PROTEZIONE DEI DATI

Gli obiettivi della protezione dei dati sono:

- *Segretezza*: protezione delle informazioni da letture non autorizzate

- *Integrità*: protezione dei dati da modifiche o cancellazioni non autorizzate

- *Disponibilità*: garanzia che non si verifichino casi in cui ad utenti legittimi venga negato l'accesso ai dati

importanza assegnata a tali obiettivi varia a seconda del sistema considerato

FUNZIONALITÀ RICHIESTE

- *Autenticazione*: meccanismi per verificare l'identità dell'utente che si connette al sistema

- *Controllo dell'accesso*: meccanismi per ogni richiesta di accesso ai dati verifichino che l'utente che la effettua sia autorizzato a compiere l'accesso

- *Crittografia dei dati*: meccanismi che consentano di crittografare i dati trasmessi sulla rete di comunicazione in modo che possano essere decifrati solo da utenti autorizzati

noi vedremo il controllo dell'accesso (responsabilità del DBMS), altre funzionalità fornite da sistema operativo, hardware, reti ...

CONTROLLO DELL'ACCESSO: CONCETTI FONDAMENTALI

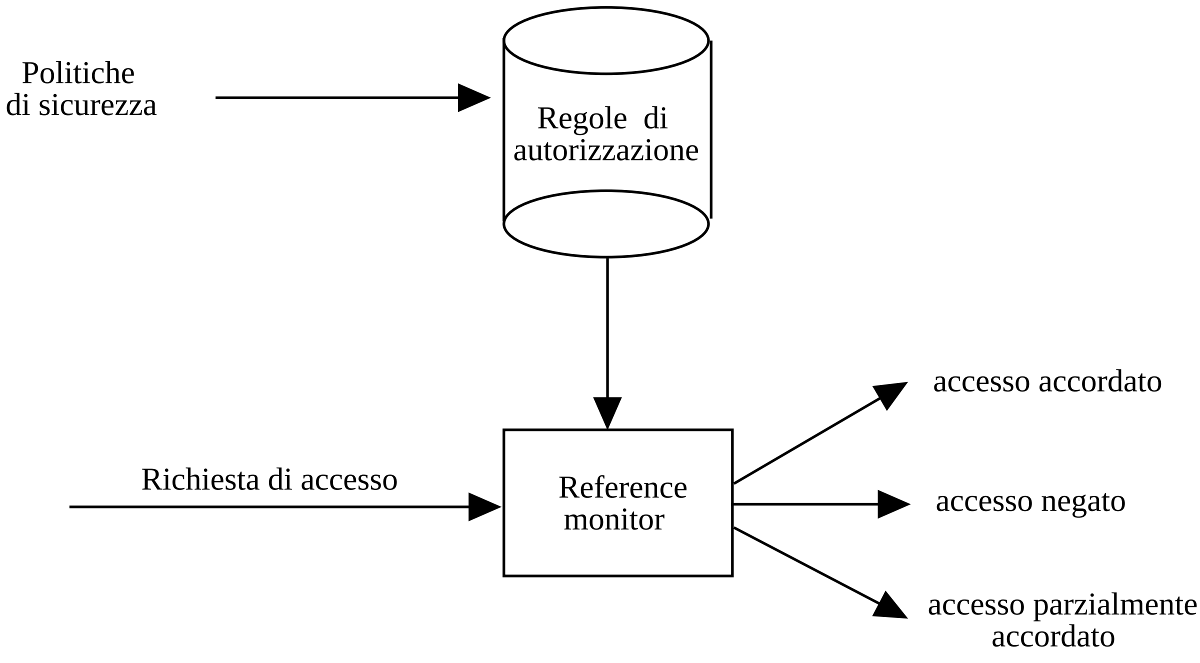
- regola le operazioni che si possono compiere sulle informazioni e le risorse in una base di dati

  lo scopo è limitare e controllare le operazioni che gli utenti effettuano, prevenendo accidentali o deliberate azioni che potrebbero compromettere la correttezza e la sicurezza dei dati

- le risorse sono costituite dai dati, memorizzati in *oggetti* a cui si vuole garantire protezione

  i *soggetti* sono agenti (utenti o programmi in esecuzione) che richiedono di poter esercitare privilegi (come lettura, scrittura o esecuzione) sui dati.

CONTROLLO DELL'ACCESSO: CONCETTI FONDAMENTALI



CONTROLLO DELL'ACCESSO: CONCETTI FONDAMENTALI

- *politiche di sicurezza*: leggi e principi secondo cui l'organizzazione vuole che siano gestite e protette le informazioni

  rappresentano un insieme di direttive ad alto livello che esprimono le scelte di fondo dell'organizzazione relative alla sicurezza dei propri dati

- sono implementate mediante traduzione in un insieme di *regole di autorizzazione* che stabiliscono le operazioni e i diritti che gli utenti possono esercitare sui vari oggetti del sistema

- il *reference monitor* è un meccanismo di controllo che ha il compito di stabilire se l'utente può essere autorizzato (totalmente o parzialmente) a compiere l'accesso.

POLITICHE DI SICUREZZA

- la politica di sicurezza adottata dipende principalmente da fattori organizzativi, quali l'ambiente di installazione, le esigenze degli utenti, i regolamenti dell'organizzazione, o i vincoli di natura legale

- due classi fondamentali:

  1. *politiche per l'amministrazione della sicurezza*

  2. *politiche per il controllo dell'accesso ai dati*

POLITICHE PER L'AMMINISTRAZIONE DELLA SICUREZZA

chi concede e revoca i diritti di accesso

Centralizzata un unico autorizzatore (o gruppo), detto DBA, controlla l'intera base di dati

Decentralizzata autorizzatori (o gruppi) diversi controllano porzioni differenti della base di dati

Ownership l'utente che crea un oggetto gestisce le autorizzazioni sull'oggetto

Gerarchica il DBA può delegare ad altri utenti la possibilità di concedere o revocare diritti su specifici oggetti

Cooperativa le autorizzazioni non possono essere concesse da un singolo utente, ma richiedono il consenso di più utenti.

