Capitolo 9

Protezione dei dati

L'impiego sempre più diffuso delle basi di dati a supporto di funzioni di tipo gestionale ha rappresentato una condizione necessaria per la crescente automazione dei sistemi informativi, ma ha reso estremamente critico il problema della protezione dei dati da essi gestiti. In un ambiente di basi di dati, un danno al patrimonio informativo si ripercuote non solo sul singolo utente o sulla singola applicazione, ma investe l'intero sistema. In molte imprese ed organizzazioni, le informazioni sono così importanti che la loro, anche parziale, corruzione o distruzione può portare alla paralisi dell'intera struttura. La rapida diffusione di Internet ha ulteriormente acuito il problema della protezione, in quanto le informazioni residenti in una base di dati sono oggi potenzialmente accessibili da una vastissima comunità di utenti sparsa in tutto il mondo.

Solitamente, in una stessa base di dati sono memorizzate informazioni di differente importanza e sensibilità. Inoltre, i dati memorizzati sono condivisi tra più utenti con diverse responsabilità e privilegi. È indispensabile pertanto disporre di tecniche, strumenti e procedure di sicurezza volti sia a garantire l'affidabilità delle elaborazioni sia a proteggere le informazioni ed i programmi da intrusioni, modifiche, furti e divulgazioni non autorizzate. In effetti, un DBMS ed i suoi *meccanismi di sicurezza* devono rispondere ad una duplice esigenza: da un lato consentire la condivisione dei dati, dall'altro far sì che questa condivisione sia *selettiva* ed opportunamente regolamentata secondo le politiche dell'azienda od organizzazione.

Gli obiettivi della protezione dei dati sono principalmente tre: *riservatezza*, ovvero la protezione delle informazioni da letture non autorizzate (tale proprietà prende il nome di *privacy* quando si riferisce a dati di natura personale); *integrità*, ovvero la protezione dei dati da modifiche o cancellazioni non autorizzate; *disponibilità*, ovvero la garanzia che non si verifichino casi in cui ad utenti legittimi venga negato l'accesso ai dati (negazione di servizio). Tali obiettivi coesistono nelle esigenze di qualsiasi sistema. Tuttavia, l'importanza assegnata loro varia sensibilmente a seconda del sistema considerato. Ad esempio, l'aspetto della riservatezza è particolarmente rilevante in ambienti come quello militare, in cui sono gestiti dati molto delicati che, se scoperti, potrebbero danneggiare la sicurezza nazionale. In ambienti di tipo commerciale e aziendale risulta invece preponderante l'aspetto dell'integrità, poiché dalla correttezza dei dati manipolati dipende spesso tutta

l'attività dell'organizzazione che utilizza la base di dati stessa.

Esistono poi ulteriori proprietà di sicurezza, particolarmente rilevanti nel caso di accessi tramite Internet. Tra queste ricordiamo: l'*autenticità*, ovvero la garanzia da parte del ricevente dell'autenticità della fonte che ha generato i dati; la *completezza*, cioè la garanzia di aver ricevuto tutti i dati per cui si ha l'autorizzazione; l'*auto-protezione*, cioè il fatto che un utente possa specificare quale informazione non vuole ricevere.

Garantire la protezione delle informazioni contenute in una base di dati è un obiettivo che coinvolge non solo il DBMS ma anche il sistema operativo, l'hardware e le reti di comunicazione. In questo capitolo, concentreremo principalmente la nostra attenzione sui meccanismi offerti da un DBMS per garantire le varie proprietà di sicurezza.

In ambito basi di dati, la riservatezza delle informazioni è in primo luogo garantita dal *meccanismo di controllo dell'accesso*, un modulo del DBMS che intercetta ogni richiesta di lettura o scrittura e consente solo quelle per cui esistono le necessarie autorizzazioni. L'integrità è invece assicurata da un insieme di meccanismi. Il meccanismo di controllo dell'accesso, in analogia a quanto avviene per le operazioni di lettura, intercetta ogni richiesta di aggiornamento di dati consentendo solo quelle per cui esistono le necessarie autorizzazioni. Inoltre, una certa forma d'integrità è garantita anche dal meccanismo di verifica dei vincoli di integrità, che assicura che i dati inseriti siano corretti rispetto ai vincoli specificati, e dal gestore della concorrenza, che assicura la correttezza dei dati anche in presenza di accessi concorrenti da parte di più transazioni. Concorrono a garantire la riservatezza e l'integrità delle informazioni anche quei meccanismi, non di stretta competenza del DBMS, che assicurano tali proprietà durante la trasmissione in rete. In particolare, l'uso di tecniche di cifratura assicura che i dati trasmessi sulla rete di comunicazione possano essere decifrati solo da utenti autorizzati, mentre le tecniche di firma digitale sono usate per assicurare l'autenticità dei dati e la verifica della loro integrità. Infine, la disponibilità è in una certa misura garantita dal gestore del ripristino, che assicura la disponibilità dei dati in presenza di malfunzionamenti hardware e software. La disponibilità dei dati può essere ulteriormente migliorata dall'uso di tecniche per il rilevamento delle intrusioni in grado di determinare sequenze di accessi inusuali, e che quindi potrebbero essere indice di un attacco al sistema, quali ad esempio un numero di richieste eccessivo effettuato per sovraccaricare il sistema e degradarne le prestazioni.

Oltre ai meccanismi sopra descritti, concorre alla protezione dei dati anche il meccanismo di autenticazione di cui ogni DBMS è dotato. Tale meccanismo verifica l'identità dell'utente che si connette al sistema e la sua autorizzazione all'accesso. I meccanismi di autenticazione adottati comunemente sono basati sull'uso di login e password. Meccanismi più sofisticati, oggi utilizzati solo in contesti particolari, sono basati sulla rilevazione di dati biometrici, quali impronte digitali, riconoscimento della voce, scansione della retina, ecc.

