Azzolini Riccardo 2021-05-27

Controllo dell'accesso

1 Controllo dell'accesso

Il **controllo dell'accesso** regola le operazioni che si possono compiere sui dati e sulle risorse di un sistema. Più precisamente, il suo scopo è *limitare e controllare le azioni che i soggetti effettuano sulle risorse del sistema*, prevenendo azioni accidentali o deliberate che potrebbero compromettere la correttezza e la sicurezza dei dati.

Si definisce **soggetto** qualsiasi entità che svolge azioni all'interno del sistema, come ad esempio un utente, un processo, ecc. Invece, quali siano le **risorse** dipende dal tipo di sistema considerato; ad esempio, nell'ambito di un filesystem esse sono i file, mentre in un DBMS esse sono i dati.

Il controllo dell'accesso avviene per mezzo di tre concetti fondamentali:

- le *politiche di sicurezza*;
- le *autorizzazioni*;
- il *meccanismo di controllo dell'accesso* o *reference monitor*.

Le **politiche di sicurezza** sono regole e principi secondo cui un'organizzazione vuole che siano protette le proprie informazioni. In altre parole, esse sono un insieme di direttive ad alto livello che esprimono le scelte compiute da un'organizzazione in merito alla protezione dei propri dati, specificando come le risorse possono essere utilizzate. Le politiche di sicurezza sono legate ai processi che l'organizzazione deve svolgere, alle esigenze aziendali, e sono spesso definite in linguaggio naturale; un esempio è:

> "le valutazioni psicologiche di un impiegato possono essere viste solo dal suo responsabile"

Le politiche di sicurezza vengono poi tradotte in un insieme di **autorizzazioni**, le quali stabiliscono gli specifici diritti che i vari soggetti abilitati ad accedere al sistema possono esercitare sugli oggetti. Ciascuna regola specifica un *soggetto*, un *oggetto* e un *privilegio* (un diritto che questo soggetto può esercitare su questo oggetto). Ad esempio, supponendo che Mario Rossi sia il responsabile di Giovanni Bianchi, un'autorizzazione conforme alla precedente politica di sicurezza potrebbe essere:

```
(
  mario rossi,
  ValutazionePsicologica(giovanni bianchi),
  lettura
)
```

Il **meccanismo di controllo dell'accesso**, detto anche **reference monitor**, è il componente del sistema che effettua concretamente il controllo dell'accesso: esso intercetta ogni comando inviato e analizza le autorizzazioni per stabilire se il soggetto richiedente può essere autorizzato (totalmente o parzialmente) a compiere l'accesso richiesto o se invece l'accesso deve essere negato.

Prima che una richiesta arrivi al reference monitor dev'esserci una fase di autenticazione, dato che per il controllo dell'accesso si assume che l'identità del soggetto richiedente sia già stata verificata.

2 Sistemi aperti e chiusi

Ci sono due moti di interpretare le autorizzazioni per il controllo dell'accesso:

- In un **sistema chiuso** le autorizzazioni indicano per ogni soggetti i privilegi che esso *può* esercitare sugli oggetti del sistema, mentre tutti i privilegi non esplicitamente autorizzati sono negati. In altre parole, le autorizzazioni hanno una semantica positiva, specificano ciò che è ammesso fare.
- In un **sistema aperto** le autorizzazioni stabiliscono i privilegi che ciascun soggetto *non può* esercitare sugli oggetti del sistema, mentre tutti i privilegi non esplicitamente negati sono autorizzati. Le autorizzazioni hanno dunque una semantica negativa, specificano ciò che non è ammesso fare.

Il vantaggio di un sistema chiuso è la sicurezza: si modella esattamente ciò che ciascun soggetto può fare. Invece, il vantaggio di un sistema aperto è esattamente l'opposto: non è necessario immaginare e modellare tutte le possibili azioni che i soggetti potrebbero voler compiere sugli oggetti, quindi si ottiene una minore rigidità e si elimina il rischio che delle richieste lecite vengano negate solo perché ci si è dimenticati di autorizzarle, ma in compenso si corre il rischio che delle risorse rimangano esposte se ci si dimentica di negarvi l'accesso.

