Record-Based Access Control Mechanism for a Database

Li Li

Dept. of Computer Science
Oregon State University
Corvallis, Or 97331
lli@eecs.orst.edu

Abstract

Current database systems apply access control mostly on tables and columns. However, many applications require access control on *individual rows* in database tables. Therefore, we have implemented a *row-based* access control mechanism. In our model for access control, *object groups*, in addition to *user groups*, are used to reduce the number of *access rights* to be defined. In this report, we describe the relational tables storing information for access control and explain how `select`, `update`, `delete` and `insert` operations are performed under access control. We also explain how access rights are managed.

Acknowledgements

I would like to express my sincere appreciation to my major professor, Dr. Minoura, for his valuable ideas and constructive criticism. I am grateful for his endless patience and continued faith shown in me throughout the lifecycle of this project.

I also would like to thank my family, my husband, Rongkun Shen, my Mom and my sweet son, for their love and support during the implementation of the project and writing of this report. The encouragement of my family enables me to finish this project.

Table of Contents

1. Introduction

In many database applications, allowing every user to access all the data is not desirable. Therefore, an access control mechanism restricts access to data by users. Traditional access control mechanisms [1] for databases are *table- and column-based*, where each user is allowed to perform only certain operations, such as *insert, select, update, and delete* operations, on certain tables and columns. However, in many applications, a user should be limited to access only certain *rows* in tables. In this project, we have implemented an access control mechanism *applicable to individual rows* in tables.

A simple way for protecting a row of data in a table is to specify for each row a set of users that can access it. This approach works for a small database with a small group of users. However, when a database is large and shared by many users, the table storing the *access rights* of the users on rows becomes too large and cannot be maintained efficiently. One solution to this problem is to use user groups, as in the Unix system [2], where access rights are assigned to *user groups* as well as to individual users. A user on a Unix system can be a member of one or more user groups.

In addition to user groups, we introduce *object groups.* An object group can contain multiple objects. If a user groups is granted access rights on an object group, every user in the user group inherits those access rights on each object in the object group. When objects are grouped, we can manage access rights more efficiently than when access rights are assigned to individual users.

In Section 2, we introduce our access-control model. Section 3 describes the design of the relational tables used to store the data for access control. We will discuss in Section 4 how `select, update, delete,` and `insert` operations are performed with access control. Section 5 and 6 explain how access rights are managed. In Section 7, we explain the procedures used for access control. Section 6 concludes this report.

2. Access Control Model

In our *access control* mechanism, we use *user groups* and *object groups* to reduce the number of *access rights* to be defined. A *user group* is a common idea used in operating systems such as Windows and Unix.

In addition to *user group*s, we introduce *object groups* because we can manage access rights more efficiently by granting them to groups of objects rather than by doing so to individual objects.

The schema diagram for our *access control* mechanism is shown in Figure 1. Entity types `ac_user`, `ac_object_group`, `ac_object`, and `ac_user_group` represent users, user groups, objects, and object groups, respectively. `ac_user_group_membership` represents the membership relationships between users and user groups. `ac_right` is the relationship type associating `ac_user_group` and `ac_object_group`. `access permission`, `ownership`, `insert permission` are attributes defined for `ac_right`. `access permission` specifies *read* or *write* permission.

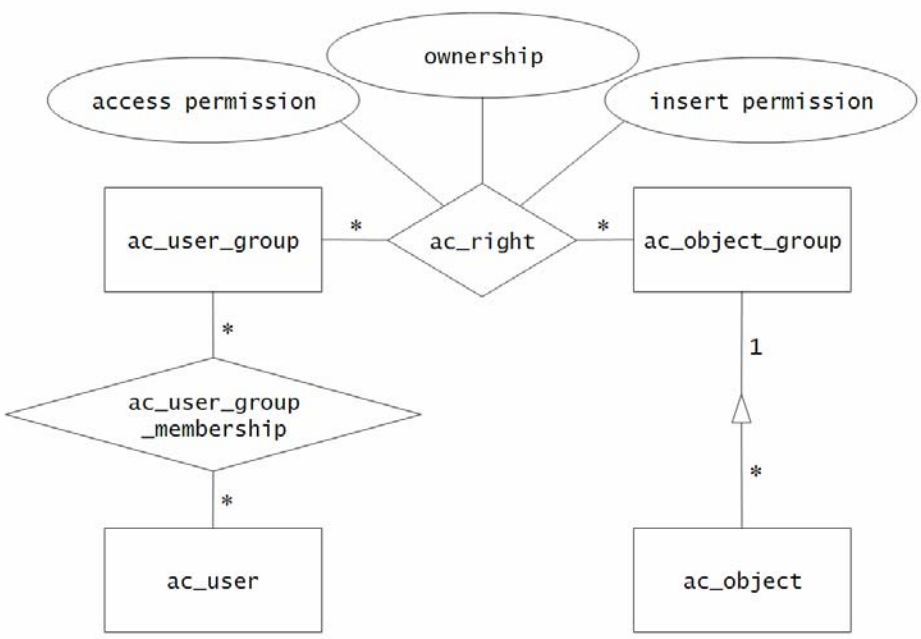

Figure 1: Schema diagram for access control.