POLITICHE PER IL CONTROLLO DELL'ACCESSO AI DATI

definiscono i metodi utilizzati per decidere se concedere o meno l'accesso ai dati

<u>LIMITAZIONE DEGLI ACCESSI</u>

decidere la quantità di informazione a cui ciascun utente può accedere

Need to know (politica del minimo privilegio) - molto restrittiva, permette ad ogni utente l'accesso solo ai dati strettamente necessari per eseguire le proprie attività

Maximized sharing (politica della massima condivisione) - consente agli utenti il massimo accesso alle informazioni nella base di dati, mantentendo comunque delle informazioni riservate.

LIMITAZIONE DEGLI ACCESSI

- polica need to know

  + offre ottime garanzie di sicurezza ed è adatta a basi di dati con forti esigenze di protezione

  - può portare ad un sistema eccessivamente protetto, negando anche accessi che non comprometterebbero la sicurezza del sistema

- politica maximized sharing

  + soddisfa il massimo numero possibile di richieste di accesso

  viene utilizzata in ambienti in cui esiste una certa fiducia tra gli utenti e in cui non è sentita una forte esigenza di protezione.

SISTEMI APERTI O CHIUSI

- in un sistema chiuso l'accesso è permesso *solo se* esplicitamente autorizzato

  in un sistema aperto l'accesso è permesso *a meno che* non sia esplicitamente negato

- in un sistema chiuso le regole di autorizzazione indicano per ogni soggetto i diritti che egli può esercitare sugli oggetti del sistema, questi diritti sono i soli che verranno accordati dal meccanismo di controllo

  in un sistema aperto le regole di autorizzazione indicano per ogni soggetto i diritti che egli non può esercitare sugli oggetti del sistema: questi diritti sono i soli che gli saranno negati.

SISTEMI APERTI O CHIUSI

- un sistema chiuso implementa la politica del minimo privilegio, un sistema aperto implementa la politica della massima condivisione

- un sistema chiuso offre maggiori garanzie di sicurezza: una regola inavvertitamente cancellata o non inserita restringe ulteriormente l'accesso, mentre in un sistema aperto permette accessi non autorizzati

- la maggior parte delle basi di dati oggi esistenti sono sistemi chiusi

  la scelta dipende comunque dalle caratteristiche e dalle esigenze di protezione dell'ambiente nel quale la base di dati è utilizzata.

GRANULARITÀ

- granularità a cui il controllo dell'accesso deve essere effettuato

- requisito minimo: possibità di specificare nelle regole di autorizzazione gli oggetti a cui l'utente può accedere

  nel modello relazionale una relazione o uno o più attributi di una relazione

- tali tipi di controllo vengono chiamati **controlli dipendenti dal nome** poiché fanno esplicito riferimento al nome dell'oggetto

  in tal caso una regola di autorizzazione ha la forma (`s`,`o`,`m`), dove `s` è l'utente a cui si vuole concedere il modo di accesso `m`, ed `o` è il nome dell'oggetto su cui il modo di accesso è concesso.

GRANULARITÀ

- una forma più sosfisticata è il **controllo in base al contenuto**, che permette di condizionare l'accesso ad un oggetto al valore di una o più delle sue componenti

  esempio: si può autorizzare un impiegato ad accedere solo alle informazioni sugli impiegati il cui stipendio non supera una certa soglia

- **controlli dipendenti dal contesto** permettono di specificare insiemi di attributi che non devono mai essere acceduti contemporaneamente o che, viceversa, devono sempre essere acceduti insieme

  esempio: non si può richiedere contemporaneamente l'accesso al nome e allo stipendio di un impiegato, o le informazioni riguardanti una persona in via di arresto possono essere accedute solo se l'esito dell'arresto è incluso nelle informazioni.

GRANULARITÀ

- **controlli dipendenti dalla storia degli accessi** permettono di condizionare l'esito di una richiesta di accesso alla storia degli accessi eseguiti precedentemente

  esempio: specificare che un utente può accedere ad un determinato dato solo se il numero di accessi da lui compiuti su quel dato in un determinato intervallo di tempo non supera una certa soglia

- la scelta della granularità è guidata da un' analisi costi-benefici che tale scelta comporta: esprimere sofisticati requisiti di protezione vs controllo dell'accesso meno efficiente.

CONTROLLO DELL'ACCESSO

- le politiche di sicurezza devono stabilire quando e come i soggetti possono accedere ai dati contenuti nel sistema, e se e come possono venire trasmessi i diritti di accesso

- le politiche per il controllo degli accessi si suddividono in
  - *politiche discrezionali*
  - *politiche mandatorie*

  tali politiche differiscono nel modo in cui sono definite e gestite le regole di autorizzazione.

Politiche discrezionali

- richiedono che vengano stabiliti, mediante apposite regole di autorizzazione, i diritti che ogni soggetto possiede sugli oggetti del sistema

- il meccanismo di controllo esamina le richieste di accesso accordando solo quelle che sono autorizzate da una regola

- permettono agli utenti di concedere o revocare dei diritti di accesso sugli oggetti, a loro discrezione

- vantaggio: sono estremamente flessibili e adatte a numerosi contesti applicativi

  svantaggio: non forniscono alcun controllo sul flusso di informazioni nel sistema, non impongono restrizioni sull'uso che un utente fa dell'informazione, una volta acceduta.

