Discretionary Access Control in System R

Philip W. L. Fong

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada

CPSC 525/625 (Fall 2016)
The DAC of System R has significantly influenced the DAC of subsequent DBMS, including the SQL standard.
Outline

1. Grant
2. Revocation
3. Recursive Revocation
4. Access Control Via Views
Permissions

- `select, insert, delete, update`
  - compared to the read, write, execute of UNIX
- Each of the above can be paired with a grant option
  - Receiving a privilege with grant option means that the receiving user can further propagate the privilege to other users.
GRANT { <privileges> | ALL [PRIVILEGES] } 
ON <relation> 
FROM { <users> | PUBLIC } [WITH GRANT OPTION];
GRANT { <privileges> | ALL [PRIVILEGES] } ON <relation> FROM { <users> | PUBLIC } [WITH GRANT OPTION];

- <privilege>
  - privileges to be granted
  - ALL and ALL PRIVILEGES indicate the granting of all supported privileges
GRANT { <privileges> | ALL [PRIVILEGES] } ON <relation> FROM { <users> | PUBLIC } [WITH GRANT OPTION];

- <privilege>
  - privileges to be granted
  - ALL and ALL PRIVILEGES indicate the granting of all supported privileges

- <relation>
  - name of relation
GRANT { <privileges> | ALL [PRIVILEGES] }  
ON <relation>  
FROM { <users> | PUBLIC } [WITH GRANT OPTION];

- **<privilege>**
  - privileges to be granted
  - **ALL** and **ALL PRIVILEGES** indicate the granting of all supported privileges

- **<relation>**
  - name of relation

- **<users>**
  - users to which granting applies
  - **PUBLIC**: grant to all users
GRANT { <privileges> | ALL [PRIVILEGES] } 
ON <relation>
FROM { <users> | PUBLIC } [WITH GRANT OPTION];

- <privilege>
  - privileges to be granted
  - ALL and ALL PRIVILEGES indicate the granting of all supported privileges

- <relation>
  - name of relation

- <users>
  - users to which granting applies
  - PUBLIC: grant to all users

- WITH GRANT OPTION
  - The granted privileges can be further granted to others
The **UPDATE** Privilege

- **select, insert, delete**
  - applies to whole relation
- **update**
  - `update(a_1, \ldots, a_n)`
  - `a_1, \ldots, a_n` are attributes
**Example**

Leo:  GRANT update(phone) ON Customers TO Marc;
Leo:  GRANT select ON Videos TO Beth, Gena WITH GRANT OPTION;
Gena: GRANT select ON Videos TO Matt;
Leo:  GRANT ALL PRIVILEGES ON Movies, Videos TO Helen WITH GRANT OPTION;
Helen: GRANT insert, select ON Videos TO Beth;
Outline

1. Grant
2. Revocation
3. Recursive Revocation
4. Access Control Via Views
REVOKE {
  <privileges> | ALL [PRIVILEGES]
}
ON <relation>
FROM {
  <users> | PUBLIC
};
REVOKE { <privileges> | ALL [PRIVILEGES] } ON <relation> FROM { <users> | PUBLIC };
REVOKE { <privileges> | ALL [PRIVILEGES] } ON <relation> FROM { <users> | PUBLIC };

- <privileges>
  - select, update, insert, delete
  - ALL or ALL PRIVILEGES are shorthands for all the privileges
- <relation>
  - name of the relation
REVOKE { <privileges> | ALL [PRIVILEGES] } 
ON <relation>
FROM { <users> | PUBLIC }; 

- <privileges>
  - select, update, insert, delete
  - ALL or ALL PRIVILEGES are shorthands for all the privileges

- <relation>
  - name of the relation

- <users>
  - users
  - PUBLIC is a shorthand for all users
Examples

REVOKE update, insert ON Movies FROM Hellen;
REVOKE update ON Customers FROM Marc;
REVOKE select ON Videos FROM Gena;
Examples: Multiple Sources of Privileges

Leo:    GRANT select ON Videos TO Beth, Gena
        WITH GRANT OPTION;
Helen:  GRANT insert, select ON Videos TO Beth;
Leo:    REVOKE select ON Videos From Beth, Gena;
Examples: Multiple Sources of Privileges