In questo capitolo ci concentreremo principalmente sui meccanismi di sicurezza propri dei DBMS. In particolare, ci focalizzeremo sul controllo dell'accesso. Uti-

lizzeremo, ove possibile, l'esempio della videoteca per illustrare i vari concetti; utilizzeremo altri domini applicativi ove questo sarà necessario. Rimandiamo il lettore al Capitolo 8 per la gestione del ripristino ed il controllo della concorrenza, ed al Capitolo 3 per la specifica dei vincoli d'integrità.

9.1 Controllo dell'accesso: concetti fondamentali

Il controllo dell'accesso regola le operazioni che è possibile compiere sui dati e le risorse in una base di dati. Lo scopo è quello di limitare e controllare le operazioni che gli utenti, già riconosciuti dal meccanismo di autenticazione, effettuano, prevenendo azioni accidentali o deliberate che potrebbero compromettere la correttezza e la sicurezza dei dati. Il meccanismo di controllo dell'accesso riveste quindi un ruolo fondamentale per la protezione dei dati gestiti da un DBMS, basti ricordare che la maggior parte delle violazioni di sicurezza avvengono di solito ad opera di utenti legittimi.

Indipendentemente dal contesto in cui si applica, nel controllo dell'accesso distinguiamo tre entità fondamentali: gli *oggetti*, cioè le risorse a cui vogliamo garantire protezione; i *soggetti*, ovvero le entità "attive" che richiedono di poter accedere agli oggetti; i *privilegi*, che determinano le operazioni che i soggetti possono effettuare sugli oggetti. I soggetti possono essere ulteriormente classificati in: *utenti*, cioè singoli individui; *gruppi*, ovvero insiemi di utenti; *ruoli*, ovvero funzioni aziendali a cui assegnare un insieme di privilegi per lo svolgimento delle loro mansioni; *processi*, cioè programmi in esecuzione per conto di utenti. Naturalmente, la tipologia di soggetti, oggetti e privilegi dipende dal contesto da proteggere. In ambito basi di dati relazionali, esempi di oggetti sono tabelle, singole tuple o colonne di una tabella, viste, mentre esempi di privilegi sono quelli che denotano operazioni eseguibili tramite comandi SQL (ad esempio, `SELECT`, `INSERT`, `UPDATE`, ecc.).

Le componenti principali di un sistema per il controllo dell'accesso sono illustrate nella Figura 9.1. Innanzitutto, la concessione o meno di un determinato accesso deve rispecchiare le *politiche di sicurezza* dell'organizzazione, ovvero le regole ed i principi secondo cui l'organizzazione vuole che siano protette le proprie informazioni. Le politiche di sicurezza rappresentano quindi un insieme di direttive ad alto livello che esprimono le scelte compiute da un'organizzazione in merito alla protezione dei propri dati. Possono essere considerate come i requisiti per lo sviluppo di un sistema di controllo dell'accesso e dipendono da vari fattori, quali ad esempio l'ambito in cui l'organizzazione opera, la sua natura, la legislazione vigente, le esigenze degli utenti o i regolamenti interni. Esempi di politiche di sicurezza sono: "solo l'amministratore della base di dati può concedere o revocare privilegi di accesso" o "le valutazioni psicologiche di un impiegato possono essere viste solo dal suo responsabile". Le politiche, per poter essere utilizzate ai fini del controllo dell'accesso, devono essere tradotte in un insieme di *autorizzazioni* che stabiliscono gli specifici diritti che i vari soggetti abilitati ad accedere al sistema possono esercitare sugli oggetti, in conformità con le politiche di sicurezza adottate. Le autorizzazioni, nel loro formato base, possono essere rappresentate

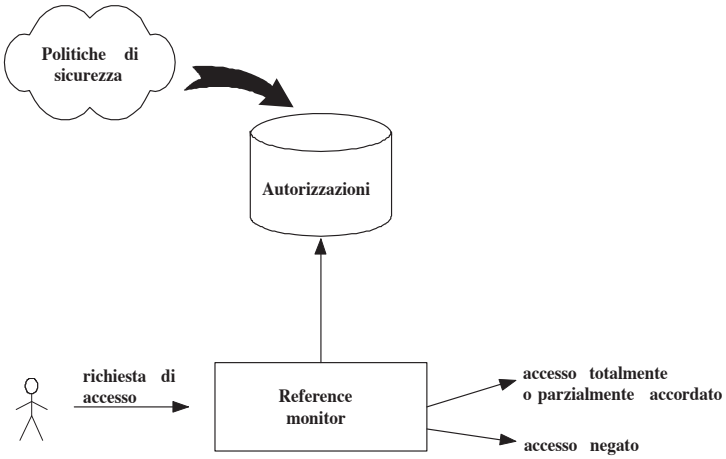

Figura 9.1: Controllo dell'accesso

mediante una tupla (s, o, p), dove s è il soggetto a cui l'autorizzazione è concessa, o è l'oggetto sui cui è concessa l'autorizzazione e p è il privilegio che s può esercitare su o. Ad esempio, se consideriamo la politica illustrata precedentemente relativa alle valutazioni psicologiche e supponiamo che Mario Rossi sia il responsabile di Giovanni Bianchi, la tupla `(mario rossi,valutazionePsicologica(giovanni bianchi),lettura)` è un esempio di autorizzazione conforme alla politica, dove `valutazionePsicologica(giovanni bianchi)` indica il documento contenente la valutazione psicologica dell'impiegato Giovanni Bianchi. Le autorizzazioni devono essere memorizzate nel sistema per poter essere utilizzate ai fini del controllo dell'accesso. Solitamente, viene utilizzato per la loro rappresentazione lo stesso modello dei dati utilizzato per la rappresentazione degli oggetti. Ad esempio, in un DBMS relazionale le autorizzazioni sono memorizzate in cataloghi (vedi Capitolo 7).

