Acten (Action Entity) Model

- Proposed by Bussolati et al 1983
- As an extension to the TG model
  - Further administrative privileges
  - Predicates on authorization
- Two separate graphs
  - Access Authorizations
  - Administrative privileges
- Aim of the extension:
  - Correcting the non-selectivity of admin privileges
  - Enlarging the controls on the priv. transport flow

Acten -2

- Every security-relevant element of the system is viewed as entity.
- Entity regards both subjects and objects.
- Entities are resource types.
- Access modes
  - Static
  - Dynamic
  - Both are called actions.
Static Actions

• Their execution does not modify the auth. State of the system
• Consists of
  – Use: between users, operators and I/O resources, applications, …
  – Read: to access the contents of an entity
  – Write
  – Create: of entities
  – Delete: of an entity

Dynamic Actions

• Their execution modify the auth. State of the system
• Are:
  – Grant: held by E_i towards E_j related to an static mode m and referred to E_k allows E_i to grant E_j ,m on E_k.
  – Revoke:
  – Delegate: Allows E_i to grant E_j ,the dynamic authorization ‘grant’ related to m on E_k. Authorization to delegate/abrogate are limited to a few entities.
  – Abrogate
Acten …

- If $E_i$ hold the highest level access mode (create/remove), and the delegate/abrogate on $E_j$, then $E_i$ is the owner of $E_j$.
- The owner of an entity is unique.
- The delegate/abrogate access modes are given exclusively to the owner.
- Hierarchy of access modes
  - Create/Delete 4
  - Update 3 Delegate/abrogate 2
  - Read 2 Grant/revoke 1
  - Use 1

When $E_i$ is authorized to access $E_j$ in mode at level $L$, it is also authorized to access $E_j$ according to all access modes at levels $L' \leq L$.

- When an access mode is revoked, all the modes at a higher classification are revoked
**Authorizations**

- Authorization is a binary relation between entities: characterized by the involved entities and access mode, by a direction and one or more attributes (predicates).
- A static authorization is indicated as:
  \[ A_{ij} = a \sim \{P_{ij}\} \], where \( a \) is the access mode of \( E_i \) on \( E_j \) if the condition in \( P_{ij} \) is satisfied
- The predicate considers only the system conditions (time or origin)
- Value dependent predicates should be applied to the entity itself to create a more restricted entity (similar to VIEW)

**Authorizations**

- Dynamic access modes are characterized by exp of the form
  \[ A_{ij/k} = a \sim \{P_{ij}\} \]
- For each entity there are two sets: the auth set and the protection set as:
  - \( S_a(E_i) \)
  - \( S_p(E_i) \)
  - \( S_a(E_i) \) is all the authorization that \( E_i \) holds on the other entities
  - \( S_p(E_i) \) is all authorization held by the system entities onto \( E_i \)
Entity classification

- Based on $S_a(E_i)$ and $S_p(E_i)$, entities are classified as:
  - Active entity: if $S_a(E_i) \neq 0$; $E_i$ can execute actions on the other entities
  - Passive entity: if $S_p(E_i) \neq 0$;
  - Active/passive entity: if $S_a(E_i) \neq 0$; and $S_p(E_i) \neq 0$; e.g. processes, transactions, …

Acton model structures

- The model elements are represented in two graphs:
  - SG: auth for static actions. Nodes are entities, arcs are the auth. If an auth is constrained, the arc is crossed.
Acten ....

– DG: auth for dynamic actions: nodes: entities; arcs are the auths. Arcs are labeled of the form “DA SA {M}”, M is only used for deleg/ab

Data structures in Acten

• Table of security classes:
  – for each entity Ei,
    • The static action Ai+ at the highest level that Ei can execute on the system entities:: Active potentiality of Ei.
    • The static action Ai- is the highest level that can be executed on Ei by the system entities:: Passive potentiality of Ei.
  – No Active potentiality is defined for the passive entities and no passive potentiality is defined for the active entities.
  – The table allows the admin to assign an auth. level and a prot. level to each entity.
## Security class table

<table>
<thead>
<tr>
<th>E1: user</th>
<th>Potentiality</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1+</td>
<td>Update</td>
</tr>
<tr>
<td></td>
<td>A1-</td>
<td>-</td>
</tr>
<tr>
<td>E2: application</td>
<td>A2+</td>
<td>Update</td>
</tr>
<tr>
<td></td>
<td>A2-</td>
<td>Create/delete</td>
</tr>
<tr>
<td>E3: Peripheral</td>
<td>A3+</td>
<td>Create/delete</td>
</tr>
<tr>
<td></td>
<td>A3-</td>
<td>Use</td>
</tr>
<tr>
<td>E4: Data Record</td>
<td>A4+</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>A4-</td>
<td>Create/delete</td>
</tr>
<tr>
<td>E5: Application</td>
<td>A5+</td>
<td>Create/delete</td>
</tr>
<tr>
<td></td>
<td>A5-</td>
<td>Create/delete</td>
</tr>
</tbody>
</table>

## Acten ....

- Table of entity states: shows the entities that own the other entities and the list of owned entities.