We now state the rules for *access control*.

1. A *user group* is a set of *users*. A *user* may be a member of multiple *user group*s.
2. An *object* is a *table* or a *row* of a table.
3. An *object group* is a set of objects. An object can be a member of at most one *object group*.
4. If *right* r is provided for *user group Ug* on *object group Og*, then any user *u* in *Ug* has right *r* on any

object *o* in *Og*. A right is a *read permission*, a *write permission, insert permission*, or an *ownership*. When an ownership is defined between *Ug* and *Og*, we also say that each member of *Ug* owns *Og*. The rights defined between *Ug* and *Og* can be represented by a 5-tuple (*Ug*, *Og*, *a*, i, *os*), where *Ug* is a *user group, Og* is an *object group, a* is either *read permission, write permission, or null, I is insert permission*, and *os* is *ownership*.

5. A user with a *read permission* on an object can read the object. If the object is a *table*, the user can read any *rows* in the table.
6. A user with a *write permission* on a row of a table can *read*, *update*, and *delete* it. If a *write permission* is on a table, the user can *read, update*, and *delete* any rows in the table.
7. A user with an *insert permission* on a table can insert new rows into the table.
8. An *owner* of an *object group* can add new rights on that *object group* and delete existing rights on that *object group*. Those rights may be defined for any *user group*s.
9. By default, every user belongs to *user group* `PUBLIC`.

According to rule 8, if a user with an ownership of an *object group* grants the ownership to another user, that user also becomes an owner of the *object group*. There can be *multiple* owners for each *object group*. One of the objects in an *object group* acts as a *group leader* that representing the *object group*. The other objects in this object group are group members.

There are two steps in the *access control* mechanism. The first step is *authentication*. We use the ID and the password of a user to check the identity of the user. The second step is *authorization,* where the access control mechanism decides whether or not a validated user is allowed to perform a requested action.

3. Relational Tables for Access Control

We now describe the design of the relational tables that store the information for *access control*. The information for each user is stored in table `ac_user` as shown in Figure 2, where we have four users `u1`, `u2`, `u3`, and `u4`. Each user has a default user group, which initially contains only that user at the time when the user create a user account in database.

ac_user_id	ac_user_name	password	first_name	last_name	state	city	addressline1
1	u1	12345	aa	bb	OR	AAl	Some place 1
2	u2	12345	cc	dd	OR	BB	Some place 1
3	u3	12345	ee	ff			Some place 1
4	u4	12345	dd	gg			Some place 1

addressline2	zip	telephone	fax	email	default_ac_user_group_id
Some place 2	97330	1234567	1234567	lli@cs.orst.edu	1
Some place 2	97330	7654321	7654321	u2@cs.orst.edu	4
Some place 2	97330	1111111	1111111	u3@cs.orst.edu	5
Some place 2	97330	2222222	2222222	u4@cs.orst.edu	6

Figure 2: A simple example of table ac_user.

The information for each *user group* is stored in table ac_user_group as shown in Figure 3, where we have three user groups Ug1, Ug2, and Ug3.

ac_user_group_id	ac_user_group_name
1	Ug1
2	Ug2
3	Ug3

Figure 3: A simple example of table ac_user_group.

We use table ac_user_group_membership to maintain information on user group memberships as shown in Figure 4, where u1 and u2 are in user group 1, u1 and u3 in user group 2, and u4 in user group 3.

ac_group_membership_id	ac_user_id	ac_user_name	ac_user_group_id
1	1	u1	1
2	2	u2	1
3	1	u1	2

4	3	u3	2
5	4	u4	3

Figure 4: A simple example of table ac_user_group_membership.

The information for each object is stored in table ac_object as shown in Figure 5. The column ac_object_type stores the types of objects, value "r" for a row and value "t" for a table. The column table_name stores the table name. The column row_name indicates the column that contains the primary keys of the row objects. The column row_id stores the primary key value of each row object. Only row objects need row_name and row_id. In this example, object 1 is the first row in table crop whose primary key value in column crop_id is 1. Each object group is identified by the ID of its group leader. Therefore, Object 1 and object 2 are rows in table crop, belonging to object group designated by object 1. Object 3 is the table crop in object group designated by its group leader, object 3.

ac_object_id	ac_object_type	table_name	row_name	row_id	ac_object_group_leader_id
1	r	crop	crop_id	1	1
2	r	crop	crop_id	2	1
3	t	crop			3

Figure 5: A simple example of table ac_object.

Table ac_right stores information on access rights defined between *user group*s and *object groups* as shown in Figure 6. A value of column access_permission can be r for *read permission*, w for *write permission* or null. A value of column ac_insert is true or false indicating whether *insert permission* is granted or not. A value of the column is_owner is true or false.

Ac_permission_id	ac_user_group_id	ac_object_group_leader_id	ac_permission	ac_insert	is_owner
1	1	1	r	false	true
2	2	3	null	false	false
3	3	3	w	true	true

Figure 6: A simple example of table `ac_right`.