Come illustrato nella Figura 9.1, il controllo dell'accesso è effettuato mediante il *meccanismo di controllo dell'accesso*, detto anche *reference monitor*, il cui compito è quello di intercettare ogni comando inviato al DBMS e stabilire, tramite l'analisi delle autorizzazioni, se il soggetto richiedente può essere autorizzato (totalmente o parzialmente) o meno a compiere l'accesso richiesto. Ogni meccanismo di controllo dell'accesso è basato su un *modello di controllo dell'accesso*, che definisce la natura delle componenti base di un'autorizzazione (soggetti, oggetti e privilegi), le loro inter-relazioni e tutte le funzionalità per la verifica e l'amministrazione delle autorizzazioni.

Nei paragrafi successivi illustreremo le più rilevanti politiche di sicurezza.

9.2 Politiche di sicurezza

Rispetto al controllo dell'accesso, le politiche di sicurezza possono essere suddivise in due classi fondamentali: *politiche per il controllo dell'accesso* e *politiche per l'amministrazione del controllo dell'accesso*, che verranno discusse nei paragrafi successivi.

9.2.1 Politiche per il controllo dell'accesso

Le politiche per il controllo dell'accesso definiscono i criteri secondo cui concedere o rifiutare l'accesso alle informazioni memorizzate in una base di dati. Nei paragrafi successivi illustreremo le dimensioni in base a cui possono essere classificate.

9.2.1.1 Limitazione degli accessi

Un primo aspetto che le politiche di controllo dell'accesso devono regolamentare è relativo alla quantità di informazione a cui ciascun soggetto può accedere. Rispetto a questa dimensione, le principali opzioni sono:

- **Need to know – Principio del minimo privilegio**. È un principio molto restrittivo che permette ad ogni soggetto l'accesso solo a quei dati strettamente necessari per eseguire le proprie attività.
- **Maximized sharing – Principio della massima condivisione**. Lo scopo di questo principio è di consentire ai soggetti il massimo accesso alle informazioni nella base di dati, mantentendo comunque alcune informazioni riservate.

Entrambi i principi presentano vantaggi e svantaggi e la scelta di quale adottare dipende fortemente dal dominio considerato. Il principio del minimo privilegio offre ottime garanzie di protezione ed è pertanto adatto a contesti con forti esigenze di sicurezza. Il principale svantaggio è che può portare ad un sistema eccessivamente protetto, negando anche accessi che non comprometterebbero la sicurezza del sistema. Il principio della massima condivisione, invece, ha il vantaggio di soddisfare il massimo numero possibile di richieste di accesso; pertanto, viene solitamente utilizzato in ambienti, quali quelli accademici, in cui esiste una certa fiducia tra gli utenti che accedono alle informazioni ed in cui non è sentita una forte esigenza di protezione.

I sistemi per il controllo dell'accesso possono essere inoltre distinti in sistemi aperti e sistemi chiusi. In un sistema chiuso l'accesso è permesso *solo se* esplicitamente autorizzato, mentre in un sistema aperto l'accesso è permesso *a meno che* non sia esplicitamente negato. In un sistema chiuso, quindi, le autorizzazioni indicano per ogni soggetto i privilegi che egli *può* esercitare sugli oggetti del sistema. Tali privilegi sono i soli che verranno accordati dal meccanismo di controllo. Al contrario, in un sistema aperto le autorizzazioni stabiliscono, per ogni soggetto, i privilegi che egli *non può* esercitare sugli oggetti del sistema. Tali privilegi sono i

soli che gli saranno negati. Un sistema chiuso ben si adatta, quindi, a realizzare il principio del minimo privilegio, mentre un sistema aperto è adatto ad implementare il principio della massima condivisione. Un sistema chiuso offre maggiori garanzie di sicurezza rispetto ad un sistema aperto, in quanto un'autorizzazione inavvertitamente cancellata o non inserita restringe ulteriormente l'accesso, mentre in un sistema aperto la stessa operazione permette accessi non autorizzati. Per questa ragione la maggior parte dei DBMS commerciali si comporta come un sistema chiuso.

9.2.1.2 Controllo dell'accesso

Un altro aspetto regolato dalle politiche per il controllo dell'accesso riguarda i criteri in base a cui i soggetti possono accedere agli oggetti nel sistema e se e come i diritti d'accesso possono venire trasmessi a terzi. Rispetto a queste dimensioni, le politiche possono essere classificate in tre categorie: *politiche discrezionali*, *politiche mandatorie* e *politiche basate su ruoli*.

Politiche discrezionali. Tali politiche attuano il controllo dell'accesso sulla base dell'identità del soggetto richiedente. In particolare, tali politiche sono tradotte in un insieme di autorizzazioni che stabiliscono esplicitamente, per ogni soggetto, i privilegi che questo può esercitare sugli oggetti del sistema. Il meccanismo di controllo esamina le richieste di accesso accordando solo quelle che sono concesse da un'autorizzazione. Queste politiche vengono dette discrezionali, in quanto permettono agli utenti di concedere o revocare dei diritti di accesso sugli oggetti, a loro discrezione.

Le politiche discrezionali hanno il vantaggio di essere estremamente flessibili e, quindi, adatte a numerosi contesti applicativi, in quanto possono modellare svariati requisiti di sicurezza, configurando opportunamente l'insieme di autorizzazioni. Il loro principale svantaggio è che non forniscono alcun meccanismo di controllo sul flusso di informazioni nel sistema, in quanto non impongono alcuna restrizione su come l'informazione possa essere trasmessa da un soggetto autorizzato ad un altro non autorizzato ad accedervi. Questo rende i meccanismi discrezionali vulnerabili rispetto ad attacchi, quali quelli perpetrati tramite *cavalli di Troia*. I cavalli di Troia sono programmi che apparentemente svolgono un compito utile (ad esempio stampare una lista di titoli di film) ma che al loro interno contengono delle istruzioni nascoste che utilizzano fraudolentemente le autorizzazioni degli utenti che li eseguono per trasferire illegalmente informazioni non violando, al contempo, le limitazioni imposte dalla politica discrezionale. Il seguente esempio chiarisce meglio il concetto.

