software requirements reuse approaches. *[Journal name not provided]*, *93*, 223–245.

[93] Kitchenham, B. (2004). Procedures for performing systematic reviews. *[Journal name not provided]*, *33*, 1–26.

[94] Wohlin, C. (n.d.). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*.

[95] Badampudi, D., Wohlin, C., & Petersen, K. (n.d.). Experiences from using snowballing and database searches in systematic literature studies. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*.

[96] Yousaf, N., et al. (2022). Investigation of latest CASE tools for database engineering: A systematic literature review. In *2022 International Conference on Frontiers of Information Technology (FIT)*.

[97] Patel, S. S. (2022). Unified Modelling Language (UML) Tool for Software Architecture. *[Journal name not provided]*, *28*(11), 1618–1627.

[98] Dias, G. (2017). Evolvement of computer aided software engineering (CASE) tools: A user experience. *[Journal name not provided]*, *6*(3), 55.

[99] Dalli, D. (2010). *Automating the Design Process of A Secure Database*. University of Malta.

[100] Asghar, S., et al. (n.d.). Analytical framework for studying the effects of user participation in DSS development. In *4th International Conference on New Trends in Information Science and Service Science*.

[101] Meta Case. (2023). *MetaEdit+ Modeler – Supports your modeling language*. https://www.metacase.com/mep/ (Accessed August 1, 2023).

[102] Informer Technologies. (2023). *DB-Main 9.2*. https://software.informer.com/about.html (Accessed August 2, 2023).

[103] Mouratidis, H., & Giorgini, P. (2006). *Integrating Security and Software Engineering: Advances and Future Visions*. IGI Global.

[104] Singh, M., & Singh, C. (2017). Multi Tenancy Security in Cloud Computing. *[Journal name not provided]*, *4*(116), 1–7.

[105] Ziani, D., & Al-Muwayshir, R. (2017). Improving privacy and security in multitenant cloud ERP systems. *[Journal name not provided]*, *8*(5).

[106] Sun, P. (2020). Security and privacy protection in cloud computing: Discussions and challenges. *[Journal name not provided]*, *160*, 102642.

[107] Li, X., et al. (2010). Multi-tenancy based access control in cloud. In *2010 International Conference on Computational Intelligence and Software Engineering*.

[108] Anwar, M., & Imran, A. (2015). Access control for multi-tenancy in cloud-based health information systems. In *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*.

[109] Kumar, P. R., Raj, P. H., & Jelciana, P. (2018). Exploring data security issues and solutions in cloud computing. *[Journal name not provided]*, *125*, 691–697.

[110] Kumar, P., & Kumar Bhatt, A. (2020). Enhancing multi-tenancy security in the cloud computing using hybrid ECC-based data encryption approach. *[Journal name not provided]*, *14*(18), 3212–3222.

[111] Torkura, K. A., et al. (2021). Continuous auditing and threat detection in multi-cloud infrastructure. *Computers & Security*, *102*, 102124.

[112] Mishachandar, B., Vairamuthu, S., & Pavithra, M. (2021). A data security and integrity framework using third-party cloud auditing. *[Journal name not provided]*, *13*(5), 2081–2089.

[113] Giuri, L., & Iglio, P. (n.d.). A role-based secure database design tool. In *Proceedings of the 12th Annual Computer Security Applications Conference*.

[114] Abramov, J., Sturm, A., & Shoval, P. (2011). A pattern-based approach for secure database design. In *CAiSE 2011 International Workshops*, London, UK, June 20–24, 2011.

[115] Wazid, M., et al. (2019). Design of secure key management and user authentication scheme for fog computing services. *Future Generation Computer Systems*, *91*, 475–492.

[116] Zahra, R., Vella, J. G., & Cachia, E. (2019). Specifications of requirements to ensure proper implementation of security policies in cloud-based multi-tenant systems. *[Publication info not provided]*.

[117] AlJahdali, H., et al. (2014). Multi-tenancy in cloud computing. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*.