According to table `ac_right` shown in Figure 6, user group 1 owns object group 1 and has *read permission* on it. From table `ac_user_group_membership`, both `u1` and `u2` are in user group 1, and from table `ac_object`, object group 1 contains objects with `ac_object_id` 1 and 2, which we call `object` 1 and `object` 2. Therefore, we infer that `u1` owns `object` 1 and `object` 2 and has *read permission* on them, but does not own or have any access permission on object 3. We now summarize that the access rights in our example as the permission matrix shown in Figure 7:

Object / User	Object 1	Object 2	Object3
u1	r/o	r/o	null
u2	r/o	r/o	
u3			null
u4			w/i/o

Figure 7: Permission matrix.

4. Access Control for Operations on Data

We now describe how the *access control* mechanism is enforced for the operations performed on a database.

The *access control* mechanism for a `select` operation is implemented by query modification, in which an additional condition that restricts the data to be retrieved is added to the where clause of the SQL `select` statement. An `update`, or `delete` statement is performed only when the target object is updatable by the current user. An `insert` statement is performed when the current user has an *insert permission* on the target table.

4.1 Access Control for a Select Operation

Assume that an authenticated user issues an SQL select statement of the form

`select` *select-list* `from` *from-list* `where` *condition;*

The query modification mechanism then adds to this query as the additional condition access restriction for *access control*:

`select` *select-list* `from` *from-list* `where` *condition* `and` *access-restriction;*

For an example, consider the following SQL statement that retrieves crop records containing "`Corn`" in crop names.

```
select *
from crop c
where c.name like '%Corn%';
```

If this query is issued by user `u1`, the SQL statement is modified as follows

```
select *
from crop c, ac_object o, ac_right r, ac_user_group_membership m, ac_user u
where c.name like '%Corn%'
and ((o.table_name='crop' and o.ac_object_type='t') or
     (o.table_name='crop' and o.ac_object_type='r' and c.cropid = o.row_id))
and o.ac_object_group_leader_id = r.ac_object_group_leader_id
and (r.ac_permission ='w' or r.ac_permission ='r')
and r.ac_user_group_id = m.ac_user_group_id
and m.ac_user_id = u.ac_user_id and u.ac_user_name ='u1';
```

First, tables `ac_object`, `ac_right`, `ac_user_group_membership`, and `ac_user`, are added to the `from` clause.

Second, the factors following `c.name like '%Corn%'` are for *access control*. The factor

```
((o.table_name='crop' and o.ac_object_type='t') or
 (o.table_name='crop' and o.ac_object_type='r' and c.cropid = o.row_id))
```

makes sure that current object is table `crop` or is a row in table `crop`. If the object is a table, `u1` should have a *read* or *write permission* to the table. On the other hand, if the object is a row, `u1` should have *read* or *write permission* to the row.

Then the factors

1. `and o.ac_object_group_leader_id = r.ac_object_group_leader_id`
2. `and (r.ac_permission ='w' or r.ac_permission ='r')`
3. `and r.ac_user_group_id = m.ac_user_group_id`
4. `and m.ac_user_id = u.ac_user_id and u.ac_user_name ='u1';`

ensure that each combination of selected `o, r, m,` and `u` satisfies the following conditions.

1. Object `o` belongs to the current object group.
2. The access permission for the current user group on the current object group is '`r`' (readable) or '`w`' (writable).
3. The current user group is related to the current object grou*p*.
4. User `u1` belongs to the current user group.

4.2 Access Control for an Update or Delete Operation

When a user wants to update a row, he must have a *write permission* to that row, or a *write permission* to the table containing the row. User `u1` has *write permission* to the target row in table `crop` if the following SQL statement returns a result.

```
select *
from ac_user u, ac_user_group_membership m, ac_right r, ac_object o
where u.ac_user_name ='u1'
and u.ac_user_id = m.ac_user_id
and m.ac_user_group_id = r.ac_user_group_id
and r.ac_permission = 'w'
and r.ac_object_group_leader_id = o.ac_object_group_leader_id
and ((o.table_name='crop' and o.ac_object_type='t') or
       (o.table_name='crop' and o.ac_object_type='r' and o.row_id = $cropid)) ;
```

The *access control* mechanism for a `delete` operation is similar to an `update` operation, except that after the deletion of a row, we should delete the entry for that row object from table `ac_object` as well. The following `delete` SQL statement can be used for this purpose.

```
delete from ac_object o
where o.table_name='crop' and o.ac_object_type='r' and o.row_id = $cropid
```

Furthermore, we need to check whether the object to be deleted is a *group leader* or not. If it is, the object group it represents should be deleted as well. And all the members in this object group should be deleted from both table `ac_object` and their resptively table storing detail data fro each member. The following SQL statement is used to find out the `ac_object_group_leader_id` for the object group whose *group leader* is the object to be deleted.

```
select o.ac_object_id, o.ac_object_group_leader_id
from ac_object o
where o.table_name='crop' and o.ac_object_type='r' and o.row_id = $cropid
and o.ac_object_id = o.ac_object_group_leader_id;
```

