A Novel Access Control Strategy for Distributed Data Systems

Jianying Zhang\textsuperscript{1}, Xiukun Wang\textsuperscript{1}, Hongbo Liu\textsuperscript{1,2}, and Jun Meng\textsuperscript{1}

\textsuperscript{1} School of Computer, Dalian University of Technology, Dalian, 116023, China.
\textsuperscript{2} School of Information, Dalian Maritime University, Dalian, 116026, China.
\{zhangjy,jsjwxk,lhb,mengjun\}@dlut.edu.cn

Abstract. It is one of the most important challenges to balance between security and scalability in large distributed data systems. In this paper, we introduce a new data distribution model, which is a generalized tree structure called as multitree. And its access control strategy is investigated. In our multitree data model, the database schema is expressed as a schema graph, and a database instance is imagined as a data graph. It is different from the traditional hierarchical data model, since a node in our multitree model can have many parent nodes. All the data graphs or schema are transformed into multitrees. The complex data relation of the distributed data systems is reduced based on graph theory. The complexity of distribution is decreased significantly. In the multitree model, each user has a maximum access range corresponding to its multitree. It is integrated naturally with security. We use organization structure to bound the data range that a user can access, and use roles to restrict the operations that the user can perform. The scalability of data distribution and access control administration are evaluated through the instance adapted from the TPC-C database. The results illustrates our data distribution model is helpful for the system to be resilient and scalable. It is suitable for large distributed system and cloud relational database.

Keywords: Multitree, Data distribution, Multi-hierarchical model, Access control, Cloud database

1 Introduction

Implementing distributed application systems, the data model and access control have to ensure the distribution is reliable, scalable, and secure enough. It is extremely difficult for relational database to design the distributed data systems. Most of large scale storage systems are non-relational key/value databases, such as eBay Odyssey [1], Yahoo PNUTShell [2], Google BigTable [3, 4], Amazon Dynamo [5], and Amazon SimpleDB. In these systems, scalability, consistency, availability and partition tolerance properties are commonly desired. In addition to all the properties, a high-quality distributed data systems must take security into account, especially in cloud computing [6–11]. But enhancing system security usually weakens significantly its scalability and openness. To balance between
security and scalability/openness, many access control approaches are proposed. Sandhu et al [12] presented the RBAC96 model, in which the concept of hierarchy is introduced into roles. And organization structures are also formed into role hierarchies. However, organization structures are not suitable to be implemented directly as roles, since these are naturally administrative domains. Oh et al [13, 14] introduced the ARBAC02 model. In their model, organization structure are used to define user and permission pools with a refined prerequisite condition specification. So they are independent of roles and role hierarchies. However, Gilbert and Lynch [15] proved theoretically that it is impossible to achieve all the properties in current distributed systems. Pritchett [16] proposed the BASE (Basically Available, Soft state, Eventually consistent) model, which is diametrically opposed to ACID (Atomicity, Consistency, Isolation, Durability). Instead of standard SQL, private API (Application Programming Interface) was the main interfaces. The model hardly provides complex queries, integrity constraints and joins, all of which have to be completed through complex programming.

In this paper, we introduce a data distribution model, which is based on data multitrees, namely semantic clusters of relational data. And the access control strategy of the data model is discussed in detail. In our data distribution model, the database schema is expressed as a schema graph, and a database instance is imagined as a data graph. Tuples are nodes in the data graph, and references between these tuples are directed edges. We introduce a generalized tree structure called as multitree [17, 18], in which a node can have many parent nodes. It is different from the traditional hierarchical data model, since the traditional hierarchical data model suffers from the limitation of only single root. All the data graphs or schema are transformed into multitrees. If circuits and diamonds [17] can be reduced or removed from the database schema graphs, the produced data graphs are data multitrees. Even if schema graphs and data graphs contain circles or diamonds, both are also globally imagined as multitrees. Since the granularity of multitrees is coarser than that of fragments, the complexity of distribution is decreased significantly. In the multitree model, each user has a maximum access range corresponding to its multitree. So it is integrated naturally with security. Our access control is refined into two parts, namely operation access control and data access control. The multitree model and its access control strategy are helpful to broken through the dilemma between scalability and security control.