I sistemi come filesystem e DBMS implementano sostanzialmente sistemi chiusi, mentre i sistemi aperti vengono in genere usati quando ci sono solo poche risorse da proteggere, pochi privilegi da negare. In ambito Web si usano solitamente dei sistemi ibridi, che permettono di definire autorizzazioni sia positive che negative, combinando così i benefici dei sistemi chiusi e aperti.

3 Tipi di politiche

A breve verrà presentato nel dettaglio il controllo dell'accesso in SQL standard, ma per fare ciò è prima necessario introdurre i tipi di politiche per il controllo dell'accesso che esso supporta. Infatti, nel corso degli anni sono stati sviluppati diversi modi per definire le politiche di sicurezza e tradurle in autorizzazioni, in base anche all'evoluzione delle esigenze nel tempo. SQL standard supporta due tipi di politiche:

- *discrezionali* (*DAC*, *Discretional Access Control*)
- *basate sui ruoli* (*RBAC*, *Role-Based Access Control*)

(poi alcuni DBMS estendono lo standard permettendo anche politiche diverse e più complicate).

3.1 Politiche discrezionali

Le politiche **discrezionali** sono definite sulla base dell'*identità* del soggetto richiedente, e vengono tradotte in un insieme di autorizzazioni le quali stabiliscono esplicitamente, per ogni soggetto, i privilegi che questo può esercitare sugli oggetti del sistema. Di conseguenza, se si hanno tanti soggetti e tante risorse è può essere necessario specificare numerose autorizzazioni.

Il vantaggio delle politiche discrezionali è l'estrema flessibilità, la possibilità di dettagliare esattamente tutte le azioni che ciascun soggetto può eseguire, mentre lo svantaggio è appunto il grande numero di autorizzazioni (proporzionale al prodotto tra il numero di soggetti, il numero di oggetti e il numero di possibili privilegi), che rende difficile la gestione delle autorizzazioni e il loro aggiornamento quando le esigenze di un soggetto cambiano, soprattutto considerando che spesso è necessario definire delle autorizzazioni solo per periodi di tempo limitati.

Queste politiche vengono dette *discrezionali* perché sono nate con il principio che un soggetto che ha ricevuto un privilegio possa a sua volta, a sua discrezione, concedere quello stesso privilegio ad altri soggetti (e poi eventualmente revocarlo). Sostanzialmente, l'idea era quella di facilitare la gestione delle (potenzialmente numerose) autorizzazioni delegandola agli stessi soggetti autorizzati. In pratica, ad esempio in SQL standard, quando si concede un privilegio a un soggetto si può dare o meno l'autorizzazione di passarlo ad altri.

3.2 Politiche basate sui ruoli

I primi RDBMS (DBMS relazionali) e le prime versioni dello standard SQL fornivano solo il controllo dell'accesso discrezionale, ma presto si capì la necessità di facilitare in qualche modo la gestione delle autorizzazioni, che erano spesso troppo numerose. Una

soluzione è raggruppare le autorizzazioni associandole non ai singoli soggetti ma ai *ruoli* che i soggetti ricoprono.

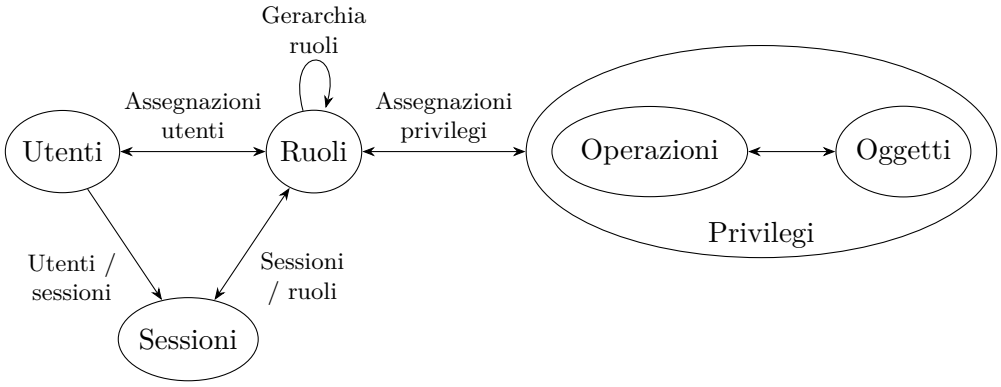
Un **ruolo** rappresenta una funzione all'interno di un'azienda o organizzazione (ad esempio "direttore", "commesso" o "cliente"). Con le politiche **basate sui ruoli** le autorizzazioni sono concesse non ai singoli soggetti ma ai ruoli, poi ciascun soggetto è abilitato a ricoprire uno o più ruoli, acquisendo così le associazioni a essi associate.