Politiche mandatorie

- regolano l'accesso ai dati mediante la definizione di classi di sicurezza per i soggetti e gli oggetti del sistema

  le classi di sicurezza sono ordinate parzialmente (o totalmente) da una relazione d'ordine

- la classe di sicurezza assegnata ad un oggetto rappresenta il livello di sensitività dell'oggetto: maggiore è la classe assegnata ad un oggetto, più ingente sarà il danno derivante dal rilascio delle informazioni in esso contenute a soggetti non autorizzati

- la classe di sicurezza assegnata ad un soggetto è una misura del grado di fiducia che si ha nel fatto che tale soggetto non commetta violazioni.

Politiche mandatorie

- il controllo dell'accesso è regolato da una serie di *assiomi di sicurezza* che stabiliscono la relazione fra la classe di un soggetto e quella di un oggetto affinché al primo sia concesso di esercitare un modo di accesso sul secondo

  tale relazione dipende dal modo di accesso considerate

- sono applicate in ambienti, quali quello militare, dove la quantità di informazione da proteggere è elevata, ci sono forti esigenze di protezione ed è possibile classificare rigidamente gli elementi del sistema

- i sistemi che adottano una politica mandatoria sono spesso indicati come *sistemi multilivello*.

Politiche mandatorie

- possono essere classificate anche come politiche per il controllo del flusso, poiché evitano che le informazioni una volta accedute vengano trasferite verso oggetti con classificazione inferiore e quindi più accessibili

  $\Rightarrow$ c'è un controllo completo sul sistema di autorizzazione

- la flessibilità è però ridotta e la circolazione di informazioni tra gli utenti è più difficile.

CONTROLLO DELL'ACCESSO

- la politica mandatoria e la politica discrezionale non sono mutuamente esclusive, possono cioè essere applicate insieme

- la politica mandatoria non controlla più le richieste di accesso ma le autorizzazioni che vengono assegnate ad un soggetto, mentre alla politica discrezionale è affidato il compito di controllare le richieste di accesso.

MODELLI PER IL CONTROLLO DELL'ACCESSO

Derivano dai modelli precedenti per la protezione di risorse in un sistema operativo

differenze fondamentali tra i due ambienti:

- maggior numero di oggetti da proteggere
- diversi livelli di granularità (relazione, tupla, o singolo attributo)
- protezione di strutture completamente logiche (viste) invece di risorse reali (file)
- diversi livelli architetturali con requisiti di protezione differenti
- rilevanza non solo della rappresentazione fisica dei dati, ma anche della loro semantica.

MODELLO DI BASE

modello concettuale di riferimento per rappresentare le regole di autorizzazione nei modelli che adottano una politica discrezionale

tre componenti fondamentali:

- un insieme di *oggetti* O (cioè di entità su cui viene richiesto l'accesso);

- un insieme di *soggetti* S (cioè di entità che richiedono gli accessi agli oggetti);

- una *matrice di accesso* M, in cui ogni riga corrisponde ad un soggetto, ogni colonna ad un oggetto, e l'elemento $M[s, o]$ contiene i diritti di accesso che s può esercitare su o

  $M[s, o]$=`null` significa che s non ha nessun diritto su o, se $M[s, o]$=`all` che s ha su o tutti i diritti previsiti dal modello.

Esempio

Matrice di accesso per la relazione `Impiegati`

	Imp#	**Nome**	**Data_A**	**Stipendio**
dirigenti	`all`	`all`	`all`	`all`
impiegati	`read`	`read`	`read`	`null`

i `dirigenti` possono esercitare tutti i diritti su ogni attributo della relazione, mentre gli `impiegati` possono leggere tutti gli attributi, ad eccezione dell'attributo `Stipendio`.

Matrice di accesso

- la matrice di accesso è solo un modello concettuale: in molte situazioni reali la matrice è troppo sparsa (ogni soggetto ha accesso solo ad una piccola porzione della base di dati)

- le sue dimensioni sono troppo elevate per essere effettivamente implementata come matrice

- implementazione come insieme di

  1. *access control list*

  2. *capability list*.

Matrice di accesso

- **access control list**

  la access control list associata ad un oggetto o contiene un elemento per ogni soggetto s del sistema che ha qualche diritto su o, tale elemento contiene la lista dei diritti che s ha su o

- **capability list**

  la capability list associata ad un soggetto s contiene un elemento per ogni oggetto o su cui s ha qualche diritto, tale elemento contiene i diritti che s ha su o.

ESTENSIONI AL MODELLO DI BASE

- nel modello di base una regola di autorizzazione può essere rappresentata come una tripla `(s,o,m)`, dove `s` è il soggetto, `o` è l'oggetto e `m` è il modo di accesso che `s` può esercitare su `o`

- il modello di base è quindi in grado di modellare il controllo in base al nome, ad un livello di granularità arbitraria: `o` può infatti essere un'intera relazione, un insieme di attributi, o un singolo attributo

- per modellare forme più sofisticate di controllo dell'accesso, è però necessario modificare opportunamente la struttura delle regole di autorizzazione.

ESTENSIONI AL MODELLO DI BASE

- **controllo in base al contenuto**: si introduce un *predicato ausiliario* `p`, utilizzato per esprimere condizioni sul valore dell'oggetto

- una regola di autorizzazione (`s,o,m,p`) autorizza il soggetto `s` ad esercitare il modo di accesso `m` sull'oggetto `o` *solo* se `o` soddisfa la condizione espressa da `p`

  esempio: (`impiegati,Impiegati,read,`
  `Stipendio,Stipendio` $<$ 2000)

- `p` può essere anche esprimere vincoli aggiuntivi (un certo accesso può avvenire solo in determinate ore del giorno)

  $\Rightarrow$ `p` è la congiunzione di due predicati: un *predicato sui dati* `p(d)` e un *predicato di sistema* `p(s)`.