Esempio 9.1 Supponiamo che Anna e Paolo siano due dipendenti della videoteca e che Anna sia la responsabile di Paolo. Supponiamo che, in base alle politiche di sicurezza, Anna possa accedere sia in lettura sia in scrittura a tutte le relazioni della base di dati della videoteca, mentre a Paolo è proibito vedere i titoli dei film noleggiati dai clienti della videoteca. Supponiamo che Paolo regali ad Anna

un programma di utilità, ad esempio un programma per la gestione dell'agenda, in cui ha fraudolentemente inserito alcune istruzioni (il cavallo di Troia). Tali istruzioni: *(i)* creano una tabella `FilmClienti` che contiene, per ogni cliente, i titoli dei film che ha noleggiato; *(ii)* concedono a Paolo l'autorizzazione ad effettuare interrogazioni su di essa. Quando Anna esegue il programma, il meccanismo di controllo dell'accesso autorizzerà o meno le operazioni effettuate dal programma in base alle autorizzazioni possedute da Anna. In questo modo Paolo, senza violare i controlli della politica discrezionale, potrà accedere ad informazioni per cui non ha le necessarie autorizzazioni. □

Alcuni di questi limiti di sicurezza sono superati dalle politiche mandatorie.

Politiche mandatorie. Tali politiche regolano l'accesso ai dati mediante la definizione di *classi di sicurezza*, chiamate anche etichette, 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 è una misura della sensibilità dell'informazione che l'oggetto contiene: maggiore è la classe assegnata ad un oggetto più ingente è il danno derivante dal rilascio delle informazioni in esso contenute a soggetti non autorizzati. La classe di sicurezza assegnata ad un soggetto è, invece, una misura del grado di fiducia che si ha nel fatto che tale soggetto non commetta violazioni della politica ed, in particolare, non trasmetta ad altri informazioni riservate. Il controllo dell'accesso è regolato da una serie di *assiomi di sicurezza* che stabiliscono la relazione che deve intercorrere fra la classe di un soggetto e quella di un oggetto affinché al primo sia concesso di esercitare un privilegio sul secondo. La relazione che deve essere soddisfatta dipende dal tipo di privilegio considerato. In generale, gli assiomi sono volti ad evitare qualsiasi trasferimento d'informazione da un certo soggetto od oggetto verso soggetti od oggetti con classificazione minore o non comparabile.

Le politiche mandatorie sono applicate in ambienti, quali quello militare, dove ci sono forti esigenze di protezione ed è possibile classificare rigidamente soggetti ed oggetti del sistema. I sistemi che adottano politiche mandatorie sono spesso indicati come *sistemi multi-livello*.

La politiche mandatorie e discrezionali non sono però mutuamente esclusive, possono cioè essere applicate in combinazione. In questo caso, le politiche mandatorie non controllano più le richieste di accesso ma le autorizzazioni che vengono assegnate ad un soggetto, mentre alle politiche discrezionali è affidato il compito di controllare le richieste di accesso. Se una richiesta di accesso soddisfa il controllo discrezionale, cioè esiste per essa un'autorizzazione, allora soddisfa anche gli assiomi relativi al controllo mandatorio, per il controllo preventivo effettuato sulle autorizzazioni. Quando le due politiche sono utilizzate congiuntamente, le politiche discrezionali servono per restringere ulteriormente gli accessi consentiti dalle politiche mandatorie, per determinate classi di soggetti e/o oggetti.

Politiche basate su ruoli. In aggiunta alle politiche mandatorie e discrezionali, un terzo tipo di politiche, basate sul concetto di *ruolo*, è stato successivamente proposto. In base a tali politiche, i privilegi non sono direttamente assegnati agli

utenti ma sono mediati dal concetto di ruolo. Un ruolo rappresenta una funzione all'interno di un'azienda od organizzazione (sono esempi di ruoli per il dominio della videoteca `direttoreVideoteca`, `commesso` e `cliente`). Le autorizzazioni non sono concesse ai singoli utenti ma ai ruoli identificati. Ogni utente è poi abilitato a ricoprire uno o più ruoli ed in questo modo acquisisce le autorizzazioni ad essi associate. Il concetto di ruolo semplifica notevolmente la gestione delle autorizzazioni e rende più diretta la modellazione delle politiche aziendali. Infatti, in un'azienda/organizzazione spesso i privilegi esercitabili non sono legati all'identità dei singoli ma piuttosto al ruolo che essi ricoprono all'interno della stessa. La semplificazione amministrativa è dovuta principalmente a due fattori: il primo è che un ruolo di solito raggruppa un cospicuo numero di privilegi. Quando un utente deve ricoprire la mansione associata ad un certo ruolo, invece di specificare per lui tutte le autorizzazioni associate al ruolo, basta abilitarlo a ricoprire quel ruolo e questo comporta automaticamente l'acquisizione dei privilegi associati al ruolo stesso. La seconda semplificazione è dovuta al fatto che i ruoli rappresentano una componente del sistema più stabile rispetto agli utenti, che possono cambiare frequentemente e/o cambiare le loro mansioni. Ad esempio, un cambio di mansioni in un sistema che supporta i ruoli viene gestito molto facilmente: basta revocare all'utente interessato dal cambio di mansione l'autorizzazione a ricoprire i ruoli associati alla vecchia mansione ed abilitarlo a ricoprire i ruoli corrispondenti alla nuova.

Un'altra caratteristica importante delle politiche basate su ruoli è che, configurando opportunamente il sistema dei ruoli, possono essere usate per realizzare sia politiche discrezionali sia mandatorie, e questo accresce notevolmente il loro campo di impiego.