Le politiche basate sui ruoli hanno due principali vantaggi:

- Il numero di autorizzazioni da gestire è ridotto, perché un ruolo raggruppa solitamente numerosi privilegi, e un soggetto che viene abilitato a ricoprire tale ruolo acquisisce immediatamente tutti questi privilegi, senza bisogno di specificarli individualmente per il soggetto. Inoltre, ci sono tipicamente molti meno ruoli che soggetti.
- I ruoli sono più stabili rispetto ai soggetti: quando cambia la mansione di un soggetto nell'organizzazione è sufficiente revocare a tale soggetto l'autorizzazione a ricoprire i ruoli associati alla vecchia mansione e abilitarlo a ricoprire i ruoli corrispondenti alla nuova mansione, mentre le autorizzazioni dei ruoli sugli oggetti rimangono invariate. Invece, con le politiche discrezionali sarebbe necessario cancellare le autorizzazioni del soggetto su alcuni oggetti e aggiungerne delle altre.

4 Modello NIST per le politiche basate sui ruoli

Il controllo dell'accesso basato sui ruoli si diffuse così ampiamente (fu implementato in vari DBMS, ecc.) che il NIST definì un modello standard di riferimento, adottato poi in SQL standard a partire dallo standard SQL:1999 (SQL 3). Il modello è rappresentato dal seguente grafo:



- Un *privilegio* è l'associazione tra un'operazione e un oggetto su cui la si può fare.

- I privilegi sono associati ai ruoli a cui sono concessi (ad esempio, il ruolo "docente" potrebbe essere associato al privilegio "SELECT su tabella_1").
- I ruoli sono assegnati agli utenti. Un utente può anche ricoprire più ruoli.
- Oltre ai ruoli normalmente assegnati a esso, un utente può ricoprire temporaneamente altri ruoli mediante delle *sessioni*.
- Al fine di ridurre ulteriormente il numero di autorizzazioni da gestire, i ruoli possono essere organizzati in una **gerarchia**. Tale gerarchia è definita dalla **relazione di ereditarietà**, una relazione parziale sull'insieme dei ruoli denotata con $\geq$; dati due ruoli R_1 e R_2, $R_1 \geq R_2$ significa che:
  - R_1 eredita automaticamente tutti i privilegi assegnati a R_2;
  - tutti gli utenti associati al ruolo R_1 sono automaticamente associati anche al ruolo R_2.

5 Controllo dell'accesso in SQL standard

Il controllo dell'accesso in SQL standard adotta il paradigma di *sistema chiuso* (un accesso è consentito solo se esiste un'esplicita autorizzazione per esso) e implementa politiche sia *discrezionali* che *basate sui ruoli*. Gli oggetti a cui si concede l'accesso sono sia le relazioni di base (le tabelle) che le viste, e i principali privilegi previsti dal modello corrispondono alle operazioni effettuabili tramite i diversi comandi SQL (ad esempio SELECT, INSERT, UPDATE e DELETE).

L'amministrazione dei privilegi è *decentralizzata* mediante il concetto di **ownership**: l'utente che crea un oggetto (relazione di base o vista) diventa **owner**, **proprietario** di tale oggetto, ricevendo tutti i privilegi su di esso, la possibilità di concedere tali privilegi ad altri utenti e anche la possibilità di *delegare* ad altri utenti l'amministrazione di tali privilegi (come previsto dal modello di controllo dell'accesso discrezionale).