ESTENSIONI AL MODELLO DI BASE

- si può voler tener traccia del soggetto che specifica la regola (importante in sistemi non centralizzati)

  ad esempio, in genere una regola può essere modificata solo dal soggetto che l'ha creata

  $\Rightarrow$ regola diventa quintupla (`a,s,o,m,p`), dove `a` è il soggetto che specifica la regola di autorizzazione

- il modello prevede anche la possibiltà di *delegare* ad altri la facoltà di concedere privilegi su un determinato oggetto

  una regola di autorizzazione oltre ad autorizzare un soggetto ad esercitare un determinato modo di accesso su un oggetto, può anche consentire al soggetto la possibiltà di delegare ad altri il privilegio ricevuto.

ESTENSIONI AL MODELLO DI BASE

- la regola di autorizzazione viene estesa con un flag `f` che indica se il soggetto della regola può o meno delegare ad altri il privilegio ricevuto

- la regola (`a`,`s`,`o`,`m`,`p`,`f`) stabilisce quindi che `a` autorizza `s` ad esercitare il modo di accesso `m` su `o`, condizionatamente al soddisfacimento di `p`, ed inoltre autorizza (se `f`=1) o meno (se `f`=0) `s` a concedere ad altri il modo di accesso `m` su `o`.

ESTENSIONI AL MODELLO DI BASE

- è spesso utile specificare delle *procedure di ausilio*, da eseguire quando una regola è utilizzata nella validazione di una richiesta di accesso

  tali procedure devono essere eseguite quando una richiesta ha esito negativo (es.: registrazione della richiesta di accesso)

- insieme di coppie $\{(\mathtt{c}_1,\mathtt{pa}_1),\ \ldots,(\mathtt{c}_n,\mathtt{pa}_n)\}$, dove $\mathtt{pa}_i$ è una procedura di ausilio, e $\mathtt{c}_i$ è la condizione che deve essere verificata perché $\mathtt{pa}_i$ sia eseguita, $i = 1, \ldots, n$

  $\Rightarrow$ una regola di autorizzazione ha la struttura $(\mathtt{a},\mathtt{s},\mathtt{o},\mathtt{m},\mathtt{p},\mathtt{f},\{(\mathtt{c}_1,\mathtt{pa}_1),\ \ldots,(\mathtt{c}_n,\mathtt{pa}_n)\})$.

ESTENSIONI AL MODELLO DI BASE

Componenti di una regola di autorizzazione

`a`	soggetto che specifica la regola di autorizzazione
`s`	soggetto a cui è concessa l'autorizzazione
`o`	oggetto su cui l'autorizzazione è concessa
`m`	modo di accesso concesso
`p`	predicato ausiliario: esprime condizioni sul contenuto di `o`
`f`	flag: indica se il privilegio specificato è delegabile
`pa`	procedura di ausilio
`c`	condizione per l'esecuzione della procedura di ausilio

la maggior parte dei sistemi reali utilizzano però regole del tipo (`s`,`o`,`m`,`p`), cioè regole che modellano soltanto il controllo in base al nome e al contenuto.

MODELLO DI BELL E LA PADULA

- modello di riferimento per sistemi che adottano una politica di tipo mandatorio

- *oggetti*: entità passive che contengono informazioni da proteggere

  soggetti: entità attive che richiedono accesso agli oggetti (*utenti* e *processi*)

  modi di accesso: tipi di accesso di un soggetto su un oggetto:

  – `read`: leggere ma non modificare

  – `append`: modificare ma non leggere

  – `write`: sia modificare che leggere

  – `execute`: eseguire ma non leggere né modificare direttamente (modalità usata per programmi applicativi, utility, ecc.).

MODELLO DI BELL E LA PADULA

- i soggetti e gli oggetti del sistema vengono classificati mediante l'assegnamento di una *classe di accesso*

  una classe di accesso è costituita da due componenti: un *livello di sicurezza* ed un *insieme di categorie*

- il livello di sicurezza è un elemento di un insieme totalmente ordinato

  ad esempio: `Top Secret` (`TS`), `Secret` (`S`), `Confidential` (`C`) e `Unclassified` (`U`), dove: $\mathtt{TS} > \mathtt{S} > \mathtt{C} > \mathtt{U}$

- l'insieme di categorie è un insieme non ordinato di elementi, che dipendono dal tipo di ambiente considerato e dall'area applicativa in cui l'informazione deve essere utilizzata

  ad esempio: `Army`, `Navy`, `Air Force`, `Nuclear`.

Relazione di dominanza

Una classe di accesso $\mathtt{c}_1 = (\mathtt{L}_1,\mathtt{SC}_1)$ domina una classe di accesso $\mathtt{c}_2 = (\mathtt{L}_2,\mathtt{SC}_2)$, $(\mathtt{c}_1 \geq \mathtt{c}_2)$, se entrambe le seguenti condizioni sono verificate:

- il livello di sicurezza di $\mathtt{c}_1$ è maggiore o uguale al livello di sicurezza di $\mathtt{c}_2$ (cioè $\mathtt{L}_1 \geq \mathtt{L}_2$)

- l'insieme di categorie di $\mathtt{c}_1$ include l'insieme di categorie di $\mathtt{c}_2$ (cioè $\mathtt{SC}_1 \supseteq \mathtt{SC}_2$).

se $L_1 > L_2$ e $SC_1 \supset SC_2$, si dice che $\mathtt{c}_1$ *domina strettamente* $\mathtt{c}_2$ $(\mathtt{c}_1 > \mathtt{c}_2)$

$\mathtt{c}_1$ e $\mathtt{c}_2$ si dicono *incomparabili* $(\mathtt{c}_1 <> \mathtt{c}_2)$ se né $\mathtt{c}_1 \geq \mathtt{c}_2$ né $\mathtt{c}_2 \geq \mathtt{c}_1$ valgono.