9.2.2 Politiche per l'amministrazione del controllo dell'accesso

Un altro importante aspetto regolato dalle politiche di sicurezza è chi può concedere e revocare privilegi. Questo aspetto è maggiormente rilevante per i meccanismi di controllo dell'accesso discrezionali e basati su ruoli, mentre nei sistemi mandatori di solito la politica di amministrazione è molto semplice in quanto non vi sono autorizzazioni esplicite e l'amministratore della sicurezza è di solito l'unico che può modificare le classi di sicurezza di soggetti ed oggetti. Le politiche discrezionali e basate su ruoli permettono, invece, di adottare soluzioni più articolate. In particolare, le principali politiche di amministrazione che possono essere adottate sono:

- **Centralizzata**. Un unico autorizzatore (o gruppo), detto amministratore della sicurezza, può specificare e revocare le autorizzazioni.
- **Basata su ownership**. L'utente che crea un oggetto, detto *owner*, gestisce anche la concessione e la revoca di autorizzazioni sull'oggetto.
- **Decentralizzata**. In base a questa politica, l'owner o l'amministratore della sicurezza possono delegare ad altri la facoltà di specificare e revocare autoriz-

zazioni per determinati oggetti e modi di accesso. Molti DBMS commerciali adottano la politica basata su ownership con delega dell'amministrazione.

- **Cooperativa**. Autorizzazioni su particolari oggetti non possono essere concesse da un singolo utente, ma richiedono il consenso di più utenti. Questa politica di amministrazione è particolarmente adatta ad ambienti cooperativi e distribuiti.

9.3 Modelli di controllo dell'accesso

I modelli di controllo dell'accesso per basi di dati sono stati sviluppati a partire dai modelli definiti per la protezione di risorse in un sistema operativo. Tuttavia, i modelli usati nell'ambito di un sistema operativo non sono completamente adatti ad essere utilizzati in una base di dati. Le differenze fondamentali tra i due contesti sono innanzitutto dovute al fatto che il controllo dell'accesso in una base di dati deve avvenire a diversi livelli di granularità (ad esempio, nel caso di basi di dati relazionali, a livello di relazione, vista, tupla o singolo attributo). Inoltre, un modello di controllo dell'accesso in un sistema operativo protegge risorse reali (ad esempio stampanti, supporti di memorizzazione, ecc.), mentre in una base di dati le risorse da proteggere possono essere strutture completamente logiche (ad esempio viste), alcune delle quali possono corrispondere allo stesso oggetto fisico.

Nei paragrafi successivi descriveremo i più significativi modelli di controllo dell'accesso.

9.3.1 Modello a matrice di accesso

Tra i modelli sviluppati nell'ambito dei sistemi operativi, quello che ha maggiormente influenzato i modelli per il controllo dell'accesso per basi di dati è il modello di Lampson, successivamente esteso da Graham e Denning e formalizzato da Harrison, Ruzzo ed Ullman (quest'ultimo è conosciuto come *modello HRU* dalle iniziali dei cognomi dei suoi ideatori). Tale modello è il modello concettuale di riferimento per rappresentare autorizzazioni in sistemi che adottano politiche discrezionali. Il nome del modello deriva dal fatto che lo stato corrente del sistema rispetto alle autorizzazioni è descritto mediante una matrice. Più precisamente, nel modello a matrice di accesso lo stato del sistema è descritto dalla tripla (S,O,M), dove: S è l'insieme dei soggetti, cioè delle entità che richiedono gli accessi, O è l'insieme degli oggetti, cioè delle entità su cui viene richiesto l'accesso, ed M è la *matrice di accesso*, in cui ogni riga corrisponde ad un soggetto, ogni colonna ad un oggetto e l'elemento $M[i, j]$ contiene la lista dei privilegi esercitabili dal soggetto s_i sull'oggetto o_j, scelti tra l'insieme di privilegi che il modello mette a disposizione.

Il meccanismo di controllo dell'accesso autorizza solo gli accessi per cui esiste il relativo privilegio nella matrice di accesso.

Esempio 9.2 La Figura 9.2 illustra un esempio di matrice di accesso, per la protezione di risorse in un sistema operativo. Ad esempio, in base alla Figura

	marco.doc	**edit.exe**	**giochi.dir**
Marco	{read,write}	{execute}	{execute}
Anna	-	{execute}	{execute,read,write}

Figura 9.2: Matrice di accesso

9.2 sia Marco sia Anna possono eseguire i programmi contenuti nella directory `giochi.dir`, ma solo Anna può leggere e modificare i file in essa contenuti. Viceversa, Marco è il solo che può leggere e modificare il file `marco.doc`. □

Notiamo come la matrice di accesso sia solo un modello concettuale, non adatto per un'effettiva implementazione. Infatti, in molte situazioni la matrice risulterebbe troppo sparsa e di dimensioni troppo elevate, in quanto ogni soggetto ha solitamente accesso solo ad una piccola porzione degli oggetti del sistema. Le soluzioni utilizzate nella pratica per implementare la matrice di accesso sono essenzialmente due:

- **Access Control List (ACL)**. In questo caso la matrice viene implementata mediante un insieme di liste associate ad ogni oggetto del sistema. L'ACL associata ad un oggetto contiene un elemento per ogni soggetto che può esercitare un qualche privilegio sull'oggetto in questione. Tale elemento contiene la lista dei privilegi che il soggetto può esercitare sull'oggetto.
- **Capability List**. È un metodo complementare a quello delle ACL ed implementa la matrice di accesso mediante liste associate ai soggetti. La capability list associata ad un soggetto contiene un elemento per ogni oggetto su cui il soggetto può esercitare un qualche privilegio. Tale elemento contiene la lista dei privilegi che il soggetto ha su quel determinato oggetto.