Then, we delete all the group members in this object group using the following SQL statement.

```
delete from ac_object o
where o.ac_object_group_leader_id = $ac_object_group_leader_id;
```

Finally, we need to delete all the access rights related to this object group.

```
delete from ac_right where ac_object_group_leader_id = (select
ac_object_group_leader_id from ac_object where row_id = $primary_key and
table_name = '".$table_name."')";
```

4.3 Access Control for an Insert Operation

When insertion of a row into a table is attempted, the *access control* mechanism must check first whether the user has *insert permission* on that table or not. If the user has the permission, the `insert` operation can be executed. If the object representing the new row is a *group leader*, a new *object group* should be created. On the other hand, when the object is not a group leader, it should be added to the correct object group. We represent the object groups using a tree structure. The root nodes of the tree stand for the object group leaders and the nodes in the lower hierarchical levels stands for group members in the object groups delegated by the group leaders.

The tree view given in Figure 9 contains three kinds of nodes, which are nodes for hillslope, nodes for rotation, and nodes for crop. On a field, a farmer may have his rotation of crops. So crops are associated with a rotation, and rotations are associated with a field. A farmer should be allowed to access the records

for his fields, rotations and crops. In this case, a field node can be used as a group leader representing the farmer's fields, rotations, and crops. A hillslope is often a farm field. Therefore, among the three kinds of nodes, hillslope nodes are root nodes and group leaders.

In Figure 9, the node `Yolo Farm,` is a hillslope and acts as a *group leader.* The object group representing by this node contains other group members, rotation node `yolo tomato-tomato-corn,` and crop nodes `yolo processing tomatoes`, `yolo processing tomatoes`, and `yolo corn 150 bu.`

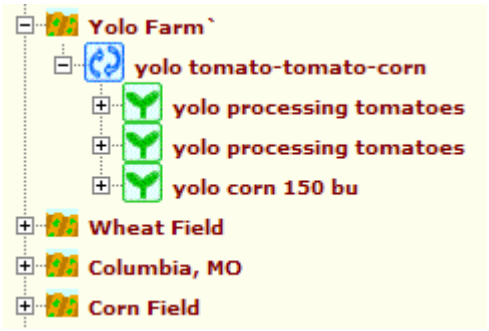

Figure 9: Group leaders and group members.

4.3.1 Inserting a Group Leader

When a *group leader* node is inserted, a new *object group* should be created to contain this newly inserted object. Suppose we want to insert a row into table `hillslope,` which is a table storing hillslop data entries.
There are four steps involved in executing an `insert` operation for a group leader object.

1. The *access control* mechanism checks whether a user has an *insert permission* to the target table or not. Notice that an *insert permission* can only be applied on a table object. If the query result is non-empty with the following SQL statement, user `u1` has an *insert permission* on the target table.

```
select *
from ac_user u,  ac_user_group_membership m, ac_right r, ac_object o,
where u.ac_user_name ='u1'
```

```
and u.ac_user_id = m.ac_user_id
and m.ac_user_group_id = r.ac_user_group_id
and r.ac_insert = 1
and r.ac_object_group_leader_id = o.ac_object_group_leader_id
and o.table_name='hillslope' and o.ac_object_type='t';
```

2. After u1 passes the *insert permission* check-up, he can insert a new *hillslope* into table hillslope using the following insert SQL statement:

```
insert into hillslope values (nextval('hillslope_id'_seq), 20, 20, 'field1',
'OR', 20, 'some shape', 20, 20);
```

where $next_hillslope_id = nextval('hillslope_id'_seq);

3. Since the new object is a *group leader* node in the data tree, we should add it into table ac_object, setting it as a group leader for a new object group.

```
insert into ac_object values ($next_ac_object_id, 'r', $next_ac_object_id ,
$next_hillslope_id, 'hillslope',  'hillslopeid');
```

where $next_ac_object_id = nextval('ac_object_id'_seq)

4. In the last step, the *access rights* are defined between the default user group for u1 and the new object group. Each user has a default user group when his account is created in the database. The default user group initially contains only that user as a member. By default, write permission, insert permission and ownership are granted initially. The access rights can be redefined later by the owner of the object group.

```
insert into ac_right values ( $next_ac_object_id , (select
default_ac_user_group_id from ac_user where ac_user_name = 'u1') , 'w',1, 1);
```

4.3.2 Inserting a Group Member

Suppose we want to insert a row into table crop. This is a group member node. Its group leader node is a a hillslope the crop associates with. There are three steps involved in executing the insert operation.