Esempio

classi di accesso:

$$c_1 = (\mathtt{TS}, \{\mathtt{Nuclear}, \mathtt{Army}\})$$
$$c_2 = (\mathtt{TS}, \{\mathtt{Nuclear}\})$$
$$c_3 = (\mathtt{C}, \{\mathtt{Army}\})$$

- $c_1 \geq c_2$
- $c_1 > c_3$ ($\mathtt{TS} > \mathtt{C}$ e $\{\mathtt{Army}\} \subset \{\mathtt{Nuclear}, \mathtt{Army}\}$)
- $c_2 <> c_3$ ($c_2 \not\geq c_3$ perchè $\{\mathtt{Nuclear}\} \not\supseteq \{\mathtt{Army}\}$ e $c_3 \not\geq c_2$ perchè $\mathtt{TS} > \mathtt{C}$).

MODELLO DI BELL E LA PADULA

Lo stato del sistema è descritto dalla quadrupla (A,M,$\mathcal{L}$,G), dove:

- A è *l'insieme degli accessi correnti* triple della forma (s,o,m): il soggetto s sta esercitando la modalità di accesso m sull'oggetto o

- M è *la matrice degli accessi*: informazioni sui diritti dei soggetti sugli oggetti

- $\mathcal{L}$ è *la funzione di livello*: associa ad ogni elemento del sistema la sua classe di accesso
  $\mathcal{L} : O \cup S \rightarrow \mathcal{C}$

- G è la *gerarchia di oggetti corrente* cioè l'insieme degli oggetti correntemente accessibili dai soggetti (rappresentata da un albero).

Assiomi

- **Proprietà di sicurezza semplice**

  uno stato $(\mathtt{A},\mathtt{M},\mathcal{L},\mathtt{G})$ soddisfa la proprietà di sicurezza semplice se per ogni elemento $\mathtt{a} = (\mathtt{s}, \mathtt{o}, \mathtt{m}) \in \mathtt{A}$ una delle seguenti condizioni è verificata:

  1. $\mathtt{m} = \mathtt{execute}$ oppure $\mathtt{m} = \mathtt{append}$

  2. $\mathtt{m} = \mathtt{read}$ oppure $\mathtt{m} = \mathtt{write}$ e $\mathcal{L}(\mathtt{s}) \geq \mathcal{L}(\mathtt{o})$.

  un soggetto con classe di accesso $(\mathtt{C},\{\mathtt{Army}\})$ non può leggere dati con classi di accesso $(\mathtt{C},\{\mathtt{Navy},\mathtt{Air\ Force}\})$ o $(\mathtt{U},\{\mathtt{Air\ Force}\})$

- evita che soggetti leggano informazioni con classe di accesso maggiore o incomparabile, e assicura che i soggetti abbiano accesso diretto solo alle informazioni per cui hanno la necessaria classificazione.

Assiomi

- **Star ($*$) Proprietà**

  uno stato (A,M,$\mathcal{L}$,G) sodddisfa la $*$-proprietà, se per ogni elemento a$=$ $(\mathtt{s},\mathtt{o},\mathtt{m}) \in \mathtt{A}$ una delle seguenti condizioni è verificata:

  1. m$=$ read oppure m $=$ execute

  2. m$=$ append e $\mathcal{L}(\mathtt{s}) \leq \mathcal{L}(\mathtt{o})$

  3. m$=$ write e $\mathcal{L}(\mathtt{s}) = \mathcal{L}(\mathtt{o})$

  un soggetto con classe di accesso (C,{Army,Nuclear}) non può scrivere informazioni in oggetti con classe di accesso (U,{Army,Nuclear})

- la $*$-proprietà è definita per prevenire il flusso di informazioni verso classi di accesso minori o non comparabili.

MODELLO DI AUTORIZZAZIONE DEL SYSTEM R

- gli oggetti del modello sono relazioni (sia di base che viste)

- i privilegi previsti sono:

  – `alter`: aggiungere una nuova colonna ad una relazione

  – `index`: creare un indice su una relazione

  – `delete`: cancellare tuple da una relazione

  – `insert`: inserire tuple in una relazione

  – `select`: selezionare tuple da una relazione

  – `update`: modificare valori di attributi in tuple di una relazione

non esiste il privilegio di `drop`.

MODELLO DI AUTORIZZAZIONE DEL SYSTEM R

- il modello implementa una politica di tipo discrezionale e supporta il controllo dell'accesso in base sia al nome che al contenuto

- il sistema è un sistema chiuso: un accesso è concesso solo se esiste una esplicita regola che lo autorizza

- l'amministrazione dei privilegi è decentralizzata mediante ownership: quando un utente crea una relazione, riceve automaticamente tutti i diritti di accesso su di essa ed anche la possibilità di delegare ad altri tali privilegi.

Delega dei privilegi

- la delega dei privilegi avviene mediante la *grant option*: se un privilegio è concesso con grant option l'utente che lo riceve può non solo esercitare il privilegio, ma anche concederlo ad altri

- la grant option è analoga al flag del modello di base

- un utente può concedere un privilegio su una determinata relazione solo se è il proprietario della relazione, o ha ricevuto tale privilegio con grant option.

L'operazione di GRANT

`GRANT` *Lista Privilegi* | `ALL[PRIVILEGES]`
`ON` *Lista Relazioni* | *Lista Viste*
`TO` *Lista Utenti* | `PUBLIC`
`[WITH GRANT OPTION];`

- si possono garantire privilegi sia su relazioni che su viste
  i privilegi `alter` e `index` si applicano solo a relazioni

- i privilegi si applicano ad intere relazioni (o viste)
  per il privilegio di `update` è necessario specificare le colonne a cui si applica

- le parole chiave `ALL` o `ALL PRIVILEGES` (equivalenti) consentono di concedere con un solo comando tutti i privilegi su una determinata relazione
  non possono essere utilizzate su viste.

