BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **22**, No 4
Sofia • 2022 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2022-0036

A New Attribute-Based Access Control Model for RDBMS

Jaafer Al-Saraireh, Majid Hassan

Princess Sumaya University for Technology, Jordan
E-mails: J.saraireh@psut.edu.jo majidamto@gmail.com

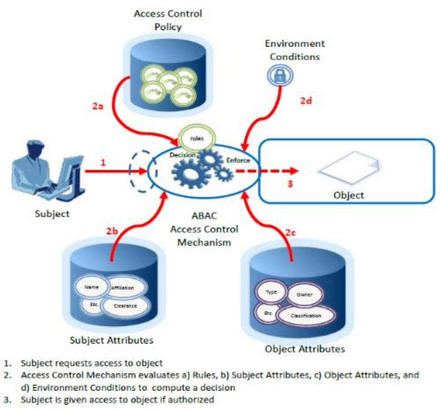
Abstract**:** *One of the challenges in Attribute-Based Access Control* (*ABAC*) *implementation is acquiring sufficient metadata against entities and attributes. Intelligent mining and extracting ABAC policies and attributes make ABAC implementation more feasible and cost-effective. This research paper focuses on attribute extraction from an existing enterprise relational database management system – RDBMS. The proposed approach tends to first classify entities according to some aspects of RDBMS systems. By reverse engineering, some metadata elements and ranking values are calculated for each part. Then entities and attributes are assigned a final rank that helps to decide what attribute subset is a candidate to be an optimal input for ABAC implementation. The proposed approach has been tested and implemented against an existing enterprise RDBMS, and the results are then evaluated. The approach enables the choice to trade-off between accuracy and overhead. The results score an accuracy of up to 80% with no overhead or 88% of accuracy with 65% overhead.*

Keywords**:** *ABAC, Access control, Entity cardinality rank, Dominance level rank.*

1. Introduction

Database management is considered an essential component of many information systems to store data [1, 2]. Access control is a facility that determines the legitimacy of every attempt by a user to access any resource of the system. There are, mainly, four Access Control models: (1) Mandatory Access Control model (MAC); (2) Discretionary Access Control model (DAC); (3) Role-Based Access Control model (RBAC); (4) Attribute-Based Access Control (ABAC).

In security systems, three concepts are recognized: access control policies, access control models, and access control mechanisms [3]. In ABAC, authorization to access the objects is not directly granted to the users. It uses attributes of the users, objects, requested operations, and environment to define access authorization on objects. Users must prove that they possess the specified attributes they claim to own. For this reason, ABAC access control relies on user authentication at the time of a request. Fig. 1 illustrates the ABAC model.



1. Subject requests access to object
2. Access Control Mechanism evaluates a) Rules, b) Subject Attributes, c) Object Attributes, and d) Environment Conditions to compute a decision
3. Subject is given access to object if authorized

Fig. 1. ABAC model [4]

ABAC permits or denies actors' requests based on some attributes of the participant entities and environmental conditions. ABAC controls access to resources by evaluating rules against the attributes of entities relevant to a request [5]. This flexibility enables addressing security policy rules without mapping explicit relationships between subjects and objects. In the ABAC model, access requests are evaluated by changing attribute values per request. Access decisions are always evaluated the same way without looking for a specific subject/object relationship each time requests are made. The typical ABAC model has different attribute categories: User Attributes, Object Attributes, and Environmental Attributes.

The rest of the paper is organized as follows. We discuss related work in Section 2, describe the methodology in Section 3, Section 4 presents the proposed approach, Section 5 experiments, evaluation and shows the results of the experiments. We present our conclusions and future work in Section 6.

2. Related work

One of the challenges in ABAC implementation is acquiring sufficient metadata on entities and attributes. Some researchers, such as [6], have found that the absence of a sufficient set of entities and attributes causes the ABAC system not to work properly. They state that entities and their attributes must be established, maintained, and made available before ABAC implementation. However, they do not point to the issue of gathering and addressing the related system entities and their attributes that will be involved in the ABAC system.

The combining MAC and ABAC have been presented by [7]. A model that preserves the MAC Access control approach while enhancing access control decisions with security attributes so that security clearance of a subject or object is an attribute. Subjects, objects, and the environment are considered as entities with finite attributes. Mandatory attributes are established and attached to each entity, and any permission request mandatorily requires these attributes. The main focus of the research is on combining the two models of MAC and ABAC, but the study's attributed extracting and selection problems are still undiscussed. A new access control model called Attribute-Rule ABAC (AR-ABAC was proposed by [8]; the suggested model recognizes the attribute rules concepts that address the correlation between users and objects and the opportunity of accessing object entities based on their security levels. The presented model specifies assumptions that specify the appropriate attributes that should be used and the number of attributes token for making access decisions. While the proposed model sets that policies are responsible for attribute selection, the model does not answer how this could be achieved.

In [9], the focus is on how to extract and define a set of authorization attributes for the ABAC system. Using natural language processing, they triy providing algorithms to extract attributes from ABAC policy documents automatically. Their research differs from this research by the extraction method and source of extraction. Also, another research [10] has tried to utilize natural language processing to extract and define the constraints of ABAC security policies. Both ABAC policies and constraints imply the extraction of ABAC attributes too. The research focuses on ABAC constraints rather than the attributes with which the rules are formulated.

The work of [11] offers ABAC policy mining from access request logs. Their research reflects the importance of automated mining ABAC attributes and policies; their algorithms rely on an existing working ABAC system. They do not consider cases when no ABAC system exists. An efficient ABAC policy retrieval method based on attribute and value levels in multimedia networks is presented by [12]. The approach is based on retrieving policies that satisfy the request at the attribute level by computing based on the binary identifier. The depth index is implemented in this research to reconstruct the policy decision tree at the attribute value level. A new access control model was presented by [13]. It is a model based on combined RBAC and ABAC. ABAC with anonymous access has been introduced by [14]; the proposed model inherits the features of the ABAC model, such as fine-grained authorization, policy flexibility, and unlimited object types.

Reachability analysis for effective user attributes based on the direct attributes assigned to the user or its member user groups is presented by [15]. This approach uses the polynomial-time algorithm to solve special schemes' instances under restricted conditions. A multi-level security attribute-based access control scheme is presented by [16]. The approach is based on static and dynamic user attributes combined with the assigned security level that satisfies the requirements of NIST's ABAC model. A new access control model for smart homes is presented by [17]. The model captures different user, environment, operation, and device characteristics based on a dynamic fine-grained ABAC approach.