The rest of the paper is organized as follows. Related works about the data distribution model and access control strategy are reviewed in Section 2. In Section 3, we introduce a multitree model for distributed data systems. And its access control strategy is presented in Section 4. The performance of our approach is evaluated in Section 5. Finally conclusions and some future works are given in Section 6.
2 Related works

In the distributed data system, the basic unit is the most important prerequisite of data distribution. Bachman [19] analyzed how to partition processes and data in distributed environments as a critical issue in distributed systems. Moreover, the distribution of data is more critical [20] than that of processes. Data distribution involves what and where to distribute each basic unit. Previous work of data distribution focused on file allocation problem [21]. Multiple copies [22], single copy [23, 24], decomposition technique based heuristic algorithms [25], programs and files allocation [26] were investigated in this literature. Data allocation problem was reported in [27, 28, 26]. Relation model were typically assumed in distributed database design. Structured database decomposition [29], horizontally partitioning data [30], vertical partitioning [31] were also presented. Links were used to form a propagation of partitioning [32]. Optimal horizontal fragmentation [32], database partitioning in a cluster of processors [20], the complexity of the data allocation problem [33], distribution design of databases on a higher-order data model [34] were also introduced.

However, for the real-world application systems, it is difficult to define reasonably these data distribution units [35]. Fragments are not a proper unit for data allocation since the granularity is too fine to be controlled. And the complexity of optimized distribution computation is NP-complete [36]. Some branch-and-bound and heuristic algorithms were integrated in these allocation algorithms. But it results in the high cost of computation and management. What’s more, building a distributed database involves not only performance, but also other factors have to be taken into account, such as physically holding, matching between database size and computer power [19], and existing matured management modes. So researchers worldwide made attempts to achieve more practical and comprehensive data distribution models for the data distribution and access control, including the hierarchy structure model [12–14]. But the traditional hierarchical data model has its own limitations because of the tree structure. In the mean time, access control of large-scale distributed database systems is a challenging open problem [37]. It is difficult for these models to cope with the requirements of hundreds of roles and thousands of users. Usually organization structure [38, 39] was taken into account in many RBAC models, such as RBAC96 [40, 41], ARBAC02 [14], and UARBAC [37]. These organization structure were realized as special roles, and administrative domains were defined based on role hierarchies [42]. However, using role hierarchies as a basis for defining administrative domains is problematic in real-life scenarios, since the criteria for defining role hierarchies and administrative domains are different [37]. Administration domains are mostly defined based on the organizational structure, while roles are often defined based on job functions. Organization is implemented as an independent entity, as well as role, user and permission [43, 14], which makes it better to use organization structures controlling the data resource of the enterprises. And permission was not separately understood in the data systems. Permission is defined by Sandhu [12] that “A description of the type of authorized interactions a subject can have with an object”. Alter-
natively, the access control includes two aspects, one is what types of operation
can do, and the other is the data range the operation can deal with. However,
current RBAC models took both aspects as a whole (namely permission) instead
of taking into account individually. In this paper, we investigate a generalized
multitree data distribution model and its access control strategy.

3 Data Distribution Model

3.1 Relationship of Servers

A large distributed database provides scalable data service through many servers.
The relationship between these servers has to facilitate scale-out or scale-in.
First and foremost, data should be distributed or merged conveniently, and by
which condition to separate the highly interrelated data is the prerequisite issue.
Usually a large distributed database involves many companies, institutions, de-
partments, sectors, user groups, districts, divisions and categories with certain
relations. These organizations have similar missions or complementary functions,
and an organization may be managed by several superior organizations. It is rea-
sonable to separate or merged the data according to the organization structures.
Although hashed declustering, round-robin, range-partition declustering also can
be choose to improve the performance, instance security is not considered. But
it is the key issue of the systems in practical applications.

Assuming that each organization has a dedicated server of its own, each can
normally map to a data multitree. The relationship of the corresponding servers
is just like that of respective organizations since a distributed system normally
builds on many such organizations. So the multitree is its natural structure
representing these servers. The server multitree provides a resilient architecture.
A superior server covers an inferior server in the server multitree, which means
the superior organization administrate the superior organization. When a new
organization join in or withdraw from the distributed system, a dedicated server
can be correspondingly added to or detached from the system. In practice, some
organizations share a server with others, consequently, the server is mapped to
a combined data multitree of these inferior organizations.