L'operazione di GRANT

- con un unico comando di `GRANT` si possono concedere più privilegi su una stessa relazione e concedere privilegi sulla stessa relazione a più utenti (in entrambi i casi l'ordine è irrilevante)

- un comando di `GRANT` con soggetto `PUBLIC` è equivalente ad una delega a tutti gli utenti

- se la clausola `WITH GRANT OPTION` non è specificata l'utente che riceve i privilegi non può concederli ad altri utenti

- i privilegi si ogni utente possiede sono divisi in: *privilegi delegabili* (concessi con grant option) e *privilegi non delegabili* (senza grant option).

Esempio

```
GRANT update(Stipendio,Premio_P) ON Impiegati
   TO Rossi;
GRANT select,insert ON Impiegati
   TO Verdi,Gialli;
GRANT ALL PRIVILEGES ON Impiegati
   TO Neri WITH GRANT OPTION;
```

- `Rossi` può modificare gli attributi `Stipendio` e `Premio_P` delle tuple della relazione `Impiegati`

- `Verdi` e `Gialli` possono selezionare ed inserire tuple nella relazione `Impiegati`

- `Neri` ha tutti i privilegi sulla relazione `Impiegati` e può delegare ad altri tali privilegi.

Esempio

```
Bianchi: GRANT select,insert ON Impiegati
         TO Verdi WITH GRANT OPTION;
Bianchi: GRANT select ON Impiegati
         TO Rossi WITH GRANT OPTION;
Verdi: GRANT select,insert ON Impiegati
         TO Rossi;
```

- `Rossi` ha il privilegio di `select` (ricevuto sia da `Bianchi` che da `Verdi`) e `insert` (ricevuto da `Verdi`) sulla relazione `Impiegati`

- `Rossi` può garantire ad altri utenti il privilegio di `select` (in quanto lo ha ricevuto da `Bianchi` con grant option), ma non quello di `insert`.

Cataloghi Sysauth e Syscolauth

- le regole di autorizzazione specificate dagli utenti sono memorizzate in due cataloghi di sistema di nome `Sysauth` e `Syscolauth`, implementati come relazioni

- una tupla di `Sysauth` ha i seguenti attributi:

  `id_utente`: identificatore dell'utente a cui sono concessi i privilegi;

  `nome`: nome della relazione su cui sono concessi i privilegi;

  `creatore`: utente che ha creato la relazione;

  `tipo` $\in$ {`R`,`V`}: indica se l'oggetto è una relazione (`tipo`=‘`R`’) o una vista (`tipo`=‘`V`’);

`alter` $\in$ {Y,N}: indica se il soggetto ha (`alter`='Y') o meno (`alter`='N') il privilegio di aggiungere una nuova colonna alla relazione

Sysauth contiene un analogo attributo per i privilegi `index`, `delete`, `insert` e `select`;

`update` $\in$ {ALL,SOME,N}: indica se il soggetto ha il privilegio di `update` su tutte (`update`='ALL'), alcune (`update`='SOME'), o nessuna (`update`='N') colonna della relazione;

`grantopt` $\in$ {Y,N}: indica se i privilegi sono delegabili (`grantopt`='Y') o meno (`grantopt`='N').

Esempio

id_ut.	nome	creat.	t	a	i	d	ins	s	u	go
Bianchi	Impiegati	Bianchi	R	Y	Y	Y	Y	Y	ALL	Y
Verdi	Impiegati	Bianchi	R	N	N	N	Y	Y	N	Y
Rossi	Impiegati	Bianchi	R	N	N	N	N	Y	N	Y
Rossi	Impiegati	Bianchi	R	N	N	N	Y	Y	N	N

si suppone che la relazione `Impiegati` sia stata creata da `Bianchi`

per ogni relazione (o vista) su cui un utente ha privilegi, sono presenti al più due tuple nel catalogo `Sysauth`: una rappresentante i privilegi delegabili (`grantop`=‘`Y`’) e una rappresentante i privilegi non delegabili (`grantop`=‘`N`’).

Cataloghi Sysauth e Syscolauth

- le colonne su cui il privilegio di `update` può essere esercitato sono contenute nel catalogo `Syscolauth` che contiene una tupla (`id_utente`,`nome`,`colonna`,`grantopt`) per ogni colonna della relazione `nome` su cui l'utente identificato da `id_utente` può esercitare il privilegio di `update`

<u>Esempio</u>

id_utente	**nome**	**colonna**	**grantopt**
Bianchi	Impiegati	Imp#	Y
Bianchi	Impiegati	Nome	Y
Bianchi	Impiegati	Mansione	Y
Bianchi	Impiegati	Data_A	Y
Bianchi	Impiegati	Stipendio	Y
Bianchi	Impiegati	Premio_P	Y
Bianchi	Impiegati	Dip#	Y

Cataloghi Sysauth e Syscolauth

- quando un utente `u` esegue un comando di `GRANT`, il meccanismo di controllo accede ai cataloghi `Sysauth` e `Syscolauth` per determinare se `u` ha il diritto di delegare i privilegi specificati nel comando

- l'insieme dei privilegi delegabili che l'utente `u` possiede è intersecato con l'insieme dei privilegi specificati nel comando di `GRANT`

  se l'intersezione è vuota, il comando non viene eseguito

  se l'intersezione coincide con i privilegi specificati nel comando, il comando viene eseguito, e tutti i privilegi specificati vengono concessi

  altrimenti il comando viene eseguito parzialmente, cioè solo i privilegi contenuti nell'intersezione vengono accordati.