The literature reviews explain that a poor selection of involved entities and attributes could dramatically increase the complexity of the ABAC system, making managing such a system very tough. Many kinds of literature suggested models and approaches to either combine the features of two models or how to re-configure some ABAC features to enhance ABAC functionality. Still, none have discussed how to address a good attribute selection method to formulate ABAC permissions. This proposed research introduces a mechanism that aims to choose the optimal subset of attributes to be involved in ABAC permissions addressing keeping the ABAC model at its least complexity with the minimum overhead and cost. This proposed research contributes to existing knowledge by: (1) offering automated extraction of entities and attribute metadata from an existing RDBMS for ABAC implementation; (2) enhance the ability to decide what attributes are likely candidates to play the input of ABAC policy rules.

3. Methodology

Implementing ABAC security in existing enterprise RDBMS has a subset of challenges. One challenge is collecting and extracting sufficient entities and attributes metadata for use in the ABAC security policies rules design domain. It is hard to establish entities and attributes metadata manually for an enterprise RDBMS due to the nature of such large systems and environments. There is a strong need for automated extraction of ABAC entities and attributes metadata before the ABAC implementation.

3.1. Proposed approach model

We must address the system's security policies to implement an ABAC model in an enterprise system. Security policies of the typical ABAC model consist of conditional expressions formulating the access permission or denial rules. Such expressions are constructed using attributes of entities such as subjects, objects, and environment.

The first step of starting ABAC implementation is gaining the attributes repository of entities. It requires gathering all system entities' information, attributes, attributes values, relationships, etc., and putting it together in a well-organized repository to access later during implementation. This proposed research offers a semi-fully automated approach to extracting entities and attributes for ABAC implementation from an existing enterprise RDBMS. It makes it easier to implement ABAC by automatic attribute extraction and selection. Offered entities and attributes extraction approach relies on reverse-engineering scripts against RDBMS and some ranking formulas for weighting the selectiveness of entities and attributes.

The second step categorizes entities into three major categories, subject, object, and environmental entities. Also, when considering security policies as requested processes and obligators for operations, it is trivial to relate the security policies to the workflow processes of the enterprise. The third step separates classification dimensions for the scope of entities. The scope of entities is defined as the scope of visibility of an entity within a system where it is effective and applicable. Entities within the widest scope are important in security policy design because they apply to the whole system. We might use entities limited to the undergone scope for sub-system security policies. The scope we work within determines what entities we should involve in addressing security policies. Entities with larger scopes are the most candidates in security policy design.

Fourth step, security policies and rules expressions should be designed to serve as long as possible. Security policy rules should be valid and applicable all the time. Expressions in access permissions should involve valid entities that represent persistent and valid parts of the system. It is important to ensure that there is no chance of causing the access model engine to interpret invalid access rule expressions because this will lead to the breakdown of the security system. The fifth step is ranking each entity and its attributes; a rank is given to the entity and the attribute. Entity type and scope are assigned manually from pre-specified values. Entity state stability and lifetime ranks are given to each entity from 0 to 1. As the cardinality of both entity and attributes and entity references count are reverse-engineered from RDBMS, the cardinality rank and reference power are then calculated according to specific formulas. The following are the ranking formulas used in our proposed approach.

1. Reference Power Rank of an entity:

$$\text{Reference Power Rank} = \left(\frac{2r}{N} \times 100\right) + 1, \tag{1}$$

where *N* is the total count of entities, and *r* represents the number of references to the entity.

2. Cardinality Rank of an entity:

$$\text{Cardinality Rank} = \frac{100}{\text{Ciel}\left(\frac{k}{50}\right)}. \tag{2}$$

The entity type rank of an entity is assigned a value as follows: The sub-module level has the lowest value of 50, which is approximately a quarter of the highest power reference rank. Global-level has the value of 201 as the highest power reference rank. AC level has the value of 603 as 3 times the highest power reference rank. Unique level entities are given a value of 10050, resulting by multiplying 50 (the lowest rank of entity types) by 201 (the highest power reference rank).

3. Entity State & Lifetime Ranks of an entity are manually assigned values between 0 and 1 multiplied by 10.

4. Entity Dominance Level Rank: It is the result of multiplying all entity's ranks by each other as follows:

$$\begin{aligned}\text{Entity Dominance Level Rank} &= \text{Scope Rank} \times \text{State Stability Rank} \times \\ &\times \text{ASL Rank} \times \text{Cardinality Rank} \times \text{Reference Power Rank}.\end{aligned} \tag{3}$$

5. Attribute Dominance Level Rank: it is the result of the following formula:

$$\text{Attribute Doinance Level Rank} = \text{Log}\left(\frac{\text{Max}(E,1) \times K}{d+n}\right), \tag{4}$$

where *E* represents the Entity Dominance Level, *K* represents entity cardinality rank, *d* represents the number of distinct attributes among entity instances, and *n* represents the number of null values of attributes among entity instances.

3.2. Evaluation metrics

In this work, the rules and classification approach are defined to extract and define the attributes related to entities and the nature of the relationship between an entity and its extracted attributes. To measure the efficiency of the selected subset of attributes, we may rely on the following measures:

1. The percentage of selected attributes to the total number of attributes available in the system. This percentage should be at its least value. The minimum number of attributes in the attribute's subset, the less complexity of the ABAC model, the simpler management of the access control model, and the easier implementation of ABAC.

2. The selected attributes cover the percentage of the organization's security policies. When choosing the efficient and optimal subset of attributes, all the access

control policies will be implemented efficiently and adequately without needing to add more attributes to the subset of the selected attributes. These attributes are sufficient to implement all access roles when specifying the conditions and rules of access security policies. It is the optimal case, but when this percentage increases, we have covered more security policy scenarios and become closer to the optimal point. A greater number of attributes leads to covering more security policy scenarios. However, we do not want this to increase the complexity of our ABAC model; thus, there is some point where we achieve better ABAC complexity with the least possible number of attributes. ABAC performance and efficiency still increase with the increase of the number of attributes until we add more attributes; this will decrease the ABAC performance.