<table>
<thead>
<tr>
<th>Owner entity</th>
<th>Owned entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>E8, E5, E7</td>
</tr>
<tr>
<td>E4</td>
<td>E6</td>
</tr>
<tr>
<td>E7</td>
<td>E3, E2</td>
</tr>
</tbody>
</table>

- Table of action hierarchy: The list of static and dynamic action hierarchies.
Consistency and transformation rules

• Rules during the system life cycle:
  – Internal consistency rules: the structure of SG and DG should be consistent regarding to the nature of the represented protection system.
  – Mutual consistency rules: SG and DG should be consistent in representing the privileges. E.g. a privilege should not be propagated if the grantor does not hold that privilege.
  – Transformation rules: control the indirect authorizations and the access right propagation.

Internal consistency for SG

• A system entity can execute or undergo only those access modes which are compatible with its type and role.
  – A peripheral system cannot be deleted!
  – Use is inapplicable for a file!
Internal consistency for DG

- 4 basic rules:
  - Let $E_j$ be the owner of $E_k$, no other entities $E_i$, $\forall i, k \neq j, k \neq i$; can hold the authorization $A_{ijk}$ of type “GRANT SA”; SA is any static action. This means that owner is the only entity who has the highest right on $E_k$.
  - An entity $E_i$ can be authorized for delegate/abrogate actions on $E_k$, if $E_i$ is the owner of $E_k$.
  - Let $E_j$ is authorized to grant to $E_m$ a static action of level $L'$ with parameter $E_k$. $E_i$, (owner of $E_i$) can run $A_{ij/k}$ of type “Delegate SA” only if $L(SA) > L'$.
  - Recipient of a SA of level $L$ through a delegate action ($E_i$) can grant this SA to $E_m$ whose active potentiality ($A^+$) has level $L' > L$.

Transformation rules for SG

- Purpose: show all privileges held indirectly by an entity on the other entities.
- Defined for groups of 3 entities $E_i$, $E_j$, $E_z$ and 2 authorizations $A_{ij}$, $A_{jz}$.
- Each rule determines the static action $A_{iz}$ that can be executed via the $A_{ij}$-$A_{jz}$.
- A transf. rule is $A_{iz} = F(A_{ij}, A_{jz}, E_i, E_z)$
- The level of $A_{iz}$ cannot be higher than the level of active potentiality of $E_i$, nor less than the level of passive potentiality of $E_j$. 

Database Security, Jalili, 2nd Semester, CE, SUT
Algorithm …

- If \( L(A_{ij}) > L(A_{jz}) \) then \( L(A_{iz}) = L(A_{jz}) \)
- If \( L(A_{ij}) \leq L(A_{jz}) \) then check the active potentiality of \( E_i \)
  - if \( L(A_{jz}) > L(A_{i+}) \) then \( L(A_{iz}) = L(A_{i+}) \)
  - if \( L(A_{jz}) < L(A_{i+}) \) then \( L(A_{iz}) = L(A_{jz}) \)
- If \( A_{ij}, A_{jz} \) are constrained by some predicates \( P_{ij}, P_{jz} \), then \( P_{iz} = P_{ij} \cap P_{jz} \)
- Next slide as an example
Transformation rules for DG

- Control the flow of privileges in the system: the flow of static and dynamic actions due to the execution of dynamic actions.
- Indirect authorization
- Each rule is defined for a group of 3 entities Ei, Ej, Ek and two actions Aij/k, Ajz/k defined on the same parameter entity Ek.

......

**Figure 2.14** Chain of subordinated actions.
Mutual consistency rules for SG & DG

Ensuring that dynamic privileges are consistent with the state of auth. for access modes and vs.

1. Ei can grant Ej the privilege to execute a static action SA of level L upon Ek only if in the SG, Ei can execute SA’ of level L’ ≥ L on Ek.

