CS590U

Access Control: Theory and Practice

Access Control in Databases

(October 30th)

Guest Lecturer: Prof. Chris Clifton

Outline

- The SQL grant-revoke mechanism
  - Griffiths & Wade. "An Authorization Mechanism for a Relational Database System". [TODS, 1976]
    - Discusses access control mechanism in System R
    - The Grant/Revoke mechanism remains essentially unchanged in today's database systems
- Virtual Private Databases in Oracle
  - Course Project Topic Description: Fine Grained Access Control in Databases

The Goal of [Griffiths & Wade]

- Permits users to selectively share data while retaining the ability to restrict data access in a multi-user database system

Overview of The Approach:

- The creator of a table is fully authorized to perform any actions on the table.
- The creator may explicitly grant to any other user any or all of his privileges.
- The grantor may specify that the user is authorized to further grant the privileges.
- A grantor may revoke the granted privileges.
- Views are used for granting access to row and column subsets.

Two Types of Relations

- Base relations (physically stored)
  - e.g., EMPLOYEE(NAME, SALARY, MANAGER, DEPARTMENT)
- Views (a virtual, dynamic window on the data base)
  - e.g., DEFINE VIEW AS
    SELECT NAME, SALARY
    FROM EMPLOYEE
    WHERE DEPARTMENT = 'TOY'

Privileges on a Relation

- READ: use the relation in a query, e.g., to read tuples, or to define views
- INSERT: insert rows
- DELETE: delete rows
- UPDATE: modify existing data
  - may be restricted to a subset of the columns of the table
  - some views may not be updatable
- DROP: delete the entire table

The Syntax for Granting Permissions

- A user executes the following command:
- GRANT

  ALL RIGHTS
  | ⟨privileges⟩
  | ALL BUT ⟨privileges⟩

  ON ⟨table⟩
  TO ⟨user-list⟩
  [WITH GRANT OPTION]

7

The Implementation: Basic Version

- SYSAUTH
  - USERID: the user being authorized
  - TNAME: name of the table
  - TYPE: 'R' if a base relation, `V' if a view
  - A column for each of the privileges READ, INSERT ... (excluding UPDATE), containing a 'Y' to indicate the privilege is granted to the user
  - UPDATE: `ALL' (all columns), `NONE' (no update), or `SOME'
  - GRANTOPT: whether can be further granted

8

The Implementation: Basic Version (continued)

- For each table, a user has at most two tuples in SYSAUTH: one for grantable privileges, and one for nongrantable privileges
- SYSCOLAUTH
  - used if UPDATE is `SOME'
  - for each updatable column, a (user, table, column, grantor, grantopt) tuple is inserted into SYSCOLAUTH

9

Semantics of GRANT

- When a user issues a GRANT command, the set of privileges actually granted is the intersection of
  - the set of grantable privileges possessed by the grantor
  - and the set of privileges in the grant
- The effect of a GRANT is
  - to insert a new tuple into SYSAUTH (and several tuples into SYSCOLAUTH)
  - or to appropriately modify an existing tuple

10

An Example

- Let A be the creator of the table EMPLOEE, after the following two actions
  1. A: GRANT READ, INSERT ON EMPLOYEE TO B WITH GRANT OPTION
  2. B: GRANT READ, DELETE ON EMPLOYEE TO X
- X has READ privilege on EMPLOYEE
- X has no privilege if 1 and 2 are switched

11

Syntax for Revocation

- REVOKE

  ALL RIGHTS
  | ⟨privileges⟩
  ON ⟨table⟩
  FROM ⟨user-list⟩

12

Implication on Implementation

- Two tuples in SYSAUTH are insufficient to represent a user's rights on a table.
- The identity of the grantor must also be stored, i.e., up to two tuples for each distinct (grantee, table, grantor) combination.
- Revocation may affect more than two tuples.

13

Semantics of Revocation

- Let $G_1, G_2, \ldots, G_{i-1}, G_i, G_{i+1}, \ldots, G_n$ be a sequence of grant commands of one specific privilege on a given table by any user
  - Grants of several privileges are represented as a sequence of individual grants
- If a revocation R occurs, and G_i is the only one affected (same grantor, same user, same privilege), then the state of the authorization should be identical to the state after the sequence $G_1, G_2, \ldots, G_{i-1}, G_{i+1}, \ldots, G_n$

14

Implications

- One may make the same grant multiple times, one revoke statement revokes all of them.
- If a revokee possesses other grants of the revoked privilege from an *independent* source, then he retains these privileges.

15

Recursive Revocation

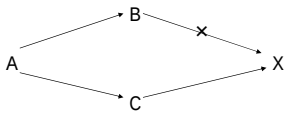
- Consider the following sequence:
  - A grants ALL RIGHTS to B with GRANT OPTION
  - B grants ALL RIGHTS to X
  - A revokes ALL RIGHTS from B
- This should be equivalent to
  - B grants ALL RIGHTS to X

  which has no effect
- Need to do recursive revocation

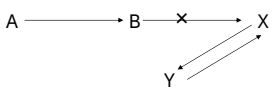
16

Independent vs. Dependent Sources

X should still have the privilege



X should not have the privilege



17

Algorithms for Revocation.