[118] Tiwari, P. K., & Mishra, B. (2012). Cloud computing security issues, challenges and solution. *[Journal name not provided]*, *2*(8), 306–310.

[119] Rao, R. V., & Selvamani, K. (2015). Data security challenges and its solutions in cloud computing. *[Journal name not provided]*, *48*, 204–209.

[120] Hoener, P. M. (2013). *Cloud computing security requirements and solutions: A systematic literature review*. [Thesis or publication info not provided].

[121] Bagate, R., & Chaugule, A. (n.d.). Cloud architecture and security measures. *[Journal name not provided]*, *1*(2), 24–27.

[122] Maddineni, V. S. K., & Ragi, S. (2012). *Security techniques for protecting data in cloud computing*. Blekinge Institute of Technology.

[123] Liu, Y., et al. (2015). A survey of security and privacy challenges in cloud computing: Solutions and future directions. *[Journal name not provided]*, *9*(3), 119–133.

[124] Abd Elmonem, M. A., Nasr, E. S., & Gheith, M. H. (2017). Automating requirements elicitation of cloud-based ERPs. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*.

[125] Tabrizchi, H., & Kuchaki Rafsanjani, M. (2020). A survey on security challenges in cloud computing: Issues, threats, and solutions. *[Journal name not provided]*, *76*(12), 9493–9532.

[126] PostgreSQL Global Development Group. (2023). *PostgreSQL 15 Documentation*. https://www.postgresql.org/docs/15/index.html (Accessed August 1, 2023).

[127] Amazon. (2023). *AWS Cloud Security*. https://aws.amazon.com/security/ (Accessed August 1, 2023).

[128] Oracle. (2023). *Database Security Guide*. https://docs.oracle.com/cd/B28359_01/network.111/b28531/index.html (Accessed August 1, 2023).

[129] Sterman, J. (2002). *System dynamics: Systems thinking and modeling for a complex world*. Massachusetts Institute of Technology, Engineering Systems Division.

[130] Haimes, Y. Y. (2018). *Modeling and managing interdependent complex systems of systems*.

[131] De Lucia, A., et al. (2010). An experimental comparison of ER and UML class diagrams for data modelling. *[Journal name not provided]*, *15*(5), 455–492.

[132] Aleryani, A. Y. (2016). Comparative study between data flow diagram and use case diagram. *[Journal name not provided]*, *6*(3), 124–126.

[133] CYBERTEC. (2023). *Secure PostgreSQL – A reminder on various attack surfaces*. https://www.cybertec-postgresql.com/en/secure-postgresql-a-reminder-on-various-attack-surfaces/ (Accessed August 1, 2023).

[134] PostgreSQL Global Development Group. (2023). *21.5. Password Authentication – Chapter 21. Client Authentication*. https://www.postgresql.org/docs/current/auth-password.html (Accessed August 2, 2023).

[135] PostgreSQL Global Development Group. (2023). *21.3. Authentication Methods – Chapter 21. Client Authentication*. https://www.postgresql.org/docs/15/auth-methods.html (Accessed August 2, 2023).

[136] Oracle. (2023). *Configuring privilege and role authorization*. https://docs.oracle.com/en/database/oracle/oracle-database/23/dbseg/configuring-privilegeand-role-authorization.html (Accessed September 16, 2023).

[137] PostgreSQL Global Development Group. (2023). *F.28. pgcrypto – Appendix F. Additional Supplied Modules*. https://www.postgresql.org/docs/current/pgcrypto.html (Accessed August 2, 2023).

[138] Oracle. (2023). *Manually encrypting data*. https://docs.oracle.com/en/database/oracle/oracle-database/23/dbseg/manually-encrypting-data.html (Accessed September 16, 2023).

[139] PostgreSQL Global Development Group. (2023). *Chapter 26. Backup and restore – Server administration*. https://www.postgresql.org/docs/15/backup.html (Accessed September 17, 2023).

[140] Amazon. (2023). *Supported AWS resources and third-party applications*. https://docs.aws.amazon.com/aws-backup/latest/devguide/whatisbackup.html (Accessed September 16, 2023).