3.2 Data Organization & Data Control

In our model, a multitree is the basic unit of distribution instead of fragment. It
is imagined as the data resource including controlled private data and referenced
data from surroundings. The multitree with organization structures is directly
implemented as the data range of data access control.

Definition 1. Such a data multitree of a relational database is a multitree
$MT_d(r, e_d)$ that (1) $r$ is the set of tuples of the database, (2) a directed edge
$r_{ik} \rightarrow r_{jl}$ of $e_d$ denotes tuple $r_{ik}$ referring to $r_{jl}$, where $r_{ik}$ is from relation $R_i$,
and $r_{jl}$ is from relation $R_j$. 
The vertex set $r$ contains four kinds of data. In this set, a few controller nodes can dominate the cluster of tuples. The set of tuples $r$ is a quadruplet $(A, B, C, D)$ where $A$ is a set of controllers’ ancestral nodes, $B$ is a set of baseline nodes in a multitree, $C$ is a set of controller data nodes, and $D$ is a set of controlled data nodes. Figure 1 illustrates these kinds of tuples of data multitrees. In the data multitrees, tuple nodes $a_1, a_2, b_1, b_2, \ldots, g_4$ and $g_5$ are from relations $A, B, C, D, E$ and $F$ respectively. Figure 1(a) is the multitree of whole database, and Figure 1(b) is a sub multitree with controller node $d_1$. In Figure 1(b), nodes $c_1, c_2, g_1, g_2, g_3$ and $g_4$ are from controlled nodes; nodes $a_1$ and $b_1$ are controllers’ ancestral nodes; nodes $f_1, f_2$ and $c_1$ are baseline nodes. A more practical example using TPC-C database is put forward in section 5.

In the multitree such as Figure 1(b), controller data node, controllers’ ancestral nodes and baseline nodes are data referenced by controlled data. So transactions can be locally executed in the server, where the data multitree settled. A server can update, delete and insert the controlled nodes which belong to the cluster. The host server doesn’t modify directly or locally controllers’ ancestral nodes and baseline nodes, for they are only used conveniently as controlled node’s reference. If a server updates non-controlled nodes, other servers must be involved in. Updating request to this non-controlled data is transmitted upward the related servers along the multitree. The update is completed in the server, and the modification is replicated downward, then the original site is updated. The server only updates its controlled data. The data organization structure are applied to localize databases and simplify data distribution. When the network connection between a server and others is failure in distributed systems, local applications can still query enough referenced data. The controlled data also can be updated, deleted or inserted if the corresponding baseline existed. The global applications would launch after network is renovated.
3.3 Multitree Replication

In traditional distributed databases, segments of relations are replication units. However, the granularity is too fine for the massively distributed databases, since many related segments should be replicated together. The referenced tuples should be replicated accordingly or else some field values, for instance sequence number, may be inapprehensible. Moreover data replications across these servers should keep consistent, weak consistency or strict consistency. All settlements for these requirements are depending on each application in traditional distributed databases. Consequently, a suit of data distribution or data replication methods independent of applications is needed. The data distribution model based on multitree will be presented in another paper. Our architecture imitates practical organizations with mature management mode. In this architecture, data that organized into the data multitree, is separated into its basic unit. They are updated in the host server, and the update be propagated to other related servers. If other organizations need locally access the data controlled by an organization, the data multitree of the origination can be replicated to the server of the organization.

There are several comments about data multitree replication. Firstly in distributed circumstances, some organizations may be reluctant to expose their private data to other organizations’ servers, then the target site need have rights to get the replications. Secondly, instead of the whole data multitree, maybe only controlled data are actually replicated since its referenced data maybe have already existed in the target site. Thirdly the usage of the replications depends on the isolation levels of transactions. For instance, if the transaction chose the strictest isolation level serializable, though local replication existed, the data must be accessed from the original data multitree to eliminate data inconsistencies.