Tutte le operazioni di amministrazione dei privilegi vengono effettuate tramite due comandi SQL: *GRANT* e *REVOKE*.

6 Comando GRANT

Per concedere le autorizzazioni si usa il comando **GRANT**, che permette di:

- autorizzare privilegi sia agli utenti (DAC) che ai ruoli (RBAC);
- autorizzare un utente a ricoprire uno o più ruoli.

6.1 GRANT per l'autorizzazione di privilegi

Per l'autorizzazione di privilegi si usa il comando GRANT con la seguente sintassi:

```
GRANT {<lista privilegi> | ALL PRIVILEGES}
ON [<qualificatore oggetto>] <nome oggetto>
TO {<lista utenti> | <lista ruoli> | PUBLIC}
[WITH GRANT OPTION] [WITH HIERARCHY OPTION];
```

- `<lista privilegi>` specifica l'insieme dei privilegi da concedere. In alternativa, si può usare la parola chiave `ALL PRIVILEGES` per indicare che si vogliono concedere tutti i privilegi previsti dal modello.
- `<nome oggetto>` è il nome dell'oggetto della base di dati (che può essere una relazione, una vista, ecc.) sul quale si concedono i privilegi. Se necessario è possibile specificare un qualificatore dell'oggetto (ad esempio `TYPE` per indicare che l'oggetto è un tipo).
- I privilegi vengono concessi all'insieme di utenti elencati in `<lista utenti>`, oppure all'insieme di ruoli elencati in `<lista ruoli>`, oppure a tutti gli utenti e i ruoli del sistema se si usa la parola chiave `PUBLIC`.
- Con la clausola opzionale `WITH GRANT OPTION` si delega l'amministrazione dei privilegi agli utenti/ruoli a cui i privilegi vengono concessi: tali utenti (o gli utenti che ricoprono tali ruoli) potranno a loro volta concedere questi privilegi ad altri utenti/ruoli. Inoltre, essi potranno "espandere" la delega dell'amministrazione concedendo ad altri i privilegi con la GRANT OPTION. Ciò consente una grande flessibilità, ma comporta il rischio di perdere il controllo su chi può concedere i privilegi.
- La clausola opzionale `WITH HIERARCHY OPTION` riguarda *non* la gerarchia di ereditarietà tra ruoli, ma piuttosto quella tra relazioni (una delle caratteristiche object-oriented introdotte in SQL:1999): essa si applica solo per il privilegio SELECT e specifica che tale privilegio deve essere propagato anche a tutte le sotto-relazioni della (cioè le relazioni che ereditano dalla) relazione indicata nel comando GRANT.

I privilegi concessi con un comando GRANT si applicano a intere relazioni, ad eccezione di SELECT, INSERT e UPDATE, che possono essere applicati anche solo ad alcune colonne, elencando le colonne (separate da virgole) tra parentesi tonde dopo il nome del privilegio nel comando GRANT:

```
GRANT {SELECT | INSERT | UPDATE}(<lista colonne>) ON ...
```

Oltre ai privilegi corrispondenti ai comandi SQL è possibile concedere i seguenti:

- *REFERENCES*, che permette di utilizzare una colonna in un vincolo (ad esempio di chiave esterna) o in un'asserzione senza bisogno di avere il privilegio di accesso "vero e proprio" alla colonna/relazione;
- *TRIGGER*, che permette di specificare trigger che operano su una certa relazione;
- *UNDER*, che permette di creare sotto-tipi o sotto-relazioni;
- *USAGE*, che permette di utilizzare un oggetto dello schema nella definizione di un altro oggetto;
- *EXECUTE*, che permette l'esecuzione di una procedura o funzione (definita usando le estensioni procedurali di SQL).

Un utente può concedere un privilegio su un determinato oggetto se è il proprietario dell'oggetto oppure se ha ricevuto tale privilegio con la GRANT OPTION.