Leo:  \texttt{GRANT select ON Videos TO Beth, Gena WITH GRANT OPTION;}

Helen: \texttt{GRANT insert, select ON Videos TO Beth;}

Leo: \texttt{REVOKE select ON Videos From Beth, Gena;}

- Gena loses select privilege on Videos
Examples: Multiple Sources of Privileges

Leo:  GRANT select ON Videos TO Beth, Gena
      WITH GRANT OPTION;

Helen: GRANT insert, select ON Videos TO Beth;

Leo:  REVOKE select ON Videos From Beth, Gena;

- Gena loses select privilege on Videos
- Beth loses ability to grant select on Videos (granted by Leo)
Examples: Multiple Sources of Privileges

Leo:  GRANT select ON Videos TO Beth, Gena
      WITH GRANT OPTION;
Helen: GRANT insert, select ON Videos TO Beth;
Leo:    REVOKE select ON Videos From Beth, Gena;

- Gena loses select privilege on Videos
- Beth loses ability to grant select on Videos (granted by Leo)
- Beth can still perform select on Videos (granted by Helen)
The System Catalog **Sysauth**

- **user**: user to which privileges are granted
The System Catalog $\text{Sysauth}$

- **user**: user to which privileges are granted
- **rel**: name of relation
The System Catalog $Sysauth$

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type $\in \{R, V\}$**: base relation or view
The System Catalog **Sysauth**

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type**: $\in \{R, V\}$: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted

Two requirements:
1. Timestamp increases monotonically
2. No two *GRANT* statements have the same timestamp

Can be simply a counter

- **grantor**: who granted the privileges
- **grantopt**: $\in \{Y, N\}$: grantable or not
The System Catalog `Sysauth`

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type ∈ \{R, V\}**: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted
  - **Two requirements:**
    - Timestamp increases monotonically
    - No two `GRANT` statements have the same timestamp
- **grantor**: who granted the privileges
- **grantopt ∈ \{Y, N\}**: grantable or not
The System Catalog \textit{Sysauth}

- \textbf{user}: user to which privileges are granted
- \textbf{rel}: name of relation
- \textbf{type} \in \{R, V\}: base relation or view
- \textbf{select}: 0 (no privilege) or the timestamp at which the select privilege was granted

Two requirements:

- Timestamp increases monotonically
The System Catalog **Sysauth**

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type ∈ \{R, V\}**: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted

Two requirements:

1. Timestamp increases monotonically
2. No two `GRANT` s have the same timestamp
user: user to which privileges are granted
rel: name of relation
type ∈ \{R, V\}: base relation or view
select: 0 (no privilege) or the timestamp at which the select privilege was granted

Two requirements:
1. Timestamp increases monotonically
2. No two GRANTs have the same timestamp

Can be simply a counter
The System Catalog **Sysauth**

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type ∈ \{R, V\}**: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted
  - **Two requirements**:
    1. Timestamp increases monotonically
    2. No two `GRANT` s have the same timestamp
  - Can be simply a counter
- **update, insert, delete**: similar
The System Catalog $\text{Sysauth}$

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type $\in \{R, V\}$**: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted
  - **Two requirements**:
    1. Timestamp increases monotonically
    2. No two `GRANT` s have the same timestamp
  - Can be simply a counter
- **update, insert, delete**: similar
- **grantor**: who granted the privileges
The System Catalog **Sysauth**

- **user**: user to which privileges are granted
- **rel**: name of relation
- **type**: base relation or view
- **select**: 0 (no privilege) or the timestamp at which the select privilege was granted
  - Two requirements:
    1. Timestamp increases monotonically
    2. No two **GRANT**s have the same timestamp
  - Can be simply a counter
- **update, insert, delete**: similar
- **grantor**: who granted the privileges
- **grantopt**: grantable or not
## Example of Sysauth