4 Multitree-based Access Control

4.1 Users & regions

Many organizations have deployed large database systems, and there are various users [44]. If all users and corresponding access rights are centrally managed, the overhead would be very high. In our model, security is also distributively managed. Each organization manages its own users’ privileges. A user in one organization can only deal with the operations constrained in his or her organization. This kind of access control is mandatory. Besides this kind of access control, a user can have specific rights associate with the roles. Before discussing the access control between inter-organizations, we provide the related definitions.

**Definition 2.** Basic region. *The range of a multitree controlled by a set of given controller nodes C is defined as a basic region \( R_C \), and region for short.*

A basic region is a data multitree resident in a single server. The relationship of basic regions is still multi-hierarchical based on multitrees. In the region multitree, a region is a node, and relations of coverage are taken as links. A region
may cover and be covered by several other regions just like an organization do. Coverage means users in the superior organizations may manage not only its own organization’s data but also the information of its covered organizations.

**Definition 3.** Extended region. A basic region with all under basic regions in the region multitree forms into a whole. This union region is defined as an extended region $R_C'$ of the basic region $R_C$. Apparently, a basic region can be an extended region of its own, provided no other basic regions under its own.

In our model, regions and servers are distinct. In previous sections, we have ignored the difference for simplicity. A region is a logical concept, while a server is a physical one. On one hand, many regions can coexist in a same physical server/site. For instance, in some distributed environments, some organizations can share a server to reduce investment. Then the regions corresponding to all these organizations locate in the same server. And, not only basic regions but also extended regions can share a same physical server. On the other hand, an extended region can often be hosted on several physical servers. An extreme case is a basic region reside on several physical servers, what is more like a parallel processing. We illustrate the relationship between regions and servers in Figure 2. The servers $S_1, S_2, \ldots, S_{11}, S_{1213}, \ldots$, basic regions $S_2, S_4, S_{12}$ with their extended regions $S_2', S_4', S_{12}'$ are explicitly marked. Server $S_{1213}$ hosts basic region $S_{12}$ and $S_{13}$. Extended region $S_2'$ and $S_4'$ span on many a server. There is a diamond in the extended region $S_2$, and two paths exist between $S_2$ and $S_8$.

![Figure 2. Server & region](image)

**Definition 4.** User’s region. A database user $u$ must be mapped to a region, which may be a basic region $R_C$ or an extended region $R_C'$, just as a user must be attached to an organization. $\text{User's region}(u : \text{USER} \rightarrow \text{REGIONS}$, denotes the mapping of user $u$ onto the user’s associated region. The region is the user’s maximum access scope. And the user $u$ may fully access the controlled data while
can only read non controlled data in the corresponding multitree of the region. A region $R_1$ can be the inferior region of the region $R_2$, namely $R_1 \subseteq R_2$

**Definition 5.** Virtual user. Assuming that a region owns a user account $u_R$ of a superior region $R$, the region can create a virtual user of $R$ by mapping an extended user $u$ to $u_R$. Then, in the region or its inferior regions, $u$ is identity. But outside current region, it is taken as user $u_R$.

There exist three kinds of users. A local user $LU$ is limited to the basic region, whereas an extended user $EU$ can access the corresponding extended region. The third kind is a virtual user $VU$. For a region, one problem to be solved is creating users for its inferior regions. The region administrator has difficulties in creating and managing many users for each inferior region. It is difficult that root regions create every users who can access these regions, so does other regions. Our model provides the access control transparency, in which a superior region need only create one user account and grant it to each direct inferior region.

According to Definition 5, the virtual users of a superior region can be generated in inferior regions. The superior region only need to create one user for each direct inferior region, what simplify security administration of the superior region. And the superior region doesn’t have to know detail usages of the user account. In the inferior regions, those virtual users can be differentiated to satisfy requirements of access control or audition by using different local user identities.

Virtual users can access superior regions, but they may have partial privileges of these regions. The privileges can also be granted by current inferior region. Available privileges are obtained from superior regions, and part of these rights may assigned to specific virtual users. When access request is sent to the superior region server, only part of privileges are sent by the current region. Then the virtual user’s are restricted to partial privileges. Additionally, the range of a virtual user can also be restricted to a smaller one instead of the whole region.