2. If Ei can execute a static action Ajk of level L, then in DG Ei can grant to Ej the privilege to execute only static auth of level Li > L on Ek.

3. For each Aij/k of type “Delegate SA”, the corresponding auth Aij must exist in the SG of type create/delete.

Woods et al model
Woods …

- 1979 by Woods
- Orientation is on auth management and access control in multilevel schema DB.
- This aim is distinguished from “Multilevel security” models and arch.
- The model considers the 3-level arch of ANSI/SPARC including
  - External level: nearest to the user
  - Conceptual level: representation of the data
  - Internal level: closest to the physical storing, not bound to the hardware elements.

*Figure 2.15 Three-level database architecture in the ANSI/SPARC proposal.*
Each schema may be based on a different data model.

This model focuses on the consistency among the auth. rules in different levels.

The model considers a relational DB model for the external level and ER model for the conceptual level.

Subjects: users of the system
- Authorizer, who administer the authorizations
- Users, who access data

Objects:
- Conceptual-level objects: each O belongs to a type category, specified by a function n, specifies its type category (n(O)).
  - Type categories at conceptual level are: entity set, relationship type, and attribute.
  - An attribute may be associated with one entity set and maps the entity set to a value set.
  - A relationship type is a bidirectional association between two entity set, and may have two names.
- External-level objects: Type categories are table, view, and field (column). ‘x.y’ indicates field x of table y.
Access modes in Woods ... model

- At each level (conc. & ext.), a set TP of operations is defined
- At the conceptual level:
  - TP_entity = \{insert, delete\}
  - TP_attribute = \{insert, read, use\}
  - TP_relationship = \{insert, read, use\}
- Type category C = \{entity, attribute, relation\}
- TC = TP_entity \cup TP_attribute \cup TP_relationship = \{insert, delete, read, use\}

- At the external level:
  - TP_table = \{insert, delete\}
  - TP_field = \{update, read, use\}
- Type category E = \{table, field\}
- TC = TP_table \cup TP_field = \{insert, delete, update, use\}

- Mapping between operation at the levels.
- Mapping function φ_v : TE \times E \rightarrow TC \times C
  - Maps the external level operations into the conceptual level operations
- At the element level, it is:
  - φ_v: \langle t, e \rangle \rightarrow \langle t_{k}^{i}, C_{j}^{k} \rangle \ \forall \ k, t_{s}^{k} \in T_{d}^{k}
Woods …

- Auth. state are stated by access rules.
- Access rules are of the form \(<s, o, t, p>\), where subject \(s\) exercises the access mode \(t\) on object \(o\), under the condition \(p\).

- Access rules are defined by an authorizer.
- Basic rules are specified at the conc. level, which is a global view of data of an org.

- A table definition (at the external level) includes access constraints specifying which access types are legal for each external object \(e\) (each table and its fields)

- These constraints are stored in an Access Constraint table (AC). e.g. next slide

AC in Woods …

<table>
<thead>
<tr>
<th>(E)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>delete</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>insert</td>
</tr>
<tr>
<td>EMPLOYEE.name</td>
<td>read</td>
</tr>
<tr>
<td>EMPLOYEE.ssn</td>
<td>read</td>
</tr>
<tr>
<td>EMPLOYEE.manager</td>
<td>read</td>
</tr>
<tr>
<td>EMPLOYEE.salary</td>
<td>read</td>
</tr>
<tr>
<td>EMPLOYEE.manager</td>
<td>update</td>
</tr>
<tr>
<td>EMPLOYEE.salary</td>
<td>update</td>
</tr>
</tbody>
</table>
Access rules specified on conceptual objects are transformed into access rights on external objects.

Access to external objects can be specified in two ways:

- By accepting the access rules derived from the conc. level rules.
- Defining some new, more restrictive rules.

Fig. in next slide
Access rules at the conc. level

- In the form of <s,o,t,p>
- Stored in the conceptual rule (CR) table.
- Next slide

- The specified access type in these rules must belong to the set of legal access types for the corresponding category

Conceptual Rule Table

<table>
<thead>
<tr>
<th>Subject</th>
<th>Conceptual object</th>
<th>Access Type</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Javad</td>
<td>EMPLOYEE.name</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>Javad</td>
<td>EMPLOYEE.ssn</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>Javad</td>
<td>EMPLOYEE.manager</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>Javad</td>
<td>EMPLOYEE.salary</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>Javad</td>
<td>EMPLOYEE.salary</td>
<td>update</td>
<td>WHERE EMPLOYEE.manager = javad</td>
</tr>
<tr>
<td>Saeed</td>
<td>EMPLOYEE.name</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>Saeed</td>
<td>EMPLOYEE.ssn</td>
<td>update</td>
<td></td>
</tr>
</tbody>
</table>
Access rules at the external level

- External level access rules.
- Stored in an External Rules (ER) table

- Rules are either explicitly defined or derived from the underlying conceptual level access rules.