The suggested model results in an adequate ability to choose attributes and entity candidates to be incorporated in access policy specifications. Some metrics are followed in deciding what attributes are a candidate to be selected; some of these metrics are summarized entity type, the scope of the entity, entity lifetime, attribute relationships, and attribute acquisition.

Also, this research started by analyzing the characteristics of a typical enterprise information system RDBMS. It addressed some concepts that may help more understand entities' relationships, roles, and influences among each other. Fig. 2 shows the scope of this research and its relationship with the ABAC model components; the scope is shown in the blue area.

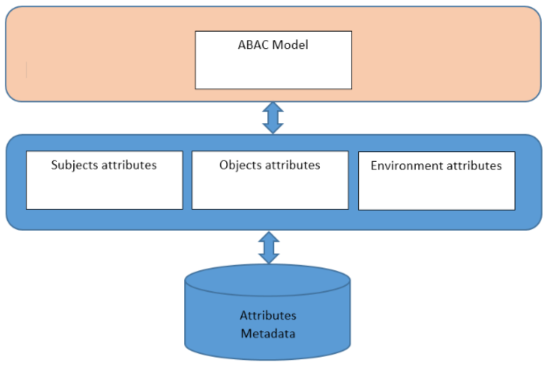


Fig. 2. General model for the proposed approach

The model will then be tested against some existing RDBMS enterprise applications to prove the results achieved. The metrics will be applied and measured to determine the presented model's advantages in commercial and enterprise applications.

4. Proposed Approach of attribute-Based Access Control (ABAC)

The proposed research first tries to provide some concepts that will lead us to a specific categorization and classification of the entities and the attribute's natures and characteristics. These classifications are supposed to help in extracting and collecting entities and attribute metadata that will be a guide to designing the AC policies for the ABAC system effectively.

4.1. Evaluation metrics

Entities in typical ABAC models are classified into main types object, actor, operation, and environmental entities. We extend this classification into more types for much better recognition of the nature of enterprise entities. The classification of entity types in our proposed ABAC model is shown in Fig. 3. We classify the entities according to their nature as follows:

1. Actors entities: define the actors of the enterprise application.
2. Session entities: distinguish every made session in the enterprise application.
3. Objects Entities: the object here refers to every subject that undergoes an operation by an actor within the enterprise application.
4. Enterprise structure entities define an enterprise organizational unit's hierarchy. The enterprise structure entity is an extension of the object entity. They are vital in determining other entities' attributes during the enterprise's workflow processes. An actor must belong to a certain organizational unit, and most objects must belong to a certain enterprise entity. Enterprise structure entities are sub-classified into two major types: (a) *Enterprise Organizational Structure* determines the enterprise organizational units and how they are related; (b) *Enterprise Job roles entities* define job roles within the enterprise, such as CEO, general managers, department heads, etc.
5. Workflow entities, such as activities, operations, and transactions, represent the enterprise's workflow. We address two types of workflow entities in terms of complexity: (a) *Operation entities* define the basic activities against object entities; (b) *Transactions entities* represent the complicated activities processing that consists of operation sequences.

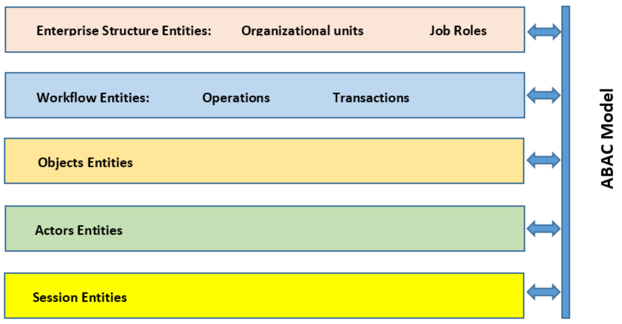


Fig. 3. Entity types classification in ABAC model

4.2. Entities scope

This research introduces entity scope, an important characteristic of candidate entity selection for ABAC policies. Entity scope is where the entity is seen or effective among the enterprise application modules. For example, the organizational units are global among all the enterprise modules in any well-integrated RDBMS application for an enterprise. Other entities may be related only to some functionalities within one module or more. Entities with wide scopes are the most candidates in selecting entities for addressing ABAC role policies. Opposite entities with a narrower range are not used widely in addressing ABAC role policies. The proposed approach classifies the scopes of the entities into three types.

1. Global entities' scope consists of all entities that are effective among all the enterprise RDBMS applications and visible and effective within them. Global entities are very useful when addressing any ABAC policies for the enterprise; most effective ABAC model policies depend on them. Some entities always return one unique entity instance wherever the entity is called to read its instances within any session in the context of the enterprise application. We call any entity that satisfies this condition Unique Instance Entity (UIE). If a UIE of E exists in a session, then the entity instance E directly inherits all its attributes to every entity within the same session. UIE could be classified as follows: (a) *Absolute Unique Instance Entities* – these entities have one single instance (i.e., record), such as the working enterprise definition entity and the global master application policy entity; (b) *Session-level Unique Instance Entities* – these are unique instances relating to the current session. These instances are specified by the environment resulting from the session initiated by an actor.

2. Access Control Module Scope consists of entities related to AC-specific purposes, such as policy roles and permissions. Entities of this scope are very special because ABAC roles often depend on attributes of actors having access to other permission entities or role entities. It is very important for the separation of duty feature enabling and unification of duties feature.

3. Sub-Modules Specific scope addresses the entities attached only to a specific sub-module or sub-system. The benefits of entity scope classification help us determine the best candidate entities to manage ABAC policy roles at a given scope. Moreover, each scope inherits the entities from scopes that fall above, as represented in Fig. 4.

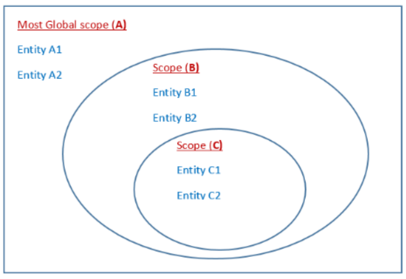


Fig. 4. Entities scopes

4.3. Entities state stability

Entity instances in RDBMS enterprise applications undergo state transition during processing. If we assumed that each entity instance initiation process transits it from the start (initiating or starting) to state E_0 "created" or "saved", as shown in Fig. 5, then every entity instance in the system has at least one state E_0.