Alcuni esempi di comandi GRANT per l'autorizzazione di privilegi sono:

```
GRANT USAGE ON TYPE indirizzo TO giovanna WITH GRANT OPTION;
GRANT EXECUTE ON aggiornaClienti TO elena, direttore;
GRANT SELECT(nome, cognome), REFERENCES(codCli) ON Clienti TO marco;
GRANT DELETE, UPDATE ON Clienti TO direttore;
```

6.2 GRANT per l'autorizzazione ai ruoli

La sintassi del comando GRANT per abilitare utenti/ruoli a ricoprire ruoli è la seguente:

```
GRANT <lista ruoli concessi>
TO {<lista utenti> | <lista ruoli> | PUBLIC}
[WITH ADMIN OPTION];
```

- `<lista ruoli concessi>` indica l'insieme di ruoli che si vogliono concedere.
- I ruoli possono essere concessi a un insieme di utenti elencati in `<lista utenti>`, a un insieme di ruoli elencati in `<lista ruoli>` oppure a tutti gli utenti/ruoli del sistema con la parola chiave `PUBLIC`.
- La clausola opzionale `WITH ADMIN OPTION` è l'analogo per i ruoli della GRANT OPTION, cioè delega l'amministrazione dei ruoli che si stanno concedendo: quando si è abilitati a ricoprire un ruolo con l'ADMIN OPTION non solo si acquisiscono tutti i privilegi specificati per quel ruolo, ma si può anche concedere a terzi la possibilità di ricoprire il ruolo, opzionalmente ancora con l'ADMIN OPTION.

La possibilità di abilitare un ruolo a ricoprire un altro ruolo è il modo in cui SQL fornisce implicitamente la possibilità di strutturare i ruoli in una gerarchia: se un ruolo R_1 è abilitato a ricoprire un ruolo R_2 (con il comando `GRANT` R_2 `TO` R_1`;`), allora

- R_1 eredita tutte le autorizzazioni di R_2,
- tutti gli utenti che possono ricoprire il ruolo R_1 possono anche ricoprire il ruolo R_2,

ovvero si ha $R_1 \geq R_2$.

Alcuni esempi di comandi GRANT per l'autorizzazione ai ruoli sono:

```
GRANT direttore TO roberto WITH ADMIN OPTION;
GRANT commesso TO direttore;
```

7 Comando REVOKE

I privilegi e i ruoli concessi possono essere revocati tramite il comando **REVOKE**, che come GRANT ha due forme: una per i privilegi e una per i ruoli. Un privilegio o ruolo può essere revocato *solo dall'utente che l'ha concesso.*

7.1 REVOKE per la revoca di privilegi

La forma di REVOKE per i privilegi è la seguente:

```
REVOKE [{GRANT OPTION FOR | HIERARCHY OPTION FOR}] <lista privilegi>
ON [<qualificatore oggetto>] <nome oggetto>
FROM {<lista utenti> | <lista ruoli>}
{RESTRICT | CASCADE};
```

- Le clausole opzionali `GRANT OPTION FOR` e `HIERARCHY OPTION FOR` permettono rispettivamente di revocare la sola GRANT OPTION o HIERARCHY OPTION per i privilegi oggetto del comando di revoca, mantenendo invece il diritto a esercitare i privilegi stessi.
- `<lista privilegi>` indica l'insieme di privilegi oggetto del comando di revoca.
- `<qualificatore oggetto>` e `<nome oggetto>` identificano l'oggetto della base di dati su cui revocare i privilegi.
- I privilegi vengono revocati all'insieme di utenti elencati in `<lista utenti>` oppure all'insieme di ruoli elencati in `<lista ruoli>`.
- Le clausole `RESTRICT` e `CASCADE`, che verranno discusse a breve, determinano come gestire il caso in cui gli utenti/ruoli a cui si vogliono revocare i privilegi hanno sfruttato la GRANT OPTION per concedere tali privilegi ad altri utenti/ruoli.