1. We check whether user u1 has *insert permission* to the target table or not by using the following SQL

statement.

```
select *
from ac_user u, ac_user_group_membership m, ac_right r, ac_object o
where u.ac_user_name ='u1'
and u.ac_user_id = m.ac_user_id
and m.ac_user_group_id = r.ac_user_group_id
and r.ac_insert = 'i'
and r.ac_object_group_leader_id = o.ac_object_group_leader_id
and o.table_name='crop' and o.ac_object_type='t';
```

2. u1 inserts a new crop into table crop as follows.

```
insert into crop values ($next_crop_id, 20, 20, 2,'new wheat', 'test')
```

where $next_crop_id = nextval('crop_id'_seq);

3. After the new row is inserted, we should add it to the object group containing the current *group leader*. But in the implementation, the group leader hillslope is in the same object group with its children node rotation, who again is the new node crop's parent node. It is easier for us to retrieve from roation, the parent node of the new crop, the information about the current ac_object_group_leader_id. We use the following SQL statement to store the corresponding information such as object_type, table_name, row_id, and row_name of new crop into table ac_object.

```
insert into ac_object values (nextval('ac_object_id'_seq), 'r', $cropid,
(select o.ac_object_group_leader_id
     from ac_object o
     where o.table_name='rotation' and o.ac_object_type='r' and o.row_id =
     $crop_rotationid),
      'crop', 'cropid');
```

5. Access Control for Access Rights

Ownership can be passed on from one user to another user. With the ownership, a user can redefine the access right to an object group for the current user group he belongs to.

Assume that user u1 wants to grant the ownership on the target object group Og1 to another user u2. The *access control* mechanism checks whether he is in a user group that owns Og1 or not. If so, u1 can add u2 into the user group that owns Og1. In the following SQL statement, if the query result is non-empty, u1 is in the user group which owns the target object group.

```
select *
from ac_user u, ac_user_group_membership m, ac_right r
where u.ac_user_name ='u1'
and u.ac_user_id = m.ac_user_id
and m.ac_user_group_id = r.ac_user_group_id
and r.is_owner = 1
and r.ac_object_group_leader_id =$object_group_leader_id;
```

To grant another user the ownership on the current object group, we use the following SQL statement.

```
insert into ac_user_group_membership values
(nextval('ac_user_group_membership_id'_seq), 2,
        select ac_user_group_id from ac_user_group where ac_user_name =
        $user_group_name);
```

To add new rights or delete existing rights on the current *object group*, we use the following SQL statement.

```
update ac_right
set ac_permission = 'r'
where ac_object_group_leader_id = $object_group_leader_id
      and ac_user_group_id in (select m.ac_user_group_id
                               from ac_user_group_membership m, ac_user u
                               where u.ac_user_name ='u1'
                               and u.ac_user_id = m.ac_user_id ));
```

6. Access Control for Administrator

An administrator has full access control defined for ordinary users, in addition, an administrator can create a new user account in table ac_user, and create a new user group in table ac_user_group.

7. Procedures for Access Control

To modularize our implementation of the access control mechanism, we implemented a PHP script called *security.phtml* containing procedures that perform operations needed for access control. In the following list, we list the name and the parameters of each procedure. We then give a brief explanation of each procedure.

Method: `check_write_permission_on_row(user_name, table_name, primary_key)`

Returns `true` if the user with `user_name` has write permission on the target row specified by `table_name` and `primary_key`. `table_name` is the name of the table containing the row, and `primary_key` is the primary key value of that row.

Method: `check_insert_permission(user_name, table_name)`

Returns `true` if the user with `user_name` has insert permission on the table specified by `table_name`.

Method: `check_ownership(user_name, group_leader_id)`

Returns `true` if the user with `user_name` owns the object group designated by `group_leader_id`.

Method: `insert_ac_object(object_type, table_name, primary_key_column, primary_key, group_leader_id)`

Inserts the new object designated by `table_name`, `primary_key_column`, and `primary_key` into table `ac_object`. `object_type` specifies the type of the object, either `r` for a row or `t` for a table. If the new object is a table object, `table_name` specifies the name of the table, and `primary_key_column` and `primary_key` are `null`s. If the new object is a row object, `table_name` specifies the name of the table containing the row. `primary_key_column` is the name of the column containing the primary key of the row, whose primary key value is passed by `primary_key`. The new object belongs to the object group whose group leader is specified by `group_leader_id`. If the object to be inserted is a group leader, `group_leader_id` is `null`.

Method: `delete_all_group_members(group_leader_id)`

Deletes all the objects whose group leader is group_leader_id.

Method: delete_ac_object(table_name, primary_key_column, primary_key)

Deletes the object specified by the table_name, primary_key_column, and primary_key. If object is a table, primary_key_column, and primary_key are null.

Method: is_group_leader(table_name, primary_key_column, primary_key)

Returns true if the object specified by the table_name, primary_key_column, and primary_key is a group leader, otherwise, false.

Method: update_ac_right(user_name, user_group_name, group_leader_id, access_right, ownership, insert_permission)

Updates the access rights for the user group specified by user_group_name on the object group whose group leader is designated by the group_leader_id. The user specified by user_name must own the object group. access_right, ownership, and insert_permission specify the rights to be updated.

Method: insert_ac_right(user_name, user_group_name, group_leader_id, access_right, ownership, insert_permission)

Inserts the access rights for the user group specified by user_group_name on the object group whose group leader is designated by the group_leader_id. The user specified by user_name must own the object group. access_right, ownership, and insert_permission specify the rights to be inserted.