L'approccio basato su ACL è quello utilizzato dai moderni sistemi operativi, mentre le capability list sono più adatte a sistemi distribuiti dove un soggetto potrebbe autenticarsi una sola volta presso un host, acquisire le sue capability ed utilizzarle per ottenere l'accesso presso tutti gli host che compongono il sistema distribuito. Naturalmente, in questo caso bisogna predisporre meccanismi che evitino che i soggetti possano coniare capability false o utilizzare capability non valide, quali ad esempio quelle relative ad un privilegio revocato.

9.3.2 Modello di Bell e LaPadula

Il modello sviluppato da Bell e LaPadula rappresenta il modello di riferimento per la maggior parte dei sistemi che adottano una politica mandatoria. Nel modello di Bell e LaPadula, i soggetti sono sia utenti sia processi. I privilegi previsti dal modello sono: `read`, che consente ad un soggetto di leggere ma non di modificare le informazioni contenute in un oggetto; `append`, che consente ad un soggetto di

modificare un oggetto ma non di leggere le informazioni in esso contenute; `write`, che consente sia la modifica sia la lettura di un oggetto; `execute`, che consente ad un soggetto di eseguire un oggetto (ad esempio un programma applicativo). Nel seguito considereremo solo i privilegi `read`, `write` ed `append` in quanto strettamente pertinenti alla gestione dati.

Essendo il modello di Bell e LaPadula un modello mandatorio, soggetti ed oggetti vengono classificati mediante l'assegnazione di una *classe di sicurezza*. Una classe di sicurezza è costituita da due componenti: un *livello di sicurezza* ed un *insieme di categorie*. Il livello di sicurezza è un elemento, scelto da un insieme totalmente ordinato. Esempi di livelli comunemente usati sono: `Top Secret` (`TS`), `Secret` (`S`), `Confidential` (`C`) e `Unclassified` (`U`), ordinati nel modo seguente: `TS` $>$ `S` $>$ `C` $>$ `U`.

L'insieme di categorie è un insieme non ordinato di elementi, che può essere anche vuoto. Gli elementi di questo insieme dipendono dal tipo di ambiente considerato e denotano aree di competenza all'interno dell'organizzazione. Ad esempio, se consideriamo l'ambito militare, possibili categorie sono: `Army`, `Navy`, `Air Force` e `Nuclear`, mentre in ambito commerciale esempi di possibili categorie sono: `Finanza`, `Vendite` e `Ricerca e Sviluppo`. Un file contenente informazioni finanziarie riservate potrebbe, ad esempio, essere classificato `(Confidential,` {`Finanza`}`)`, mentre un generale di corpo d'armata potrebbe avere come classe di sicurezza `(Top Secret,`{`Navy, Nuclear`}`)`.

L'insieme delle classi di sicurezza è parzialmente ordinato da una relazione di dominanza formalmente definita come segue.

Definizione 9.1 (Relazione di dominanza) *Una classe di sicurezza* $cs_1 = (L_1, Cat_1)$ *domina una classe di sicurezza* $cs_2 = (L_2, Cat_2)$*, (denotato con* $c_1 \geq c_2$*), se entrambe le seguenti condizioni sono verificate: (i) il livello di sicurezza di* cs_1 *è maggiore o uguale al livello di sicurezza di* cs_2 *(cioè* $L_1 \geq L_2$*); (ii) l'insieme di categorie di* cs_1 *include l'insieme di categorie di* cs_2 *(cioè* $Cat_1 \supseteq Cat_2$*).* $\diamond$

Se $L_1 > L_2$ e $Cat_1 \supset Cat_2$, allora cs_1 *domina strettamente* cs_2 ($cs_1 > cs_2$). Infine, due classi di sicurezza cs_1 e cs_2 sono *incomparabili* ($cs_1 <> cs_2$) se né $cs_1 \geq cs_2$ né $cs_2 \geq cs_1$ valgono.

Esempio 9.3 Consideriamo le seguenti classi di sicurezza:

$\texttt{cs}_1$ = `(TS,{Nuclear,Army})`
$\texttt{cs}_2$ = `(TS,{Nuclear})`
$\texttt{cs}_3$ = `(C,{Army})`

$\texttt{cs}_1 \geq \texttt{cs}_2$ in quanto $\texttt{cs}_1$ ha lo stesso livello di sicurezza di $\texttt{cs}_2$ ma un sopra-insieme delle sue categorie; $\texttt{cs}_1 > \texttt{cs}_3$, in quanto il livello di sicurezza `TS` è strettamente maggiore di `C` e {`Army`} è un sottoinsieme proprio di {`Nuclear,Army`}. Infine, $\texttt{cs}_2 <> \texttt{cs}_3$; infatti, $\texttt{cs}_2 \not\geq \texttt{cs}_3$, in quanto {`Nuclear`}$\not\supseteq$ {`Army`} e $\texttt{cs}_3 \not\geq \texttt{cs}_2$, in quanto `TS` $>$ `C`. $\square$

Lo stato del sistema è descritto dalla coppia $(A, \mathcal{L})$,[1] dove:

- A è *l'insieme degli accessi correnti*, ossia degli accessi correntemente in esecuzione; A è composto da triple della forma (s, o, p) che indicano che il soggetto s sta correntemente esercitando il privilegio p sull'oggetto o.
- $\mathcal{L}$ è *la funzione di livello* che associa ad ogni elemento del sistema la sua classe di sicurezza, formalmente: $\mathcal{L} : O \cup S \rightarrow \mathcal{CS}$, dove O ed S sono, rispettivamente, l'insieme degli oggetti e dei soggetti, e $\mathcal{CS}$ è l'insieme delle classi di sicurezza.