<table>
<thead>
<tr>
<th>user</th>
<th>rel</th>
<th>type</th>
<th>select</th>
<th>insert</th>
<th>update</th>
<th>grantor</th>
<th>grantopt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leo</td>
<td>Customers</td>
<td>R</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Leo</td>
<td>Videos</td>
<td>R</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Leo</td>
<td>Movies</td>
<td>R</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Marc</td>
<td>Customers</td>
<td>R</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>Leo</td>
<td>N</td>
</tr>
<tr>
<td>Beth</td>
<td>Videos</td>
<td>R</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>Leo</td>
<td>Y</td>
</tr>
<tr>
<td>Gena</td>
<td>Videos</td>
<td>R</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>Leo</td>
<td>Y</td>
</tr>
<tr>
<td>Matt</td>
<td>Videos</td>
<td>R</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>Gena</td>
<td>N</td>
</tr>
<tr>
<td>Helen</td>
<td>Videos</td>
<td>R</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>Leo</td>
<td>Y</td>
</tr>
<tr>
<td>Helen</td>
<td>Movies</td>
<td>R</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>Leo</td>
<td>Y</td>
</tr>
<tr>
<td>Beth</td>
<td>Videos</td>
<td>R</td>
<td>47</td>
<td>47</td>
<td>0</td>
<td>Helen</td>
<td>N</td>
</tr>
</tbody>
</table>
The System Catalog *Syscolauth*

- Specifies which column of a relation can be updated by a user.
- \((\text{user}, \text{rel}, \text{attr}, \text{grantor}, \text{grantopt})\)
Analogous to the access control matrix, the two system catalogs `Sysauth` and `Syscolauth` together constitute the *protection state* of the system.

Authorization proceeds by consulting these two catalogs.
Outline

1. Grant
2. Revocation
3. Recursive Revocation
4. Access Control Via Views
users $u_1$, $u_2$, $u_3$
Recursive Revocation: Motivating Example

- users $u_1$, $u_2$, $u_3$
- $u_1$ grants $u_2$ privilege $p$ on relation $rel$ with grant option

What happens to the authorization granted to $u_3$?
Recursive Revocation: Motivating Example

- users $u_1$, $u_2$, $u_3$
- $u_1$ grants $u_2$ privilege $p$ on relation $rel$ with grant option
- $u_2$ grants $u_3$ privilege $p$ on relation $rel$
Recursive Revocation: Motivating Example

- users $u_1$, $u_2$, $u_3$
- $u_1$ grants $u_2$ privilege $p$ on relation $rel$ with grant option
- $u_2$ grants $u_3$ privilege $p$ on relation $rel$
- $u_1$ revokes $u_2$ $p$ on $rel$
Recursive Revocation: Motivating Example

- users $u_1$, $u_2$, $u_3$
- $u_1$ grants $u_2$ privilege $p$ on relation $rel$ with grant option
- $u_2$ grants $u_3$ privilege $p$ on relation $rel$
- $u_1$ revokes $u_2$ $p$ on $rel$
- What happens to the authorization $u_2$ granted to $u_3$?
Recursive Revocation: Definition

Definition

Suppose $G_1, \ldots, G_n$ is a sequence of Grant commands, all granting privilege $p$ on relation $rel$, executed in the order specified by the sequence.

Consider the following sequence of commands:

$$G_1, \ldots, G_i, \ldots, G_n, R_i$$

where $R_i$ is the revocation of the privilege granted by $G_i$.

The authorization state (aka protection state) resulted from executing the sequence above should be equivalent to the authorization state resulted from executing the following sequence:

$$G_1, \ldots, G_{i-1}, G_{i+1}, \ldots, G_n$$
Authorization Graph

- To help track the protection state with respect to a specific privilege \( p \) on relation \( rel \).
- A node for each user holding \( p \) on \( rel \)
- An edge from \( u_1 \) to \( u_2 \) labelled with timestamp \( t \) if at \( t \) \( u_1 \) grants the privilege to \( u_2 \).
- The edge is further labelled with \( g \) if the privilege is granted with grant option.
Example

Leo: GRANT select ON Videos TO Beth, Gena
    WITH GRANT OPTION;

Gena: GRANT select ON Videos TO Matt;

Leo: GRANT ALL PRIVILEGES ON Movies, Videos TO Helen
    WITH GRANT OPTION;