Esempio

```
Bianchi: GRANT select,insert On Impiegati
         TO Gialli WITH GRANT OPTION;
Verdi: GRANT update, ON Impiegati
         TO Gialli WITH GRANT OPTION;
Rossi: GRANT select,insert ON Impiegati
         TO Neri;
```

- il primo comando di `GRANT` viene eseguito (`Bianchi` è il proprietario della relazione `Impiegati`)

- il secondo non viene eseguito (`Verdi` non possiede il privilegio di `update` sulla relazione `Impiegati`)

- il terzo viene parzialmente eseguito (`Rossi` ha i privilegi di `select` ed `insert` sulla relazione `Impiegati` ma non ha la grant option per `insert` $\Rightarrow$ a `Neri` viene concesso solo privilegio di `select`).

L'operazione di REVOKE

`REVOKE` *Lista Privilegi* | `ALL[PRIVILEGES]`
`[ON` *Lista Relazioni* | *Lista Viste*`]`
`FROM` *Lista Utenti* | `PUBLIC;`

- un utente può revocare solo i privilegi che lui ha concesso

- è possibile revocare più privilegi con un unico comando di `REVOKE`, ed un unico comando di `REVOKE` può essere utilizzato per revocare gli stessi privilegi sulla stessa relazione ad utenti diversi

  la parola chiave `PUBLIC` indica che la revoca si applica a tutti gli utenti, mentre `ALL` (o `ALL PRIVILEGES`) indica che tutti i privilegi previsti dal modello sono revocati, se la clausola `ON` non è specificata, la revoca si applica a tutte le relazioni della base di dati.

L'operazione di REVOKE

- il comando di `REVOKE` non consente di revocare in modo selettivo il diritto di `update` (cioè nomi di colonne)

- non è possibile revocare solo la grant option: occorre revocare tutto il privilegio concesso con grant option e successivamente rieseguire il comando di `GRANT` senza grant option

- quando si esegue una operazione di revoca, l'utente a cui i privilegi sono stati revocati perde tali privilegi, a meno che essi non gli provengano anche da altre sorgenti *indipendenti* da quella che effettua la revoca.

Esempio

```
REVOKE select, insert ON Impiegati
    FROM Verdi,Gialli;
REVOKE update ON Impiegati FROM Rossi;
REVOKE ALL ON Impiegati FROM Neri;
```

- vengono revocati a `Verdi` ed a `Gialli` i diritti di selezionare ed inserire tuple nella relazione `Impiegati`

- revoca a `Rossi` il diritto di modificare tuple della relazione `Impiegati`

- revoca a `Neri` tutti i diritti che possedeva sulla relazione `Impiegati`.

Esempio

```
Bianchi: GRANT select ON Impiegati
          TO Verdi WITH GRANT OPTION;
Bianchi: GRANT select ON Impiegati
          TO Gialli WITH GRANT OPTION;
Verdi: GRANT select ON Impiegati TO Rossi;
Gialli: GRANT select ON Impiegati TO Rossi;
Verdi: REVOKE select ON Impiegati FROM Rossi;
```

l'utente `Rossi` continua ad avere il privilegio di `select` sulla relazione `Impiegati`, anche se tale privilegio gli è stato revocato da `Verdi`, in quanto `Rossi` ha indipendentemente ottenuto tale privilegio da `Gialli`.

Revoca ricorsiva

- un'operazione di revoca del modo di accesso `m` sulla relazione `rel` all'utente u_1 da parte dell'utente u_2 ha l'effetto non solo di far perdere ad u_1 il modo di accesso `m` sulla relazione `rel`, se u_1 non ha ottenuto tale privilegio da fonti indipendenti, ma anche di modificare il sistema portandolo in uno stato equivalente a quello in cui si sarebbe trovato se u_2 non avesse mai concesso ad u_1 il modo di accesso `m` sulla relazione `rel`

- dopo un'operazione di revoca il sistema deve ricorsivamente revocare tutti i privilegi che non avrebbero potuto essere concessi se l'utente specificato nel comando di revoca non avesse ricevuto il privilegio revocato.

Esempio

```
REVOKE select ON Impiegati FROM Gialli;
```

- sia `Gialli` che `Rossi` perdono il privilegio di `select` sulla relazione `Impiegati`: `Rossi` non avrebbe potuto ricevere il privilegio di `select` da `Gialli` se quest'ultimo non lo avesse ricevuto, con grant option, da `Bianchi` e non esiste nessuna fonte indipendente da `Gialli` da cui `Rossi` ha ricevuto il privilegio di `select`.

Revoca ricorsiva

- *Siano* $G_1, \ldots, G_n$ *una sequenza di operazioni di grant di un singolo privilegio sulla stessa relazione, tali che* $\forall i, j = 1, \ldots, n$, *se* $i < j$, *allora* G_i *è eseguita prima di* G_j. *Sia* R_i *la revoca del privilegio concesso con l'operazione* G_i. *La semantica della revoca ricorsiva impone che lo stato del sistema dopo l'esecuzione della sequenza:*

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

  sia identico allo stato che si avrebbe dopo l'esecuzione della sequenza:

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

- è utilizzata dalla maggior parte dei DBMS relazionali oggi in commercio, quali per esempio Oracle, Informix e DB2.