In our model, the maximum range that a user can access is its own region. This kind of access control is mandatory. The user’s ultimate rights involve other constrains, such as role based rights, user’s rights, and application rights.

### 4.2 Multitree-Based access control model

**Definition 6.** The core multitree-Based access control is defined as follows:

- **REGIONS**, **USER**, **ROLES**, **OPS**, and **MT**(regions, users, roles, operations, multitree, respectively)
- **REGION** is an administrative domain, it is defined as $\text{REGION}(MT, \text{USERS, ROLES, OPS, UR, RA, UA})$, where **OPS** are SQL operations(select, insert, update, etc) on the relations of the data multitree $MT$, and $\text{UR} \subseteq \text{USERS} \times \text{ROLES}$, a many-to-many mapping between users and roles(user-to-role assignment relation), and $\text{UA} \subseteq \text{USERS} \times \text{OPS}$, a many-to-many mapping between users and **OPS**(user-to-operation assignment relation), and $\text{RA} \subseteq \text{ROLES} \times \text{OPS}$, a many-to-many mapping between roles and **OPS**(role-to-operation assignment relation).
- **Assign users**: \((r : ROLES) \rightarrow 2^{\text{USERS}}\), the mapping of role \(r\) onto a set of users. Formally: \(\text{assign users}(r) = \{u \in \text{USERS} \mid (u, r) \in UR\}\).

- **Assign operation permissions to role**: \((r : ROLES) \rightarrow 2^{\text{OPS}}\), the mapping of role \(r\) onto a set of operations. Formally: \(\text{assign operation permissions to role}(r) = \{o \in \text{OPS} \mid (r, o) \in RA\}\).

- **Assign operation permissions to user**: \((u : \text{USERS}) \rightarrow 2^{\text{OPS}}\), the mapping of user \(u\) onto a set of operations. Formally: \(\text{assign operation permissions to user}(u) = \{o \in \text{OPS} \mid (u, o) \in UA\}\).

- **REGION** can include or be included by other regions. This is, the Multitree \(MT_1\) of a region \(R_1\) is a sub multitree of a multitree \(MT_2\) that corresponds to one of \(R_1\)'s superior region \(R_2\), denoting \(MT_1 \subseteq MT_2 \iff R_1 \subseteq R_2\). Apparently, both the inclusion relation for data multitrees and for regions are transitive.

- For a region and its parent regions, **ROLES** sets, as well as **OPS** and **RA**, are distinct. They can normally be same, since the organization may has common management mode. This can help local administrators make access control policy by inheriting from parental regions' policies, or extending child regions' policies.

- **access**: \(\text{USER} \times \text{OPS} \times \text{REGIONS} \rightarrow \text{BOOLEAN}\)

- **access**\((u, op, \text{reg}) = 1\) if user \(u\) can perform operation \(op\) in region \(\text{reg}\), 0 otherwise.

**Property 1. Access authorization for local users.** In a region, a local user can perform an operation \(op\) on the data multitree \(MT\) only if there exist a role \(r\) that is included in the role set **ROLES** and there exist a permission that is assigned to \(r\) to authorize the performance of \(op\) on \(MT\), or there exist a permission that is directly assigned to the user to authorize the performance of \(op\) on \(MT\).

\[
\text{access}(u, op, \text{reg}) \Rightarrow \exists u : \text{USER}, op : \text{OPS}, r : \text{ROLES} \mid u \in \text{region.USER} \land u.USER.TYPE = \text{LOCAL} \land op \in \text{region.OPS} \land (r \in \text{region.ROLES} \land (u, r) \in \text{region.UR} \land (r, op) \in \text{region.UA} \lor (u, op) \in \text{region.UA})
\]

**Property 2. Access authorization for extended users.**

A extended user of a region can perform available operations on its inferior regions.

\[
\text{access}(u, op, \text{reg}) \Rightarrow u.USER.TYPE \neq \text{EXTENDED} \land \text{region.MT} \subseteq \text{superior region.MT} \land \text{access}(u, op, \text{superior region}).
\]

**Property 3. Access authorization for virtual users.**

A virtual user of a region may perform operations in the region on behalf of his true identity.