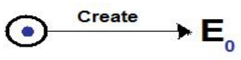


Fig. 5. Entity initiation operation

Some entities take many states during workflow processing. For example, an employment application may transit from the “initiated” state to the “submitted” state and then to the “reviewed” state and then to the “short-listed” state, etc., till it achieves the final state “admitted” or “rejected”. Because entity state transition affects the attributes of entity E_1 itself by changing the value of one or more of its attributes, the entity will be a different entity E_2. So, referring to entity E_1 in ABAC policies roles will no longer be valid when the entity converts to the state Linux Libertine E_2, as shown in Fig. 6. Entities vulnerable to changing their current state to another state have less state stability. On the other side, entities that are not vulnerable to change have high state stability. This distinction is very helpful in detecting more stable entities because they are more likely to be involved in our proposed ABAC model to address policy roles. Because entities that change their state are unstable, they would be a poor choice in addressing ABAC policies.

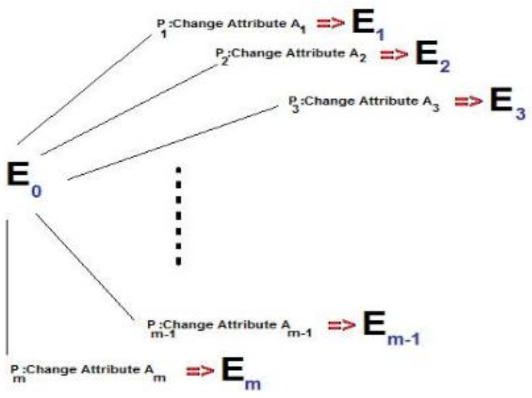

P_1:Change Attribute A_1 => E_1
P_2:Change Attribute A_2 => E_2
P_3:Change Attribute A_3 => E_3
E_0
P_{m-1}:Change Attribute A_{m-1} => E_{m-1}
P_m:Change Attribute A_m => E_m

Fig. 6. Entity transition due to changing its attributes by operations

4.4. Entity instances significance lifetime

It is the period the entity is still usable or referred to when processing its instances. In some RDBMS enterprise applications, schema entities are used for some time only for data partitioning purposes. Some enterprise RDBMS applications have a separate database schema for the same module, often on time, for several purposes, such as separating every working year in a separate database schema or separating each working project in a separate database schema. In these cases, the Entity Significance Lifetime lasts one working year or until an active project is closed. ABAC roles that depend on entities with temporary or limited significance lifetime are still only effective for that lifetime and are not usable anymore. Although we need to make ABAC policies for them, in general, ABAC policy roles are not feasible; we should redefine the entity by itself to include all its representing occurrences or duplicates.

Entity Instances Significance lifetime is the period from when the entity instance has been created until it reaches the last final state of its business cycle. Each entity instance undergoes a transition during the workflow processing, which transits it from one state to another until it reaches its final state. For example: in the RDBMS HR system, an employment application instance has reached its last final state when the employment application is either approved or rejected. The lifetime of any entity or its instance has two stages following each other:

(a) **Active Significance Lifetime (ASL).** The period when the entity instance *i* accept at least one operation "op" that transfers the entity instance *i* from its current state to another.

(b) **Not-Active Significance Lifetime (NASL).** It is the period whenever the entity instance *i* do not accept any attempt to transfer it from its current state to any other state. But is still referred to by the current workflow or accessed by the working flow processes, reports, or inquiries.

The above classification is important because it enables us to distinguish entities with a permanent lifetime from entities with a limited lifetime. We suggest relying on entity instances with a permanent significance lifetime in our proposed ABAC model. Involving entities with a short significance lifetime for their instances makes ABAC roles meaningless after losing their significance.

4.5. Instance cardinality

This refers to the number of instances the entity could have. The degree of the cardinality of an entity instance is an important feature in our model, as entities with a huge number of instances are less likely to be involved in our ABAC model. Entities with small cardinality would be a smart choice for addressing ABAC policies. Attributes values cardinality refers to the number of distinct values an attribute has among entity instances. Entities with a large volume of instances (i.e., high cardinality) might also have attributes with few distinct values. Such attributes can be selected in addressing permission roles against their entities.

4.6. Referential power

It refers to how many references are made from other entities to that entity in the enterprise RDBMS application. Some entities have a powerful referential level as they have participated in defining many entities and being a fundamental attribute of them. Entities with powerful referential levels play a vital role in defining many other entities and accordingly pass their attributes to those entities, so they are most likely to be involved in addressing ABAC roles and policies. On the other hand, entities that are not referred to or have weak referential power are seldom engaged in addressing ABAC's general policies.

4.7. Putting all together

In our proposed model, we introduce important rules to decide which entities are the candidates most to be involved in the ABAC model for an enterprise RDBMS application. In terms of the above classifications, we introduce EDL, which defines the importance degree of an entity in participating in addressing ABAC model policies. The EDL is determined according to its scope, state stability, significance lifetime, cardinality, and referential power. Entities could be categorized according to their dominance level. They could be ranked from entities with strong dominance levels to entities with weak dominance levels.

In this proposed research, a measurement has been developed to measure the dominance level of each entity. For each criteria of scope, state stability, active significance lifetime, cardinality, and referential power, a value, or a formula, is given to each criterion to measure each criterion’s degree. Below in Table 1, different measures of entity ranks are followed in the test to calculate the EDL degree.

Table 1. Measuring entity ranks

Property	The rank calculation for property		Notes
Entity Scope rank	Session Unique Level	10050	(50) lowest scope rank multiplied by (201) maximum reference power
	Absolute Unique Level	10050	(50) lowest scope rank multiplied by (201) highest reference power
	Access Control Level	603	Three times of highest reference power
	Global Level	201	As the highest reference power
	Sub-Module Level	50	Approximately a quarter of the highest reference power
Entity State Stability rank	State Stability degree×10		
Active Significant Lifetime (ASL) rank	ASL degree×10		
Entity Cardinality rank	Let k = cardinality, IF k= 0 then rank =0 Else $\text{Cardinality Rank} = \frac{100}{\text{Ciel}\left(\frac{k}{50}\right)}$		Assuming cardinalities ≤50 has the most rank value. Ciel(n) function returns the smallest integer, which is greater than or equal to the number n
Reference Power rank	Let R = the number of references for the entity. Let N = Total count of entities. IF entity scope in (session unique or absolute unique) then rank= 1 Else $\text{Reference Power Rank} = \left(\frac{2r}{N} \times 100\right) + 1$		Considering the ratio of the number of references to the half-count of entities Unique level scopes have a reference value of 1
Attribute Dominance Level Rank	Let E = Entity Dominance Level rank, k= Entity Cardinality, d= number of distinct attribute values within entity instances, n= Number of null values occurrences of attribute within entity instances. Attribute Dominance Level Rank = $= \text{Log}\left(\frac{\text{Max}(E,1) \times K}{d+n}\right)$		