Ogni modifica di stato è causata da una *richiesta*. Una richiesta può essere ad esempio una richiesta di accesso o una richiesta di modifica delle classi di sicurezza di soggetti ed oggetti. La risposta del sistema è chiamata *decisione*. Data una richiesta ed uno stato corrente, la decisione ed il nuovo stato sono determinati in base a degli *assiomi di sicurezza*. Tali assiomi stabiliscono le condizioni che devono essere soddisfatte nel sistema per effettuare una transizione di stato. Se le richieste sono soddisfatte solo se verificano gli assiomi, allora il sistema è *sicuro*.

Per quanto riguarda le richieste di accesso, il primo assioma del modello di Bell e LaPadula è il seguente:[2]

Proprietà di sicurezza semplice. Uno stato $(A, \mathcal{L})$ soddisfa la proprietà di sicurezza semplice se per ogni elemento $(s, o, p) \in A$ tale che p=`read` oppure p=`write`: $\mathcal{L}(s) \geq \mathcal{L}(o)$.

La proprietà di sicurezza semplice, conosciuta anche come *proprietà del no read-up*, assicura che i soggetti abbiano accesso diretto solo alle informazioni per cui hanno la necessaria classificazione, prevenendo letture di oggetti con classe di sicurezza maggiore o incomparabile.

Esempio 9.4 Un soggetto con classe di sicurezza `(C,{Army})` non può leggere dati con classi di accesso `(C,{Army,Air Force})` o `(U,{Air Force})`, mentre può leggere oggetti con classe di sicurezza `(U,{Army})`. □

La proprietà di sicurezza semplice non evita però che una combinazione di accessi possa compromettere la sicurezza dei dati. Ad esempio un soggetto con classe di sicurezza `(TS,`$\emptyset$`)` potrebbe estrarre informazioni da un oggetto con classe di sicurezza `(TS,`$\emptyset$`)` ed inserirle in uno con classe di sicurezza `(U,`$\emptyset$`)`, rendendole quindi accessibili a soggetti con classe di sicurezza inferiore, senza violare la proprietà di sicurezza semplice, che è stata pensata principalmente per regolamentare le operazioni di lettura. Per evitare tali situazioni, è stato definito un ulteriore assioma, riferito specificatamente alle operazioni di scrittura.

[1] Nella formulazione originaria di Bell e LaPadula, lo stato del sistema è descritto da una quadrupla che contiene, in aggiunta alle due componenti descritte, anche la matrice di accesso e la gerarchia degli oggetti. Tali componenti vengono omesse in questa sede in quanto non rilevanti per la successiva trattazione.

[2] Nel seguito, i termini assioma e proprietà saranno usati come sinonimi.

Proprietà $*$. Uno stato $(A, \mathcal{L})$ soddisfa la proprietà $*$, se per ogni elemento $(s, o, p) \in A$ tale che p=`append` o p=`write`: $\mathcal{L}(s) \leq \mathcal{L}(o)$.

La proprietà $*$ ($*$ *property*), conosciuta anche come *proprietà del no write-down*, è definita per prevenire il flusso d'informazioni dovute a scritture verso classi di sicurezza minori o non comparabili.

Esempio 9.5 Un soggetto con classe di sicurezza `(C,{Army,Nuclear})` non può scrivere informazioni in oggetti con classe di sicurezza `(U,{Army,Nuclear})` perché tali oggetti sono accessibili a soggetti che non possono legggere oggetti con classificazione `(C,{Army,Nuclear})`. □

Rispetto al controllo dell'accesso, un sistema si dice sicuro se ogni elemento aggiunto all'insieme degli accessi correnti verifica la proprietà di sicurezza semplice e la proprietà $*$. Notiamo come questo implichi che, per il privilegio `write`, devono essere applicati entrambi gli assiomi. Un soggetto s può quindi esercitare il privilegio `write` sull'oggetto o solo se $\mathcal{L}(s) = \mathcal{L}(o)$.

L'applicazione delle due proprietà precedentemente enunciate può comportare un'eccessiva rigidità del sistema, come evidenziato dal seguente esempio.

Esempio 9.6 Supponiamo che un generale abbia classe di sicurezza `(TS,{Army, Nuclear})`, mentre un suo colonnello abbia classe di sicurezza `(C,{Army})`. Il colonnello può potenzialmente comunicare con il generale, in quanto gli è concessa la scrittura in modalità append di oggetti con classe di sicurezza maggiore, mentre non è possibile per il generale comunicare con il suo colonnello in quanto, in base alla proprietà $*$, il generale non può scrivere in oggetti con classe di sicurezza minore della sua e quindi accessibili al colonnello. □

Per ovviare agli inconvenienti discussi nell'Esempio 9.6, gli utenti possono connettersi al sistema con una qualsiasi classe di sicurezza dominata da quella a loro assegnata. Operativamente, quando un utente si connette al sistema con una certa classe di sicurezza, viene considerato dal sistema come un soggetto con classe di sicurezza pari a quella con cui si è connesso. Il generale dell'esempio precedente può connettersi al sistema con classe di sicurezza `(C,{Army})` e comunicare quindi con il suo colonnello.

In effetti, la precedente definizione di sistema sicuro non garantisce la totale sicurezza del sistema, come evidenziato da Mc Lean in [McL90]. Ad esempio, consideriamo un sistema in cui, quando un soggetto s richiede un qualsiasi tipo di accesso su un oggetto o, a tutti gli oggetti ed i soggetti del sistema è assegnata la classe di sicurezza più bassa e l'accesso è consentito. Tale sistema soddisfa la definizione di sistema sicuro enunciata in precedenza, in quanto non viola né la proprietà di sicurezza semplice né la proprietà $*$, anche se presenta ovvi problemi di sicurezza, in quanto dopo la prima richiesta di accesso tutti possono accedere a tutto. In effetti, il modello di Bell e LaPadula garantisce buoni requisiti di sicurezza solo in presenza di una *classificazione statica* di soggetti ed oggetti in classi di sicurezza (tale principio è noto come *strong tranquillity principle*). Se tale principio non è