Alcuni esempi di revoca di privilegi (o della GRANT OPTION) sono:

```
REVOKE GRANT OPTION FOR USAGE ON TYPE indirizzo FROM giovanna;
REVOKE DELETE ON Clienti FROM direttore;
```

7.2 REVOKE per la revoca di ruoli

La sintassi di REVOKE per i ruoli è invece:

```
REVOKE [ADMIN OPTION FOR] <lista ruoli revocati>
FROM {<lista utenti> | <lista ruoli>}
{RESTRICT | CASCADE};
```

- La clausola opzionale `ADMIN OPTION FOR` serve per revocare la sola ADMIN OPTION, mantenendo invece il diritto a ricoprire i ruoli oggetto del comando di revoca.
- *<lista ruoli revocati>* indica l'insieme dei ruoli oggetto del comando di revoca.
- I ruoli vengono revocati all'insieme di utenti elencati in *<lista utenti>* oppure all'insieme di ruoli elencati in *<lista ruoli>*. Revocare un ruolo a un altro ruolo significa eliminare la relazione di ereditarietà tra i due, "cancellare un tratto" della gerarchia: i privilegi del ruolo revocato non sono più ereditati dall'altro ruolo.
- Le clausole `RESTRICT` e `CASCADE` hanno una funzione analoga a quella che svolgono nella revoca dei privilegi.

Un esempio di questo comando è:

```
REVOKE direttore from roberto;
```

7.3 Semantica della revoca

L'esistenza della GRANT OPTION (per i privilegi, e dell'analoga ADMIN OPTION per i ruoli) ha due aspetti importanti nella semantica dell'operazione di revoca:

- La revoca di un privilegio (o ruolo) a un utente non comporta necessariamente la perdita di quel privilegio da parte dell'utente: se l'utente ha ricevuto lo stesso privilegio da altre fonti *indipendenti* dal soggetto che effettua la revoca, allora dopo la revoca può continuare a esercitare il privilegio (finché esso non gli viene revocato anche dalle altre fonti).
- Quando si revoca un privilegio (o ruolo) a un utente che l'aveva ricevuto con la GRANT OPTION e passato ad altri utenti/ruoli, il privilegio dovrebbe essere revocato anche a questi (purché essi non l'abbiano ottenuto anche da altre fonti indipendenti).

Un'altra conseguenza della revoca di un privilegio dovrebbe essere la cancellazione di eventuali oggetti dello schema (ad esempio viste) creati grazie a tale privilegio.

Lo standard SQL prevede due alternative per la semantica della revoca, selezionabili con le clausole `RESTRICT` e `CASCADE`.

7.3.1 `CASCADE`

Se la revoca di un privilegio P (o di un ruolo R) all'utente u è richiesta con l'opzione `CASCADE`, essa viene implementata come segue:

1. Si revoca P (o R) all'utente u, a meno che esso non abbia ricevuto P (o R) anche da altre fonti indipendenti.
2. Si esegue la revoca di P (o R) su tutti gli eventuali utenti $u_1, \ldots, u_n$ che hanno ricevuto P (o R) da u ma non da altre fonti indipendenti, e così via ricorsivamente. La revoca ricorsiva avviene anche quando u non perde P (o R) ma perde la GRANT OPTION, ad esempio perché ha ricevuto P (o R) anche da fonti indipendenti dal richiedente della revoca ma ha ricevuto la GRANT OPTION solo da quest'ultimo.

Nell'eseguire la revoca vengono anche cancellati gli eventuali oggetti dello schema creati grazie ai privilegi revocati.

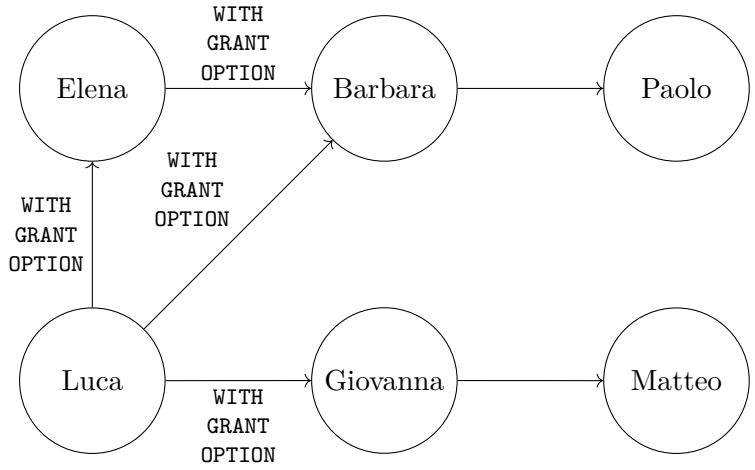
Al fine di chiarire la semantica di quest'operazione è utile considerare un esempio. Si consideri una base di dati in cui tutte le relazioni sono state create da un utente Luca (che è dunque il proprietario di ogni relazione) e sono poi stati concessi i seguenti privilegi:

```
-- luca:
GRANT SELECT ON Film TO barbara, giovanna WITH GRANT OPTION;
-- giovanna:
GRANT SELECT ON Film TO matteo;
-- luca:
GRANT ALL PRIVILEGES ON Video, Film TO elena WITH GRANT OPTION;
-- elena:
GRANT INSERT, SELECT ON Film TO barbara WITH GRANT OPTION;
-- barbara:
GRANT SELECT ON Film TO paolo;
```