5. Experiments, results, and evaluation

The proposed concepts and measures presented have been tested against the RDBMS enterprise application. The RDBMS type was Oracle 12c operating on the windows 16 server operating system. The system was hosted in an HP Blade server with

8 Quad Cores, a 2.7 GHz CPU, and 128 GB RAM. The number of tested database schemas was 6, consisting of over 2320 entity tables with over 69,150 attributes. The tested RDBMS contains a total of 158 million records.

5.1. Experiment

Initially, target database entities were refined by running scripts to exclude non-used entities. The final subset contained 877 effective entities and about 15,800 attributes containing 128 million records. Scripts were written against a database to be automatically reverse-engineered and analyzed to extract the properties and classifications addressed above. The database has been studied for tables instances cardinality, attributes cardinality, and references among database objects. Each RDBMS table has an entity type, scope, and state stability value. Also, entity instance active significance lifetime has been estimated for each database entity. For both cardinality and references of each database entity, they were automatically reverse-engineered and extracted from the database. For attributes cardinality, the reverse engineering process also helped get the number of distinct values and no null values for each attribute per entity. Below are sample scripts used to reverse-engineer the database. View for reverse-engineering the referential cross entities is presented as Algorithm 1.

Algorithm 1. Reverse Engineering the Referential Cross Entities

```
CREATE OR REPLACE FORCE VIEW V_FKS AS
  select  c.OWNER, c.TABLE_NAME, f.COLUMN_NAME, c.R_OWNER,
f2.TABLE_NAME as R_TABLE_NAME,
            f2.COLUMN_NAME as R_COLUMN_NAME,
f.CONSTRAINT_NAME, c.CONSTRAINT_TYPE, c.DELETE_RULE,
           f.POSITION, c.INDEX_NAME
  from  dba_constraints  c, dba_cons_columns f, dba_cons_columns    f2
  where c.OWNER = f.OWNER and c.CONSTRAINT_NAME =
f.CONSTRAINT_NAME and
             c.TABLE_NAME = f.TABLE_NAME and c.R_OWNER =
f2.OWNER and
             c.R_CONSTRAINT_NAME = f2.CONSTRAINT_NAME and
f.POSITION= f2.POSITION and
             c.CONSTRAINT_TYPE in ('R');
```

The script to reverse engineering the cardinality and referential cross entities are presented as Algorithm 2.

Algorithm 2. Reverse Engineering Cardinality and Referential Cross Entities

```
Declare
  v_cardinality  number;
  v_FK_count  number;
begin
```

```
For rec in ( select * from dba_tables where owner in ('schema1',' schema2',…,'
schema6'))
Loop
     Execute immediate
         ‘Select count(0) from :owner.:tbl_nm ‘ into v_cardinality using
rec.owner, rec. table_name;
     Execute immediate
         ‘Select count(0) from v_fks where R_owner=:tbl_ownr and
R_table_name = :tbl_nm ‘
             into v_FK_count   using rec.owner, rec. table_name;
     Update my_entities
     set cardinality = v_cardinality,    FK_count = v_fk_count
     where owner = rec.owner and entity_name = rec.table_name;
End loop;
```

Script to reverse-engineering attributes distinct values and null values within entities instances:

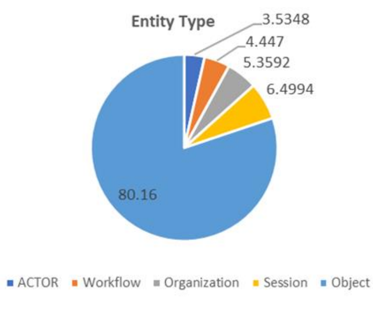
Algorithm 3. Reverse-engineering attributes distinct values and null values within entities instances

```
Declare
   v_distinct_vals  number;
   v_null_vals       number;
begin
For rec in ( select * from dba_tab_columns where the owner in
(‘schema1’,’schema2’,…,’schema6’) )
Loop
     Execute immediate
           ‘select count(0) from (Select :column_name  from :owner.:tbl_nm
where :column_name is not null group by
             :column_name) ‘ into v_distinct_vals   using rec.column_name,
rec.owner, rec. table_name, rec._column_name;
     Execute immediate
         ‘Select count(0)  from :owner.:tbl_nm  where :column_name is null) ‘
into v_nulls_vals   using rec.owner, rec.
           table_name, rec._column_name;
    Update my_attributes
    set distinct_vals = v_distinct_vals,     nulls_count = v_null_vals
    where owner = rec.owner and entity_name = rec.table_name and
attribute_name = rec.column_name;
End loop;
```

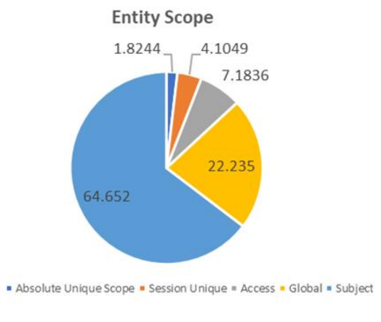
After reverse-engineering and data classification, the equations in Section 3.5 were applied against entities and attributed to getting the different ranks of their classifications. The test experiment was repeated 10 times using reverse engineering, reviewing classification, and applying the equations to calculate the ranks. The results were taken as the average of the 10 results.

5.2. Results and discussion

Below in Figs 7 (a) and (b) are the charts of entity types and scopes classification, respectively. 80% of entities have been classified as object entities, and 6.4%, 5.3%, 4.4%, and 3.5% were classified as session entities, organizational-unit entities, workflow entities, and actor entities, respectively. Whereas 64.7% of entities were categorized in sub-module scope, 22.2%, 7.2%, 4.1%, and 1.8% were classified into the global scope, access control scope, unique session scope, and absolute unique scope, respectively. Ranks of Entity State Stability are shown in Fig. 8, and Active Significant Lifetime (ASL) ranks are shown in Fig. 9.