Helen: GRANT insert, select ON Videos TO Beth;
Authorization Graph for Example

(a)
Example

Leo:  GRANT select ON Videos TO Beth, Gena
      WITH GRANT OPTION;

Gena: GRANT select ON Videos TO Matt;

Leo:  GRANT ALL PRIVILEGES ON Movies, Videos TO Helen
      WITH GRANT OPTION;

Helen: GRANT insert, select ON Videos TO Beth;

Leo:   REVOKE select ON Videos FROM Gina;

- Matt will lose his privilege as well
Another Example

Suppose Leo revokes select on Videos from Gena at time 55. This does not cause revocation of privilege granted by Gena to Matt.

After time 32, at time 40, Beth grants privilege to Gena. However, if the time at which Beth grants privilege to Gena was 60, then Matt's privilege would have been revoked.
Suppose Leo revokes select on Videos from Gena at time 55
Another Example

Suppose Leo revokes select on Videos from Gena at time 55
Doesn’t cause revocation of privilege granted by Gena to Matt
Another Example

Suppose Leo revokes select on Videos from Gena at time 55
Doesn’t cause revocation of privilege granted by Gena to Matt
- After time 32, at time 40, Beth grants privilege to Gena
Another Example

Suppose Leo revokes select on Videos from Gena at time 55
Doesn’t cause revocation of privilege granted by Gena to Matt

- After time 32, at time 40, Beth grants privilege to Gena
- However, if the time at which Beth grants privilege to Gena was 60, then Matt’s privilege would have been revoked.
Advantages/Disadvantages

- Advantage: clean semantics
- Disadvantage: sometimes a revoke is performed by a user in a certain organization role, even if the user is no longer in the role, the effect of the revocation should be preserved
Revocation in SQL

- REVOKE can be issued with one of 2 options:
  1. RESTRICT
  2. CASCADE
Revocation in SQL

- **REVOKE** can be issued with one of 2 options:
  1. **RESTRICT**
     - not executed if recursive revocation is involved
  2. **CASCADE**
Revocation in SQL

- **REVOKE** can be issued with one of 2 options:
  1. **RESTRICT**
     - not executed if recursive revocation is involved
  2. **CASCADE**
     - recursive revocation without considering timestamps
Revocation in SQL

REVOKE can be issued with one of 2 options:

1. **RESTRICT**
   - not executed if recursive revocation is involved

2. **CASCADE**
   - recursive revocation without considering timestamps
   - so long as a grant \((t_1)\) is supported by another grant \((t_2)\), even if \(t_1 < t_2\), it is not recursively revoked.
Outline

1. Grant
2. Revocation
3. Recursive Revocation
4. Access Control Via Views
Access Control Via Views

- Fine-grained access control
  - grant only a subset of tuples in a table
  - grant only some columns of a table
- Content-dependent access control
  - grant only a subset of tuples with specific contents
Example

CREATE VIEW Comedies AS
SELECT *
FROM Movies
WHERE type = 'comedy'

- Now the creator of Comedies can grant a selected set of tuples to another user
  - View creation is successful only if the creator has privilege to perform the select.
  - Granting of view access to others is allowed only if the creator has the granted privilege with grant option.
CREATE VIEW Top_of_the_Class AS
SELECT * FROM Students
WHERE Grade <
  (SELECT Grade
     FROM Students
     WHERE Name = current_user());
More Examples

CREATE VIEW My_Journeys AS
SELECT * FROM Diary
WHERE Customer = current_user();
Confidentiality

- Secure views can implement a form of confidentiality policy (e.g., top secret, secret, confidential, unclassified)

```sql
CREATE VIEW Flights_at_CONFIDENTIAL AS
    SELECT * FROM Diary
    WHERE Destination = 'THU' AND Status = 'business'
```

- One view for each security level.
Views can implement controlled invocation (Gollmann):
- temporarily amplification of privilege level
- analogous to setuid in Unix

CREATE VIEW My_Journeys AS
SELECT * FROM Diary
WHERE Customer = current_user();
Most of the materials in these slides are based on:

- [Ferrari], Chapter 2
- [Gollmann], Chapter 9