Si supponga poi che Luca esegua la revoca

```
-- luca:
REVOKE SELECT ON Film FROM barbara, giovanna CASCADE;
```

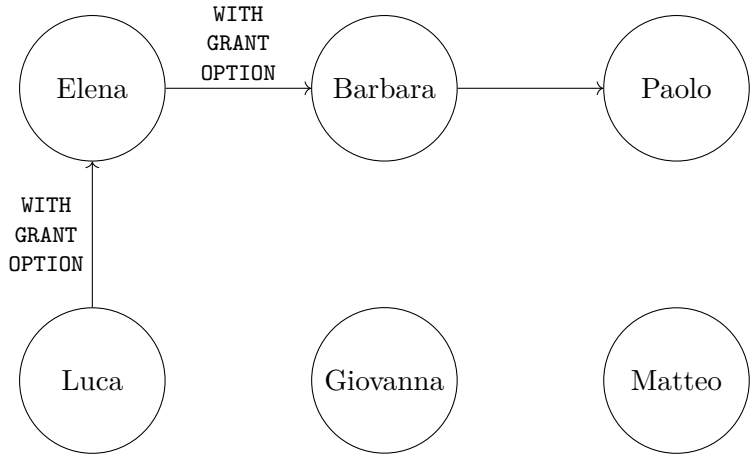
Per determinare da quali utenti viene revocato il privilegio SELECT su Film è utile analizzare il *grafo di autorizzazione* relativo a tale privilegio, che descrive quali utenti hanno concesso il privilegio a quali altri e indica se l'hanno fatto con la GRANT OPTION o meno:



Dal grafo si può determinare che:

- la revoca non fa perdere né il privilegio né la GRANT OPTION a Barbara, che li ha ricevuti anche da Elena, e di conseguenza Paolo mantiene il privilegio concesso da Barbara;
- Giovanna perde invece il privilegio, poiché l'aveva ricevuto solo dall'utente Luca che adesso lo sta revocando, e di conseguenza anche Matteo perde il privilegio, avendolo ricevuto solo da Giovanna.

Il grafo di autorizzazione ottenuto dopo la revoca è dunque:



7.3.2 `RESTRICT`

La revoca con `CASCADE` può cancellare numerose autorizzazioni e anche degli oggetti dello schema. Per evitare ciò si può usare invece l'opzione `RESTRICT`, con la quale l'esecuzione

di un comando `REVOKE` non viene concessa se essa comporterebbe la revoca di altri privilegi/ruoli oltre a quelli direttamente specificato nel comando e/o la cancellazione di oggetti dello schema. Ad esempio, nello scenario precedente il comando

```
-- luca:
REVOKE SELECT ON Film FROM barbara, giovanna RESTRICT;
```

non sarebbe eseguito, dato che comporterebbe anche la revoca del privilegio di Matteo, il quale non è specificato esplicitamente nel comando.

`RESTRICT` è utile soprattutto per evitare cancellazioni inaspettate quando si revocano privilegi concessi con la GRANT OPTION (o ruoli concessi con l'ADMIN OPTION), poiché tipicamente non si sa come la GRANT OPTION e i privilegi concessi grazie a essa si siano "diffusi" nel grafo di autorizzazione.