(a) Entities types classification



(b) Entities scopes classification

Fig. 7. Entities types and Scope classification

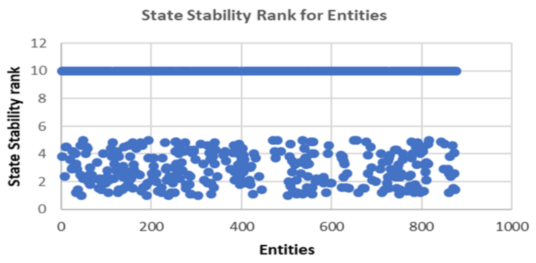


Fig. 8. Entity State Stability

63% of entities have an almost stable entity state of 10 degrees, whereas the 37% have a less stable entity state stability. 10% of entities have state stability less than or equal to 2 degrees, indicating that they show very low state stability.

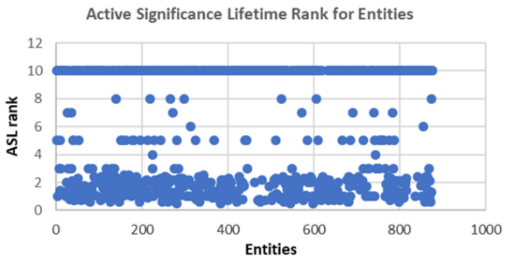
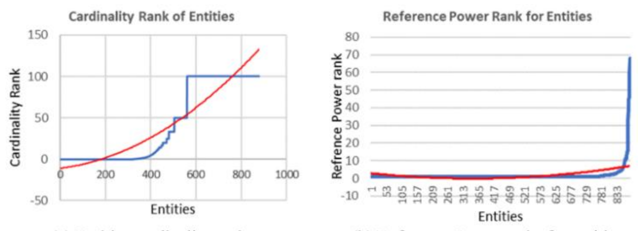


Fig. 9. Entity Active Significance Lifetime (ASL)

47% of entities have the longest ASL of 10 degrees, whereas 53% have shorter ASL degrees, varying from 1 to 8 degrees. 33% of entities have ASL degrees less than or equal to 2 degrees, indicating that they have a very short ASL. A logarithmic chart of entities' cardinality ranks is shown in Fig. 10 (a), where 35% of entities have a cardinality rank equal to 100. This means they contain very few instances, less than or equal to 50. Whereas 37% of entities have a cardinality degree equal to 0, indicating that they contain many instances, some of which include tens of millions of instances. The reference Power rank of entities was reverse-engineered and set as shown in Fig. 10 (b). 2.1% of entities have a Reference Power rank greater than or equal to 10 degrees. It means that these entities have strong referential power among other entities. 87% of entities have a very low Reference Power rank of 1 degree, indicating that they are either not being referenced or referenced only once.



(a) Entities cardinality ranks (b) Reference Power ranks for entities

Fig. 10. Entities cardinality ranks and Reference Power ranks for entities

EDL ranks have been calculated, and the linear and logarithmic values charts are shown in Figs 11 (a) and (b), respectively. Table 2 shows the percentages of entities per different partitions of EDL ranks. The Logarithmic scale shows that 18.57% of entities have a Dominance Level rank greater than or equal to 2 degrees, 6.08% of entities have a Dominance Level rank greater than or equal to 4 degrees, and 3.3% of entities have a Dominance Level rank greater than or equal to 6 degrees which means that these entities are the most candidate to be effective in ABAC model permission roles addressing. On the other hand, 74.34% of entities gained ≤1 degree of Dominance Level rank, meaning that these entities have very little chance to play roles as actor objects or session objects and seem to play roles as "objects" where permissions might be addressed against them. Figs 12-14 show logarithmic charts of Attributes Dominance Level (ADL) ranks for different entity types. Table 3 shows the percentage of entities falling in different partitions of ADL ranks for each entity type.

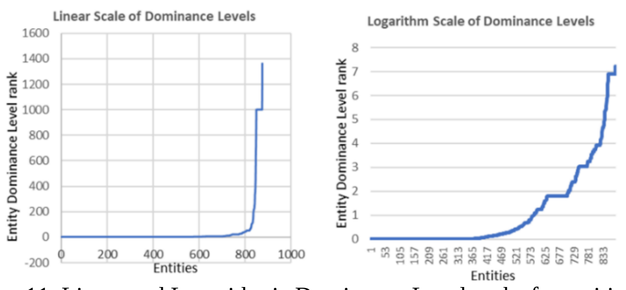


Fig. 11. Linear and Logarithmic Dominance Level ranks for entities

Table 2. Entity Dominance Level ranks and partitions (%)

Rank				
≤1	≥2	≥4	≥6	≥7
74.34%	18.57%	6.08%	3.3%	0.11%

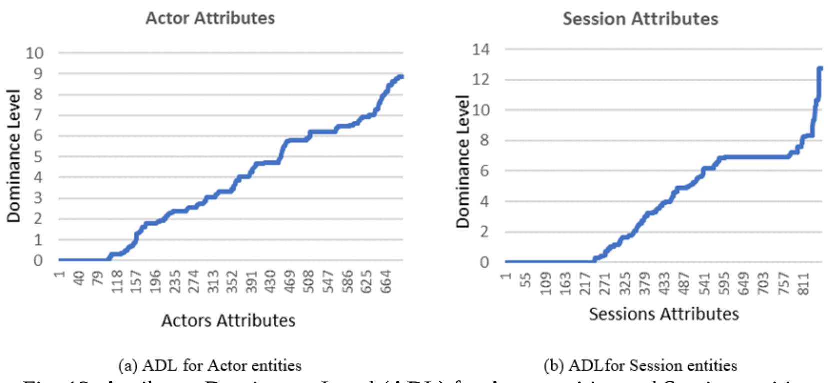


Fig. 12. Attributes Dominance Level (ADL) for Actor entities and Session entities