- The predicates in rules are a conjunction of a system predicate and a data predicate.

- When an ER is defined, it must be checked for consistency with both the access constraints (AC) and the underlying conceptual access rules

Let \(<s_i, e_i, t_i, p_i>\) be an ER, where \(e_i\) is a table or a field, the following two consistency conditions should be satisfied:

1- Consistency of the access rule with the access constraints specified for table \(t_i\).

2- Consistency of the access rule with the underlying conceptual level rules.
Access Control in Woods … model

- An access request is a 4-tuple \(<s', o', t', p'>\), \(p'\) specifies the specific occurrence of \(o'\) requested by \(s'\).

- The model considers a close policy.

- On the request, the system controls the existence of an ER, \(<s, o, t, p>\), where \(s=s'\), \(o=o'\), \(t=t'\). If exists, then access is granted for the occurrences of \(o'\) satisfying \(p'\) and \(p\).

- Then the intersection of \(p'\) and \(p\) is computed and is substitute to \(p'\) in the user query.

- The query on the external object \(e'\) is transformed into the set of \(<\text{operations, object-subset}>\), where an object-subset is a set of occurrences of conceptual-level objects corresponding \(e'\).

- Each \(<\text{operations, object-subset}>\) is then modified to be restricted by the predicates specified in the conceptual-level rules.
مثالی با دستورات SQL

شما می‌توانید دو رابطه نام بدهید: DEPARTMENT و EMPLOYEE

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>SSN</td>
<td>BDATE</td>
<td>ADDRESS</td>
<td>SEX</td>
<td>SALARY</td>
<td>DNO</td>
</tr>
</tbody>
</table>

| DEPARTMENT | | | |
|------------|-------|--------|
| DNUMBER    | DNAME | MGRSSN |

فرض کنید که کاربران DBA, A1, A2, A3, A4 و A5 قادر به ایجاد دو رابطه DEPARTMENT و EMPLOYEE باشند:

grant createtab to A1;

SQL2 در:

create schema example
authorization A1;
مثالی با دستورات SQL (ادامه)

فرض کنید که A1 بخواهد به کاربر A2 مجوزهای درج و حذف سطرها را برای هر دو رابطه اعطا کند. ولی A1 نمی‌خواهد که A2 بتواند این مجوزها را به کاربران دیگر بدهد:

\[
\text{GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;}
\]

مثالی با دستورات SQL (ادامه)

- فرض کنید که A1 بخواهد به کاربر A3 مجوز استخراج اطلاعات را برای هر دو رابطه اعطا کند و همچنین به وی اجازه دادن مجوز SELECT را به کاربران دیگر بدهد:

\[
\text{GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTIONS;}
\]

- اگر A3 می‌تواند مجوز برای رابطه SELECT اعطا کند، به کاربر A4 مجوز EMPLOYEE:

\[
\text{GRANT SELECT ON EMPLOYEE TO A4;}
\]
SQL

مثالی با دستورات

فرض کنید که A1 تصمیم بگیرد مجوز برای رابطه SELECT برای کاربر A3 داده بود، لغو کند:

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

SQL

مثالی با دستورات

فرض کنید که A1 دوباره بخواهد به A3 مجوز رابطه EMPLOYEE را همراه با اجازه انتشار این مجوز، اعطا کند (ولی به صورت محدود شده).

این محدودیت عبارت است از استخراج خصیص‌های NAME، BDATE و ADDRESS فقط برای سطرهای با DNO = 5:

```
CREATE VIEW A3EMPLOYEE AS
SELECT NAME, BDATE, ADDRESS
FROM EMPLOYEE
WHERE DNO = 5;
```
مثالی با دستورات SQL (ادامه)

اکنون A1 می‌تواند به کاربر A3 مجوز برای دید را همراه با اجازه انتشار این مجوز، اعطای کند:

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;
```

فرض کنید که A1 به‌خواه اجازه دهد که فقط بتواند خصوصیه SALARY از رابطه EMPLOYEE را تغییر دهد:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;
```