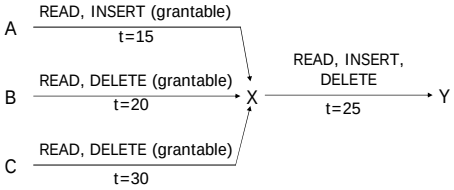
- One way is to remove the revoked grant and see whether a path of grant still exists
  - requires tracing potentially the whole grant graph to do so

18

Handling Revocation without Tracing the Grant Graph

- Modify the SYSAUTH table such that each privilege column contains a timestamp, rather than just 'Y' or 'N'.
- The timestamp is monotonically increasing and no two grants have the same time.
- After one removes the revoked grant, make sure that every outgoing grant has an incoming grant of earlier time to support it

19



At t=35, B issues REVOKE ALL RIGHTS FROM X.

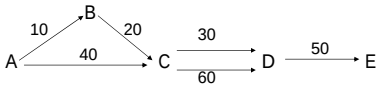
X's rights: READ{15,30} INSERT{15} DELETE{30}

X's grant to Y: READ{25} INSERT{25} DELETE{25}

X's grant to Y of DELETE at t=25 should also be revoked. READ and INSERT should be maintained

Additional Complications

- When A grants the same right twice to B, the later time should be stored in the table.



At t=70, B executes REVOKE FROM C; if C's grant to D at 60 is not recorded, then the grant from C to D is also removed and D has no right; this is inconsistent with the semantics.

21

Other Issues and Questions

- If A grants to X, X grants to Y, and A revokes from Y; A may want to make sure that Y does not have the right.
  - Solution: each right is given a label (presumably its source), at revocation time, all rights with that label are removed.
  - Question to think: shouldn't one need more than one labels on a grant to keep track of the path ?
- Question to think: Can one use the HRU access matrix model or the Take-Grant model to model grant/revoke?

22

Outline

- The SQL grant-revoke mechanism
  - Griffiths & Wade. "An Authorization Mechanism for a Relational Database System". [TODS, 1976]
    - Discusses access control mechanism in System R
    - The Grant/Revoke Mechanism remained essentially unchanged in today's database systems
- Virtual Private Databases
  - Course Project Topic Description: Fine Grained Access Control in Databases

23

The Virtual Private Databases in Oracle

- Also called Fine Grained Access Control and Row Level Security.
- Part of Oracle databases since Oracle8i release 8.1
- In a nutshell, the ability to attach, at runtime, a dynamically generated predicate (where clause) to any query issued against a database table or view.

24

More Details on VPD

- A security policy, which is a function (PL/SQL stored procedure), is created and associated with each table.
- The function, at query execution time, returns a predicate to be attached to the query.
  - typically, it makes use of values in an application context to determine the correct predicate to return.

25

Application Contexts

- An application context is a set of variable/value pairs
  - e.g., in the context named `App1', one may access variables `DeptNo', `Mgr', etc.
- An application context is always bound to some PL/SQL package, which is the only way for setting values in the context
  - this package may use database content to set the variables (using database content to protect)

26

Example 1

- The Policy: The RLS_ADMIN user can access anything; other users can only access the records they own.
  - the predicate `where owner = USER' should be returned for users other than the RLS_ADMIN

27

Implementation of Example 1

```
create function my_sec_func (p_schema in varchar2,
        p_object in varchar2) return
as
begin
   if ( user = 'RLS_ADMIN' ) then
        return '';
   else
        return 'owner = USER';
   end if;
end;
```

28

Example Policies VPD supports

- Employees may see only their records
- Managers may see their record and the records of people that work for them
- HrRep may see anyone in a given department

29

Advantages of VPD: Ease of Maintenance

- Implementing security policies in SQL Grant/Revoke results in
  - either a proliferation of views
    - a separate view is needed for every group of users
    - recreate a view each time the security policy changes
  - or implementing security in application logic
- VPD enables one to implement the security policy using one table and one procedure

30

Other Advantages

- Allows for easier application development
  - takes the security logic out of the application logic
- The security control is always there with the source of data (the table)
  - easy to enforce

31

Potential Research Issues With VPD (1)

- Non-declarative security policies
  - security policies are coded in PL/SQL programs.
  - may be difficult to understand
  - the effect of security policies are difficult to analyze
- A declarative, high-level approach to policy specification is more desirable
  - may be compiled into procedures
  - what should this language look like?

32

Potential Research Issues With VPD (2)

- Database content may be used to populate application contexts, thus affecting the security policies to protect the same database
  - e.g., the content of a Human Resource table may be used to protect the same table
- Question: can one gain access by changing some part of database first, then get access? How difficult to perform such analysis?

33

Potential Research Issues With VPD (3)

- How does VPD interact with other DB access control features such as grant-revoke, roles, views, and stored procedures with grant-execute?
- Without a high-level, coherent view of access control, access control specified using multiple mechanisms may interact in subtle and unexpected ways.

34

Potential Research Issues With VPD (4)

- Should the user know that the results from a query are not what she expected (because the query has been changed)?
- What happens with a complicated query involves multiple tables and recursive selects?

35

Have People Studied Some of Those Questions Before?

- Yes, 20+ years ago, but it seems there is not much follow-on work
  - Access Control in a Relational Data Base Management System by Query Modification. Michael Stonebraker and Eugene Wong. Proceedings of the 1974 annual ACM conference.
  - An authorization model for a shared data base. E. B. Fernandez, R. C. Summers, and C. D. Coleman. SIGMOD'1975.
  - Specification of Content-Dependent Security Policies. David L. Spooner. Proceedings of the 1983 annual ACM Conference on Computers.

36