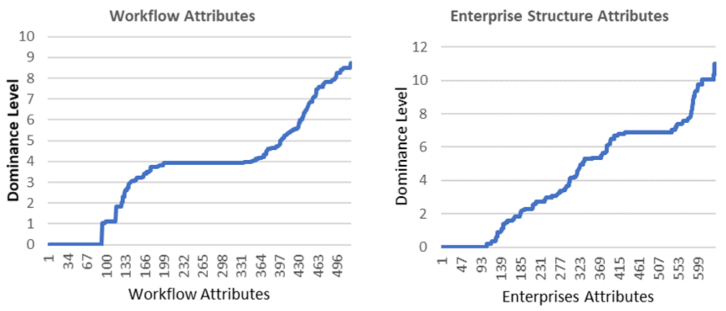


Fig. 13. Attributes Dominance Level (ADL) for Workflow entities and Enterprise Structure entities

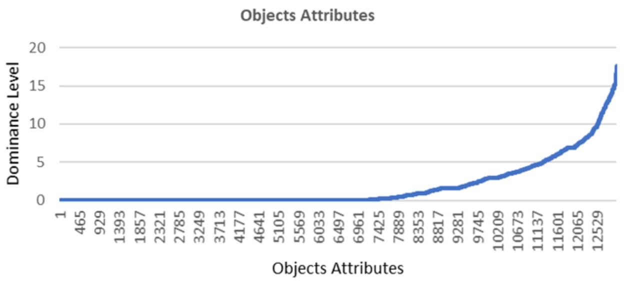


Fig. 14. Attributes Dominance Level (ADL) for Objects entities

Table 3. Different partitions (%) of Attributes Dominance Level ranks by entity type

Entity type	Rank						
	≤1	≥2	≥5	≥7	≥10	≥12	≥15
Actor Entities	17.85%	74.78%	42.8%	19.84%	5.33%	1.39%	0%
Session Entities	31.53%	60.28%	43.25%	10.58%	2.33%	1.05%	0%
Enterprise-Structure Entities	19.73%	71.68%	48.41%	16.01%	4.85%	0%	0%
Workflow Entities	19.25%	75.55%	23.05%	12.53%	0%	0%	0%
Object Entities	63.42%	30.56%	15.01%	8.58%	3.84%	2.46%	0.38%

Fig. 15 shows the logarithmic scale of ADL ranks for all attributes of entities. The tested RDBMS database consists of more than 15.8K attributes. Table 4 shows the percentage of entities per different partitions of ADL ranks.

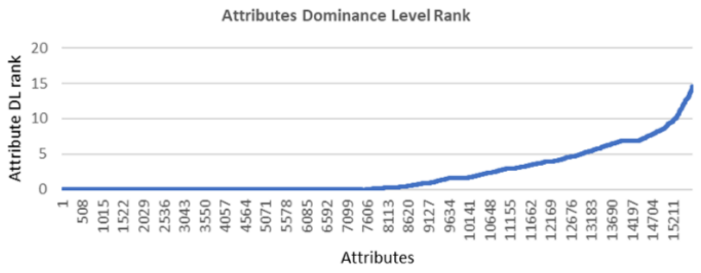


Fig. 15. Logarithmic scale of Attributes Dominance Level ranks

Table 4. Different partitions of Attributes Dominance Level ranks and partition (%)

Rank						
≤ 1	≥2	≥5	≥7	≥10	≥12	≥15
56.96%	34.82%	18.14%	8.8%	3.37%	1.96%	0.29%

The table above shows that 8.8% of attributes have a Dominance Level degree greater than or equal to 7 degrees. Only 0.32% of entities obtain a Dominance Level rank of 15 degrees or above. 56.96% of attributes have a very low Dominance Level rank of 1 degree or below. The proposed research suggests considering an EDL rank of 2 degrees from the test results, covering 18.57% of entities. It suggests considering the Attribute Dominance Level rank of 7 degrees which covers 8.8% of attributes belonging to 17.67% of entities.

5.3. Evaluation

Considering ADL rank equal to 7 degrees will populate 8.8% of all available attributes. These attributes belong to only 17.67% of system entities. This small number of attributes (which is, in our case, equal to 1389 attributes out of 15,800 attributes) makes the ABAC model much less complex and easier to maintain. Reducing the selected ADL can lead to a much more complex ABAC model. ADL rank of 5 populates 18.14% of available attributes, which belong to 32.5% of system entities. The accuracy of ADL ranks is presented in Table 5.

Table 5. Accuracy table of Attributes Dominance Level ranks

Dominance level	Accuracy, %	Shortage, %	Attributes used	Attributes populated	Complexity, %	Overhead, %
1	98.62	1.38	1,713	7,120	10.85	**309.90**
2	97.12	2.88	1,787	5,504	10.69	**216.87**
3	95.62	4.38	1,661	4,707	10.52	**170.98**
4	91.71	8.29	1,593	3,529	10.09	**103.17**
5	88.49	11.51	1,537	2,873	9.74	**65.40**
6	85.72	14.28	1,489	2,214	9.43	**27.46**
7	79.97	20.03	1,389	1,389	8.80	–20.03
10	34.04	65.96	532	532	3.37	–69.37
12	19.83	80.17	310	310	1.96	–82.15
15	2.94	97.06	46	46	0.29	–97.35

By reviewing actual security policies in the tested enterprise RDBMS application, it was found that by choosing 5 degrees of ADL rank, about 11.51% of security policies have not been covered by the subset of attributes selected above. So, it can be said that around 88.49% of security policies are covered by the subset of the attributes chosen by the proposed approach. There is an overhead of 65%, meaning attributes populated with dominance level rank 5 are 65% more than actually needed attributes. Table 5 shows the accuracy level, ABAC complexity impact, and the overhead for different values of ADL ranks.

From Table 5, we find that decreasing the Dominance Level improves the accuracy, but on the other side, the overhead increases. At a Dominance Level equal to 7, the accuracy is about 80% with no positive overhead. However, with Dominance Level 5, the accuracy is about 88.5%, with 65.4% overhead. If we can seek the best accuracy percentage with the most acceptable overhead, a dominance level between 5 and 7 will be quite satisfactory. The results of the comparison between ABAC and the proposed model are shown in Table 6.

Table 6. Comparison between ABAC and the Proposed approach

Characteristics	ABAC	Proposed model
Easy configurability	No	Yes
Fine-grained policies	Yes	Yes
Workflow control	No	Yes
Performance	Medium	High
Granularity	Medium	High
Simplicity	Hard to establish all the policies	Easy

5. Conclusion

Attribute governance in the ABAC system requires more analysis of attributes, entities, and values domains. Classifying attributes and becoming aware of the relationships among entities and attributes increases the ABAC system's efficiency and enhances the chance for entity federations in the ABAC system.