Method: delete_ac_right(user_name, user_group_name, group_leader_id,)

Deletes the access rights for the user group specified by user_group_name on the object group whose group leader is designated by the group_leader_id. The user specified by user_name must own the object group.

The following procedures can be performed only by the administrator.

Method: create_user(ac_user_name, password, first_name, last_name, state, city, addressline1, addressline2, zip, telephone, fax, email,

default_ac_user_group_id)

Creates a new user with the parameters such as ac_user_name, password, first_name, last_name, state, city, addressline1, addressline2, zip, telephone, fax, email. Initially, the new user belongs to the default user group specified by default_ac_user_group_id.

Method: create_user_group(ac_user_group_name)

Creates a new user group with user group name specified by ac_user_group_name.

Method: add_user_to_group(user_name, ac_user_group_id)

Associates the user with user_name with user group specified by ac_user_group_id.

Method: remove_user_from_group (user_name, ac_user_group_id)

Deletes the user with user_name from the user group specified by ac_user_group_id.

8. Conclusions

The table- and column-based access control is supported by current database management systems. However, row-based access control is needed for many applications. A row-based access control can restrict a user to access only those rows that she is authorized to do so. We implemented a row-based access control subsystem in this project.

The main features in our access control model are *user groups* and *object groups*. *User groups* are commonly used in operating systems, such as Windows and Unix. A *user* may be a member of multiple *user group*s. An *object* is a *table* or a *row* of a table. An o*bject group,* which is a set of objects, is a new concept introduced by our model in order to manage *access rights* more efficiently.

Access rights are assigned between a user group and an object group, so that every user in the user group inherits those access rights on every object in the object group. One of the objects in an *object group* acts as a *group leader* that represents the object group. A user with an ownership of an *object group* can redefine the access rights on the object group for any user group. User groups and object groups can

reduce the number of *access rights* to be defined and hence the storage space for storing access rights.

More future work can be adding a relational table between `User` and `Object group`, so that access rights on object groups can be defined directly for individual users. For `insert` or `delete` operation on an object, after it is inserted or deleted, we should perform maintenance for access right, such as creating a new object group for the new object or adding the newly inserted object into the correct object group.

9. References

[1] Ramakrishnan, Raghu and Johannes Gehrke, *Database Management Systems*, 2nd edition. McGraw-Hill, 2000

[2] Mark G Sobell, *A practical Guide to the Unix System*, 3rd edtion. ADDISON-WESLEY

[3] M. Stonebraker and E.Wong, Access control in a relational database management system by query miodification. In *proceedings of the 1974 annual conferences*, pages 180-186. ACM Press, 1974

[4] http://www.microsoft.com/sql/default.asp

[5] http://www.oracle.com/solutions/security/Privacy9i.pdf.