Revoca ricorsiva

- necessità di determinare se un privilegio proviene da sorgenti indipendenti rispetto a quella specificata nel comando di revoca $\Rightarrow$ `Sysauth` e `Syscolauth` sono modificati per mantenere, per ogni privilegio, anche l'utente che ha concesso il privilegio (*grantor*)

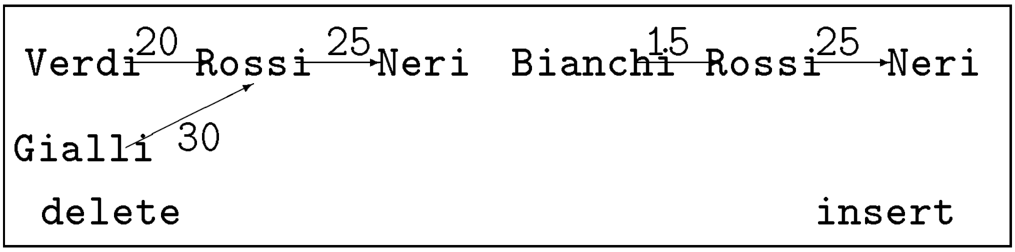
- ogni colonna relativa ad un tipo di privilegio in `Sysauth` contiene (invece di '`Y`' e '`N`') un *timestamp* che denota il tempo in cui il privilegio è stato concesso

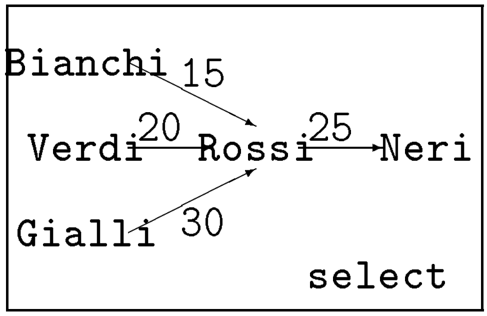
  il valore `0` indica che l'utente non ha quel privilegio, mentre un valore $t \neq 0$ indica che il privilegio è stato garantito all'utente al tempo t

  privilegi garantiti con lo stesso comando di `GRANT` hanno lo stesso timestamp.

Esempio

id_utente	grantor	nome	t	s	i	d	go
Rossi	Bianchi	Impiegati	R	15	15	0	Y
Rossi	Verdi	Impiegati	R	20	0	20	Y
Neri	Rossi	Impiegati	R	25	25	25	Y
Rossi	Gialli	Impiegati	R	30	0	30	Y





Esempio

- al tempo 35 `Verdi` esegue il comando:

  `REVOKE ALL ON Impiegati FROM Rossi;`

- si elimina la tupla
  `(Rossi,Verdi,Impiegati,R,20,0,20,Y)`
  dal catalogo `Sysauth`

- si determinano quali privilegi non avrebbero potuto essere concessi se `Rossi` non avesse ricevuto da `Verdi` i privilegi revocati:

  1. si forma la lista dei timestamp dei privilegi delegabili rimanenti a `Rossi`, dopo che i privilegi garantitigli da `Verdi` sono stati eliminati:

     `delete`={30}, `insert`={15}, `select`={15,30}

Esempio

2 si forma la lista dei timestamp dei privilegi concessi da `Rossi` ad altri utenti (nell'es. solo a `Neri`):

`delete`={25}, `insert`={25}, `select`={25}

3 un privilegio garantito da `Rossi` è revocato se `Rossi` non ha più il privilegio, oppure se ha ancora il privilegio ma con un timestamp maggiore

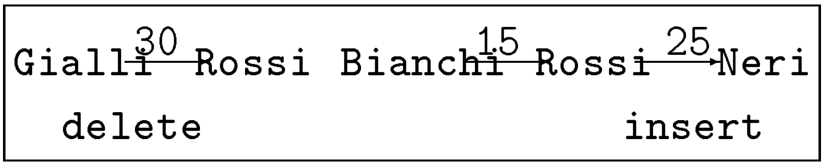
4 i passi 1,2,3 vengono eseguiti per ogni utente i cui privilegi sono stati modificati in seguito all'operazione di revoca.

Esempio

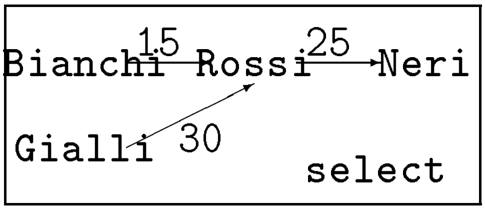
- si revoca il privilegio di `delete` a `Neri` (il timestamp associato al privilegio di `delete` per `Rossi` -30- è maggiore di 25, cioè del timestamp del privilegio di `delete` garantito da `Rossi` a `Neri`)

- `Neri` mantiene sia il privilegio di `insert` sia quello di `select` (il privilegio di `insert` concesso da `Rossi` a `Neri` era stato concesso a `Rossi` da `Bianchi` e `Rossi` avrebbe potuto concedere al tempo 25 il privilegio di `select` a `Neri` anche se non avesse ricevuto tale privilegio da `Verdi` al tempo 10, grazie alla autorizzazione ricevuta al tempo 15 da `Bianchi`).

Esempio



Gialli
30
Rossi
Bianchi
15
Rossi
25
Neri
delete
insert



Bianchi
15
Rossi
25
Neri
Gialli
30
select

Autorizzazione su viste

- le viste permettono di supportare il controllo dell'accesso in base al contenuto

  esempio: per autorizzare un utente a selezionare solo le tuple della relazione `Impiegati` relative ad impiegati che non guadagnano più di due milioni di lire, si definisce una vista che seleziona dalla relazione `Impiegati` le tuple che soddisfano tale condizione e si concede all'utente il privilegio di `select` sulla vista, invece che sulla relazione di base

- permettono di delegare privilegi su singole colonne di relazioni: basta definire una vista come proiezione sulle colonne su cui si vogliono concedere i privilegi

- permettono di delegare privilegi *statistici* (media, somma, ecc.).

Autorizzazione su viste

- i privilegi che l'utente che crea una vista può esercitare sulla vista dipendono da:

  1. la semantica della vista, ovvero la sua definizione in termini delle relazioni o viste componenti

  2. le autorizzazioni che l'utente possiede sulle relazioni o viste componenti

- i privilegi `alter` e `index` non si applicano alle viste

- non si ottongono privilegi relativi ad operazioni non consentite sulla vista (ad esempio, se la vista è definita come un join non è possibile modificare le sue colonne).