By automatically reverse-engineering an existing RDBMS enterprise application and then applying some classifications, the proposed research found that we can figure out the near-optimal subset of attributes that are most likely to be involved in ABAC model permissions addressing. Cardinality rank of both entities and attributes, Reference Power rank of entities, Entity Significance lifetime, and Entity State Stability play an essential role in determining the dominance level of both entities and attributes, which help in choosing the most candidate attributes for the ABAC model. Relying on the calculated Dominance Level rank according to the proposed formulas leads to an acceptable accuracy level in attribute selection. We can achieve an accuracy level reaching 80% without any overhead. Accuracy can be improved to 90%, with overhead impacts not exceeding 100%.

Involving the proposed reverse-engineering approach in the extraction and analysis of attributes and entities of an existing RDBMS is very promising for automated ABAC attribute extraction. One of the proposed approach's main pros is the attributes extraction process's automation feature. The research also provides a range of accuracy and overhead, enabling the implementing team to merely trade-off between accuracy and overhead according to the most appropriate for each situation. It also categorizes the results of attribute extraction according to core ABAC attributes categorization.

We recommend further work based on this research's output to extend the model to: consider the inherited relationships between different entities. More classification criteria could be applied and tested against the reverse-engineered RDBMS database to enhance the accuracy and minimize the overhead, such as considering the data type and size of attributes and the purpose of each attribute.

References

1. Al-Saraireh, J. An Efficient Approach for Query Processing over Encrypted Database. – J. Comput. Sci., Vol. **13**, 2017, No 10, pp. 548-557. DOI: 10.3844/jcssp.2017.548.557.
2. Al-Saraireh, J. A Novel Approach for Query over Encrypted Data in Database. – Int. J. Inf. Comput. Secur., Vol. **11**, 2019, No 6. DOI: 10.1504/IJICS.2019.103083.
3. Mulimani, M., R. Rachh. Analysis of Access Control Methods in Cloud Computing. – Int. J. Educ. Manag. Eng., Vol. **7**, May 2017, No 3, pp. 15-24. DOI: 10.5815/IJEME.2017.03.02.
4. Ferraiolo, D. F., R. Chandramouli, V. C. Hu, D. R. R. Kuhn. A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications. – NIST Spec. Publ., October 2016. DOI: 10.6028/NIST.SP.800-178.
5. Hu, V. C., D. R. Kuhn, D. F. Ferraiolo. Attribute-Based Access Control. – Computer (Long. Beach. Calif)., Vol. **48**, February 2015, No 2, pp. 85-88. DOI: 10.1109/MC.2015.33.
6. Hu, V., D. F. Ferraiolo, D. R. Kuhn, R. N. Kacker, Y. Lei. Implementing and Managing Policy Rules in Attribute Based Access Control. – In: Proc. of 16th IEEE Int. Conf. Inf. Reuse Integr (IRI'15) October 2015, pp. 518-525. DOI: 10.1109/IRI.2015.98.
7. Kerr, L., J. Alves-Foss. Combining Mandatory and Attribute-Based Access Control. – In: Proc. of Annu. Hawaii Int. Conf. Syst. Sci., Vol. **2016-March**, March 2016, pp. 2616-2623. DOI: 10.1109/HICSS.2016.328.
8. Riad, K., Z. Yan, H. Hu, G. J. Ahn. AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing. – In: Proc. of IEEE Conf. Collab. Internet Comput (CIC'15), March 2016, pp. 28-35. DOI: 10.1109/CIC.2015.38.
9. Alohaly, M., H. Takabi, E. Blanco. Automated Extraction of Attributes from Natural Language Attribute-Based Access Control (ABAC) Policies. – Cybersecurity, Vol. 2, December 2019, No 1, pp. 1-25. DOI: 10.1186/S42400-018-0019-2/TABLES/8.
10. Alohaly, M., H. Takabi, E. Blanco. Towards an Automated Extraction of ABAC Constraints from Natural Language Policies. – IFIP Adv. Inf. Commun. Technol., Vol. **562**, 2019, pp. 105-119. DOI: 10.1007/978-3-030-22312-0_8.
11. Narouei, M., H. Takabi. A Nature-Inspired Framework for Optimal Mining of Attribute-Based Access Control Policies – In: Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST. Vol. **305**. 2019, pp. 489-506. DOI: 10.1007/978-3-030-37231-6_29.
12. Liu, J. Y., B. Bin Jia. Combining One-vs-One Decomposition and Instance-Based Learning for Multi-Class Classification. – IEEE Access, Vol. **8**, 2020, pp. 197499-197507. DOI: 10.1109/ACCESS.2020.3034448.
13. Penelova, M. Hybrid Role and Attribute Based Access Control Applied in Information Systems. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 3, pp. 85-96.
14. Zhang, Z., K. Qian, B. W. Schuller, D. Wollherr. An Online Robot Collision Detection and Identification Scheme by Supervised Learning and Bayesian Decision Theory. – IEEE Trans. Autom. Sci. Eng., Vol. **18**, July 2021, No 3, pp. 1144-1156. DOI: 10.1109/TASE.2020.2997094.
15. Gupta, M., R. Sandhu, T. Mawla, J. Benson. Reachability Analysis for Attributes in ABAC with Group Hierarchy. – IEEE Trans. Dependable Secur. Comput., January 2022, No 1, pp. 1-15. DOI: 10.1109/TDSC.2022.3145358.

16. A g h i l i, S. F., M. S e d a g h a t, D. S i n g e l é e, M. G u p t a. MLS-ABAC: Efficient Multi-Level Security Attribute-Based Access Control Scheme. – Futur. Gener. Comput. Syst., Vol. **131**, June 2022, pp. 75-90. DOI: 10.1016/J.FUTURE.2022.01.003.
17. A m e e r, S., J. B e n s o n, R. S a n d h u. An Attribute-Based Approach toward a Secured Smart-Home IoT Access Control and a Comparison with a Role-Based Approach. – Information, Vol. **13**, 2022, No 2, pp. 1-33. DOI: 10.3390/info13020060.

Received: 28.06.2022; Second Version: 09.10.2022; Accepted: 14.10.2022