10. Appendix

Codes will be appended.

```
<?php
include_once("common.phtml");

function build_part_SQL_for_selection($user_name, $table_name, $primary_key_column){
global $db;

$sSQL = ", ac_object o, ac_right r, ac_user_group_membership m, ac_user u ".
  "where o.table_name='"
  .$table_name.
  "' and( o.ac_object_type='t' or (o.ac_object_type='r' and o.row_id = "
  .$table_name
  ."."$primary_key_column." ))
  and o.ac_object_group_leader_id = r.ac_object_group_leader_id
and (r.ac_permission ='w' or r.ac_permission ='r')
and r.ac_user_group_id = m.ac_user_group_id
and m.ac_user_id = u.ac_user_id and u.ac_user_name ='".$user_name."'";

return $sSQL;
}
function check_write_permission_on_row($user_name, $table_name, $primary_key){
global $db;

$sSQL = "select *
                        from ac_user u, ac_user_group_membership m, ac_right r, ac_object o
                        where u.ac_user_name ='".$user_name."'
                        and u.ac_user_id = m.ac_user_id
                        and m.ac_user_group_id = r.ac_user_group_id
                        and r.ac_permission = 'w'
                        and r.ac_object_group_leader_id = o.ac_object_group_leader_id
                        and o.table_name='".$table_name."' and(o.ac_object_type='t' or
(o.ac_object_type='r' and o.row_id = ". $primary_key." ))" ;
   $db->query($sSQL);
     $nrows = $db->num_rows();
    if ($nrows < 1)
                        return false;
    else
                        return true;

}
function check_insert_permission($user_name, $table_name){
global $db;

    $sSQL = "select *
                        from ac_user u, ac_user_group_membership m, ac_right r, ac_object o
                        where u.ac_user_name ='".$user_name."'
                        and u.ac_user_id = m.ac_user_id
                        and m.ac_user_group_id = r.ac_user_group_id
                        and r.insert_permission = 1
                        and r.ac_object_group_leader_id = o.ac_object_group_leader_id
                        and o.table_name='".$table_name."'and o.ac_object_type='t'" ;
```

```
                            $db->query($sSQL);
     $nrows = $db->num_rows();
    if ($nrows < 1)
                        return false;
    else
                        return true;
}

function check_ownership($user_name, $group_leader_id){
global $db;

    $sSQL = "select *
                        from ac_user u, ac_user_group_membership m, ac_right r
                        where u.ac_user_name ='".$user_name."'
                        and u.ac_user_id = m.ac_user_id
                        and m.ac_user_group_id = r.ac_user_group_id
                        and r.is_owner = 1
                        and r.ac_object_group_leader_id = ".$group_leader_id ;

     $db->query($sSQL);
     $nrows = $db->num_rows();
    if ($nrows < 1)
                        return false;
    else
                        return true;
}

function insert_ac_object($object_type, $primary_key, $parent_table_name, $parent_primary_key,
$table_name, $primary_key_column){
global $db;
if ($parent_primary_key == "") //insert a group leader
{/* 1.find out the current user
2. query its default user-group
3. object group leader is nextval('ac_object_id_seq')
4. use default access rights initially
*/
   $sql_query = "SELECT nextval('ac_object_id_seq');";
    $next_value = $db->query($sql_query);
    $ac_object_id = pg_fetch_result($next_value, 0, 0);

     $sql_query = "SELECT default_ac_user_group_id from ac_user where ac_user_name =
'{$_SESSION['user_name']}';";
    $next_value = $db->query($sql_query);
    $default_ac_user_group_id = pg_fetch_result($next_value, 0, 0);

    $sSQL = "insert into ac_object values ( $ac_object_id, '"
    . $object_type. "', $primary_key, $ac_object_id, '".$table_name."', '"
    . $primary_key_column."')" ;
      $db->query($sSQL);
```

```
        $sSQL = "insert into ac_right values( nextval('ac_right_id_seq'),
$ac_object_id ,$default_ac_user_group_id,'w', 1, 1 )";
 $db->query($sSQL);
 //insert_right($default_ac_user_group_id, $ac_object_id, "w", 1 , 1);
}
else // insert a group member
{$sSQL = "insert into ac_object values (nextval('ac_object_id_seq'), '"
    . $object_type. "','". $primary_key.", (select ac_object_group_leader_id from ac_object where
table_name = '".$parent_table_name."' and row_id = '". $parent_primary_key."'), '".$table_name."',
'"
    . $primary_key_column."')" ;
   $db->query($sSQL);
                        }
}

function delete_ac_object ($table_name, $primary_key)
{
                        if(is_group_leader($table_name, $primary_key)){
                           // is a group leader
                        delete_right_for_object_group_leader($table_name, $primary_key);
                        delete_all_group_members ($table_name, $primary_key);
                        delete_row_object_from_ac_object ($table_name, $primary_key);

                        }else{// is a group member

                           delete_row_object_from_ac_object ($table_name, $primary_key);

                        }
}

function is_group_leader ($table_name, $primary_key){
global $db;

    $sSQL = "select * from ac_object where row_id =". $primary_key. " and table_name = '".
$table_name. "' and ac_object_id = ac_object_group_leader_id";
    $db->query($sSQL);
    $nrows = $db->num_rows();
    if ($nrows < 1)
                        return false;
    else
                        return true;

}

function delete_all_group_members ($table_name, $primary_key){
global $db;

    $sSQL = "select * from ac_object where ac_object_group_leader_id = (select
ac_object_group_leader_id from ac_object where row_id = $primary_key and table_name =
'".$table_name."')";
```

```
     $db->query($sSQL);
     $nrows = $db->num_rows();
    for ($i =0; $i <$nrows; $i++) {
      $db->next_record();
                        $delete_fields[$i]['table_name'] = trim($db->f("table_name"));
                        $delete_fields[$i]['row_id'] = trim($db->f("row_id"));