[141] Oracle. (2023). *Oracle Recovery Manager (RMAN)*. https://www.oracle.com/ro/database/technologies/high-availability/rman.html (Accessed September 16, 2023).

[142] Amazon. (2023). *AWS Audit Manager*. https://aws.amazon.com/audit-manager/ (Accessed September 16, 2023).

[143] PostgreSQL Global Development Group. (2023). *pgbench – PostgreSQL client applications*. https://www.postgresql.org/docs/14/pgbench.html (Accessed August 3, 2023).

[144] Amazon. (2023). *Amazon CloudWatch concepts*. https://docs.aws.amazon.com/Amazon-CloudWatch/latest/monitoring/cloudwatch_concepts.html (Accessed September 16, 2023).

[145] Nanda, A. (2023). *Oracle Database 11g: The top features for DBAs and developers*. https://www.oracle.com/technical-resources/articles/database/sql-11g-spa.html (Accessed September 16, 2023).

[146] Zahra, R., & Vella, J. G. (2020). Incorporating security features in system design documents utilized for cloud-based databases. In *Conference Proceedings of 3rd International Conference on Information Systems and Management Science (ISMS)*.

[147] Oracle. (2023). *Historical test database provided by Oracle*. https://www.oracle.com (Accessed August 2, 2023).

[148] Camps, R. (2002). *Transforming N-ary relationships to database schemas: An old and forgotten problem*.

[149] Watt, A., & Eng, N. (2014). *Database design (2nd ed.)*. Victoria, Australia: BCcampus.

[150] Feynman, R., & Objectives, C. (2016). EBNF: A notation to describe syntax. *[Journal name not provided]*, *[Volume not specified]*, 10.

[151] Markoska, E., et al. (2015). Cloud portability standardization overview. In *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*.

[152] Chen, H., & Liao, H. (2010). A comparative study of view update problem. In *2010 International Conference on Data Storage and Data Engineering*.

[153] Teimoor, R. A. (2021). A review of database security concepts, risks, and problems. *[Journal name not provided]*, *5*(2), 38–46.

[154] PostgreSQL Global Development Group. (2023). *CREATE DATABASE SQL Commands*. https://www.postgresql.org/docs/current/sql-createdatabase.html (Accessed August 2, 2023).

[155] Yıldırım, M., & Mackie, I. (2019). Encouraging users to improve password security and memorability. *[Journal name not provided]*, *18*, 741–759.

[156] Kime, C. (2023). *How to prevent SQL injection: 5 key prevention methods*. https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/#prevention-methods (Accessed August 1, 2023).

[157] PostgreSQL Global Development Group. (2023). *25.1. SQL Dump – Chapter 25. Backup and Restore*. https://www.postgresql.org/docs/10/backup-dump.html (Accessed August 1, 2023).

[158] PostgreSQL Global Development Group. (2023). *25.3. Continuous archiving and point-in-time recovery (PITR) – Chapter 25. Backup and Restore*. https://www.postgresql.org/docs/10/continuous-archiving.html (Accessed August 1, 2023).

[159] Bermbach, D., Wittern, E., & Tai, S. (2017). *Cloud service benchmarking*.

[160] Vieira, M., & Madeira, H. (2005). Towards a security benchmark for database management systems. In *International Conference on Dependable Systems and Networks (DSN'05)*.

[161] Trustwave Holdings Inc. (2015). *AppDetectivePRO 8.6 user guide*. Trustwave Holdings, Inc.

[162] PostgreSQL Global Development Group. (2023). *19.8. Encryption options – Chapter 19. Server setup and operation*. https://www.postgresql.org/docs/current/encryption-options.html (Accessed August 2, 2023).

[163] E-SPIN. (2023). *The de-facto standard for corporate auditors and IT advisors*. https://www.e-spincorp.com/the-de-facto-standard-for-corporate-auditors-and-it-advisors/ (Accessed August 1, 2023).