\[
\text{access}(u, op, \text{reg}) \Rightarrow u \in \text{sub region.USERS} \land u.USER.TYPE \neq \text{VIRTUAL} \land u.MAPING.USER = u. \land \text{sub region.MT} \subseteq \text{region.MT} \land \text{access}(u, op, \text{region}).
\]

In summary, a user \(u\) can access operation \(op\) in region \(\text{region}\) only if he or she is authorized as a local user of \(\text{region}\), or an extended user of the superior regions, or a virtual user created in the inferior regions.

\[
\text{access}(u, op, \text{region}) \Rightarrow
\]
\[ u : \text{USER}, \text{op} : \text{OPS}, r : \text{ROLES} \in \text{region.USER} \land u.\text{USER_TYPE} = '\text{LOCAL}' \land \text{op} \in \text{region.OPS} \land (r \in \text{region.ROLES} \land (u, r) \in \text{region.UR} \land (r, \text{op}) \in \text{region.UA}) \land (u.\text{USER_TYPE} = '\text{EXTENDED}' \land \text{region.MT} \subseteq \text{superior.region.MT} \land \text{access}(u, \text{op}, \text{superior.region.UA})) \land (u \in \text{sub.region.USER} \land u.\text{USER_TYPE} = '\text{VIRTUAL}' \land u.\text{MAPING.USER} = u.\text{v} \land \text{sub.region.MT} \subseteq \text{region.MT} \land \text{access}(u.\text{v}, \text{op}, \text{region})) \]

5 Evaluation and Discussion

In this section, we evaluate the multitree model through the data distribution and access control administration. The instance is adapted from TPC-C [45]. It is a wholesale supplier with a number of distributed warehouses and sales districts. Each regional warehouse covers some districts. And each district serves many customers. All warehouses maintain stocks for large number of items sold by the Company.

![Fig. 3. TPC-C Schema](image-url)
According to the schema shown in Figure 3, we select relation Warehouse as controller relation. All data nodes except those of relation Item are controlled by the data nodes from relation Warehouse directly or indirectly. Each Warehouse corresponds to a real warehouse, which may has its own region and resides at its own server. In addition, there exists a top region in the system. It controls the whole data multitree including the regions of ten Warehouse. In Figure 4, region Warehouse2 covers two independent region district1 and district2, which the other districts in the Warehouse forms into the basic region of warehouse2. Whereas, the extended region of warehouse2 still hold district1 and district2. Warehouse3 and warehouse4 form into one basic region.

Each basic region can normally reside in one server. For example, an independent server named Top Server is required to accommodate the top region. And there are other combinations. Region Warehouse2, District1 and district2 can reside in one server, or in three independent servers.

Figure 5 is a user administration sample from the regions of the system. A user only can access the region which is shown in field Region. Field Grant to indicates which region does the user account granted to login, whether current basic region, the extended region or an inferior basic region. For instance, TR_user1 is a local user and grant to the basic region TR only. TR_user2 is an extended user, and is granted to region WH1 (Warehouse1). User’s region can be an inferior region, for example, user TR_user5’s region.

Virtual users can issue more global transactions than other users of the same region. For example, virtual user WH2_user4 is created in Warehouse2. And it can access the whole region. If the network is available, inferior regions forward WH2_user4’s transaction requests to the top region. The server of the top region receives these transactions, deals with it, and forwards to the servers of its inferior regions. After executions at each region involved, results are collected on the original server reversing request paths. For local users and extended users, transactions can complete in servers of their own regions. According to TPC-C’s specification, most transactions can be executed in basic regions or extended regions. Even if the network connecting to superior regions fails, these local executions may still succeed.
It is a resilient & scalable architecture. On one hand, regions of two different Warehouse can form into a new region to scale in the system by united both data multitees. On the other hand, a new inferior region can be built by extracting a multitee from parent regions to scale out the system. For example, a District can independently manage its own data by extracting the corresponding multitee from his parent Warehouse’s region, and setting up a new server to manage the extracted multitee. It apparently alleviates the workload of the parent server.