                        $delete_fields[$i]['row_name'] = trim($db->f("row_name"));
                        }
    for ($i =0; $i <$nrows; $i++) {
                        $sSQL = "delete from ". $delete_fields[$i]['table_name']." where ".
$delete_fields[$i]['row_name']." = ".$delete_fields[$i]['row_id'];
                        $db->query($sSQL);
                        }


    $sSQL = "delete from ac_object where ac_object_group_leader_id = (select
ac_object_group_leader_id from ac_object where row_id = $primary_key and table_name =
'".$table_name."')";
    $db->query($sSQL);

}

function delete_row_object_from_ac_object ($table_name, $primary_key){
global $db;

    $sSQL = "delete from ac_object where table_name = '".$table_name."' and ac_object_type = 'r'
and row_id = ". $primary_key;
  $db->query($sSQL);

}

function delete_right_for_object_group_leader($table_name, $primary_key){
global $db;

    $sSQL = "delete from ac_right where ac_object_group_leader_id = (select
ac_object_group_leader_id from ac_object where row_id = $primary_key and table_name =
'".$table_name."')";

  $db->query($sSQL);
}

function update_ac_right($user_name,  $user_group_name, $group_leader_id, $access_right,
$ownership, $insert_permission){
if ( check_ownership($user_name, $group_leader_id)){
  update_right($user_group_name, $group_leader_id, $access_right, $ownership,
$insert_permission);
  return 1;
  }
  return 0;
}
```

```
function update_right($user_group_name, $group_leader_id, $access_right, $ownership,
$insert_permission){
global $db;

    $sSQL = "update ac_right set";
    if ($access_right != null)
    {                   if ($sSet != "")
                            $sSet .=", ";
                        $sSet = $sSet."ac_permission = ".$access_right;
    }
    if ($is_owner != null)
    {                   if ($sSet != "")
                            $sSet .=", ";
                        $sSet = $sSet."is_owner = ".$ownership;
    }
    if ($insert_permission != null)
    {                   if ($sSet != "")
                            $sSet .=", ";
                        $sSet = $sSet."insert_permission = ".$insert_permission;
    }
    if ($sSet != "") {
    $sSet = " set (" . $sSet. ")";
  }
    $sWhere = "where ac_object_group_leader_id = ".$group_leader_id." and ac_user_group_id = ".
$user_group_name;

 $sSQL = $sSQL.$sSet.$sWhere;
  $db->query($sSQL);

}

function insert_ac_right($user_name,  $user_group_name, $group_leader_id, $access_right,
$ownership, $insert_permission){
if ( check_ownership($user_name, $group_leader_id)){
  insert_right($user_group_name, $group_leader_id, $access_right, $ownership,
$insert_permission);
  return 1;
  }
  return 0;
}

function insert_right($user_group_name, $group_leader_id, $access_right, $ownership,
$insert_permission){
global $db;

    $sSQL = "insert into ac_right values (nextval('ac_right_id_seq'), $group_leader_id ,(select
ac_user_group_id from ac_user_group where ac_user_group_name ='".$user_group_name."'), '".
    $access_right . "',$ownership, $insert_permission)";
```

```
  $db->query($sSQL);
}

function delete_ac_right($user_name, $user_group_name, $group_leader_id){
if ( check_ownership($user_name, $group_leader_id)){
  delete_right($user_name, $user_group_name, $group_leader_id);
  return 1;
  }
  return 0;
}


function delete_right($user_group_name, $group_leader_id){
global $db;

    $sSQL = "delete from ac_right where ac_user_group_id = (select ac_user_group_id from
ac_user_group where ac_user_group_name =".$ac_user_group_name.") and ac_object_group_leader_id
= ".$group_leader_id;

  $db->query($sSQL);
}

function add_user_to_user_group($user_name, $user_group_name){
global $db;

    $sSQL = "insert into ac_user_group_membership values
(nextval('ac_user_group_membership_id_seq'), (select ac_user_group_id from ac_user_group where
ac_user_group_name = '". $user_group_name ."'), (select ac_user_id from ac_user where ac_user_name
= '". $user_name."'))";

  $db->query($sSQL);
}

function remove_user_from_user_group($user_name, $user_group_name){//may be deleted later
global $db;

    $sSQL = "delete from ac_user_group_membership where ac_user_id = (select ac_user_id from
ac_user where ac_user_name = '". $user_name ."') and ac_user_group_id = (select ac_user_group_id
from ac_user_group where ac_user_group_name = '". $user_group_name."'))";

  $db->query($sSQL);
}

function remove_user_group($user_group_id){
global $db;

    $sSQL = "delete from ac_user_group_membership where ac_user_group_id = $user_group_id";

  $result = $db->query($sSQL);
```

```
   $sSQL = "delete from ac_user where default_ac_user_group_id = $user_group_id";

  $result =$db->query($sSQL);

  $sSQL = "delete from ac_user_group where ac_user_group_id = $user_group_id";

  $result =$db->query($sSQL);
  return $result;
}

function create_user($ac_user_name, $password, $first_name, $last_name, $state, $city,
$addressline1, $addressline2, $zip, $telephone, $fax, $email, $default_ac_user_group_name){
global $db;

   $sSQL = "insert into ac_user values (nextval('ac_user_id_seq'), '".$ac_user_name. "', '".
    $password."', '". $first_name."', '". $last_name."', '". $state."', '".$city."',
'".$addressline1."', '".
    $addressline2."', '". $zip."', '". $telephone."', '". $fax."', '". $email."',
    (select ac_user_group_id from ac_user_group where ac_user_group_name = '".
    $default_ac_user_group_id."'))";
  $db->query($sSQL);
}

function create_user_group($ac_user_group_name){
global $db;
echo $ac_user_group_name;
   $sSQL = "insert into ac_user_group values (nextval('ac_user_group_id_seq'),
'".$ac_user_group_name."')";
  $db->query($sSQL);
}

function delete_user_group($ac_user_group_name){
global $db;

   $sSQL = "delete from ac_user_group where ac_user_group_name = '".$ac_user_group_name."'";
  $db->query($sSQL);
}

function delete_user($ac_user_name){
global $db;

   $sSQL = "delete from ac_user where ac_user_name = '".$ac_user_name."'";
  $db->query($sSQL);
}
?>
```