It is also a trusted architecture. If a warehouse or district does not want put it’s private data on the cloud, it can build a server of his own, connecting to the cloud database. In this server, the entry point can be fully controlled, access control, audit can all be implemented.

6 Conclusions & Future works

In this paper, multi-hierarchical data model was introduced, and the basic distribution unit is multitee instead of fragments of relations. All the data graphs or schema are transformed into multitees. The complex data relation of the distributed data systems can be reduced based on graph theory. The complexity of distribution is decreased significantly. In the multitee model, each user has a maximum access range corresponding to its multitee. Each user is mapped to a region, then the access range is limited to the region. The evaluation results illustrates the data distribution model is resilient and scalable. Through virtual users, distributed security adminstration can be simplified. Role-based access control and audit are compatible with the constraint. So it provides integrated security. Organization structures were used to bound the data range that a user can access, and use roles to restrict the operations that the user can perform. Our access control was refined into operation access control and data access control.
Besides scalability and security, relevant issues, such as schema integration, data multitree migration, concurrency control, data consistency, cloud database recovery, and distributed transaction isolation level need to be further investigated. The multitree based data distribution could be applied in cloud database, parallel databases, mobile databases, wireless sensor network and other massively distributed computing areas. Our data model and control strategy would be helpful to achieve the goal of easily scalable and manageable, broadly applicable, multi-tenant relational systems, which provide elastic, efficient, globally available, safe, confidential and extremely robust distributed data schema for real-world applications.

Acknowledgments

The first author would like to thank Hong Yu, Nanhai Yang for their scientific collaboration in this research work. This work was supported by National Natural Science Foundation of China (Grant No.60873054), Youth Elite Teacher Fund (2009QN043).

References


design,” in Proceedings of the 1982 ACM SIGMOD international conference on
31. S. Navathe, S. Ceri, G. Wiederhold, and J. Dou, “Vertical partitioning algo-
rithms for database design,” ACM Transactions on Database Systems, vol. 9, no. 4,
32. S. Ceri, S. Navathe, and G. Wiederhold, “Distribution design of logical database
1983.
33. P. M. G. Apers, “Data allocation in distributed database systems,” ACM Trans-
34. H. Ma, K.-D. Schewe, and Q. Wang, “Distribution design for higher-order data
35. E. Cecchet, G. Candea, and A. Ailamaki, “Middleware-based database replication:
the gaps between theory and practice,” in Proceedings of the 2008 ACM SIGMOD
international conference on Management of data, (New York, NY, USA), pp. 739–
752, ACM, 2008.
36. K. P. Eswaran, “Placement of records in a file and file allocation in a computer
network,” in Proceedings of the ZFZP Congress on Information Processing, (North-
37. N. Li and Z. Mao, “Administration in role-based access control,” in Proceed-
sings of the 2nd ACM symposium on Information, computer and communications
38. J. D. Moffett, “Control principles and role hierarchies,” in Proceedings of the third
ACM workshop on Role-based access control, (New York, NY, USA), pp. 63–69,
39. J. D. Moffett and E. C. Lupu, “The uses of role hierarchies in access control,” in
Proceedings of the fourth ACM workshop on Role-based access control, (New York,
40. R. Sandhu, V. Bhamidipati, E. Coyne, S. Ganta, and C. Youman, “The arbac97
model for role-based administration of roles: preliminary description and outline,”
in Proceedings of the second ACM workshop on Role-based access control, (New
41. R. Sandhu, V. Bhamidipati, and Q. Munawer, “The arbac97 model for role-based
administration of roles,” ACM Transactions on Information and System Security,
42. J. Crampton and G. Loizou, “Administrative scope: A foundation for role-based
administrative models,” ACM Transactions on Information and System Security,
43. A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte,
A. Miege, C. Saurel, and G. Trouessin, “Organization based access control,” in Pro-
ceedings of IEEE 4th International Workshop on Policies for Distributed Systems
44. G. Berhe, L. Brunie, J. Pierson, and B. Pascal, “Content Adaptation in distributed
multimedia system,” Journal of Digital Information Management, vol. 3, no. 2,
pp. 95–100, 2005.
45. TPC, “Tpc benchmark. c standard specification revision 5.9.”