soddisfatto, la sicurezza o meno del sistema dipende dal modo con cui possono essere modificate le classi di sicurezza. Nella pratica, il principio di tranquillità sopra esposto è troppo restrittivo per essere applicato, in quanto ci possono essere situazioni in cui è necessaria una riclassificazione di soggetti ed oggetti rispetto alle classi di sicurezza. Per questo sono stati definiti principi meno restrittivi, che consentono la modifica delle classi di sicurezza sotto opportune condizioni. Viene inoltre introdotta la nozione di *soggetto fidato* (trusted), cioè un soggetto che può violare alcune delle restrizioni imposte dal modello. Ad esempio, Oracle Label Security, la componente del DBMS Oracle che consente di proteggere i dati secondo politiche mandatorie, prevede i seguenti privilegi speciali: `READ`, che consente di evitare i controlli in lettura degli assiomi mandatori; `FULL`, che consente di evitare sia i controlli in lettura sia in scrittura degli assiomi mandatori; `WRITEDOWN`, che consente di diminuire il livello di sicurezza di una classe di sicurezza; `WRITEUP`, che consente di aumentare il livello di sicurezza di una classe di sicurezza.

Notiamo, infine, come il modello di Bell e LaPadula permetta ad un soggetto di esercitare il privilegio `append` su oggetti con una classe di sicurezza superiore alla sua. Questo grado di libertà può essere utile in certi contesti e pericoloso in altri. Ad esempio, un documento a cui possono accedere in scrittura soggetti non autorizzati a leggerlo, può essere reso inconsistente da tali soggetti. Un approccio drastico al problema è quello di modificare il modello di Bell e LaPadula vietando le scritture verso classi di sicurezza maggiori o non comparabili, siano esse dovute all'esercizio del privilegio `append` o `write`.

9.3.3 Modello del System R

Uno dei modelli più rilevanti per sistemi che adottano politiche discrezionali è sicuramente il modello di controllo dell'accesso definito da Griffiths e Wade per il DBMS relazionale System R. Tale modello è stato uno dei primi che ha ricevuto un'effettiva implementazione, costituendo così la base per i meccanismi di controllo dell'accesso oggi presenti nei DBMS commerciali e per le funzionalità previste dallo standard SQL (vedi Paragrafo 9.4).

Essendo System R un DBMS relazionale, gli oggetti del modello sono relazioni di base e viste. Nel seguito utilizzeremo il termine relazione per far riferimento sia a relazioni di base sia a viste, quando una distinzione tra le due non sia necessaria. I privilegi previsti dal modello corrispondono alle operazioni effettuabili tramite comandi SQL (ad esempio, `SELECT`, `INSERT`, `DELETE`, `UPDATE`).

Il modello realizza una politica di tipo discrezionale adottando il paradigma di sistema chiuso: un accesso è concesso solo se esiste un'esplicita autorizzazione per esso. L'amministrazione dei privilegi è decentralizzata mediante ownership: l'utente che crea una relazione di base, riceve tutti i privilegi su di essa ed anche la possibilità di delegare ad altri tali privilegi. Per quanto riguarda le viste, invece, le regole sono leggermente diverse e verranno illustrate nel Paragrafo 9.3.3.6. La delega dei privilegi avviene mediante *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. Un utente può quindi concedere un privilegio su una determinata relazione solo se è il proprietario della relazione o ha ricevuto tale privilegio con grant option.

9.3.3.1 Comando `GRANT`

Le autorizzazioni sono concesse tramite il comando `GRANT`, la cui sintassi è la seguente:

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

dove:

- `<lista privilegi>` indica l'insieme dei privilegi concessi con il comando `GRANT`. Le parole chiave `ALL` o `ALL PRIVILEGES` (le due notazioni sono equivalenti) consentono di concedere con un solo comando tutti i privilegi previsti dal modello sulla relazione oggetto del comando.
- `<nome relazione>` indica il nome della relazione su cui sono concessi i privilegi.
- `<lista utenti>` indica l'insieme degli utenti a cui il comando si applica. La parola chiave `PUBLIC` consente di specificare le autorizzazioni implicate dal comando per tutti gli utenti del sistema.
- La clausola opzionale `WITH GRANT OPTION` consente la delega dell'amministrazione dei privilegi oggetto del comando, in aggiunta all'autorizzazione ad esercitarli. Se non è specificata, gli utenti che ricevono i privilegi non possono concederli ad altri utenti. Se invece è specificata, gli utenti che ricevono i privilegi possono sia esercitarli sia concederli a terzi, limitatamente alla relazione oggetto del comando.

I privilegi concessi con un comando `GRANT` si applicano ad intere relazioni, ad eccezione del privilegio `update` per cui è possibile specificare le colonne su cui si applica (racchiuse tra parantesi tonde e separate da virgole). È inoltre possibile concedere più privilegi su una stessa relazione con un unico comando (i privilegi devono essere separati tramite virgole). Analogamente, un unico comando `GRANT` può essere utilizzato per concedere privilegi sulla stessa relazione a più utenti.

Esempio 9.7 Consideriamo la base di dati relativa alla gestione della videoteca illustrata nel Capitolo 2 e supponiamo che tutte le relazioni di questa base di dati siano state create dall'utente Luca. I seguenti sono esempi di comandi `GRANT`:[3]

[3] La notazione `u:` indica che `u` è l'utente che richiede l'esecuzione del comando, detto anche *grantor*.

```
luca: GRANT update(telefono) ON Clienti TO marco;
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;
```

Il primo comando autorizza Marco a modificare l'attributo `telefono` delle tuple della relazione `Clienti`. Il secondo comando concede a Barbara e Giovanna la possibilità di effettuare interrogazioni sulla relazione `Film` e anche di concedere a terzi tale privilegio (che viene concesso da Giovanna a Matteo tramite il terzo comando). Il quarto comando concede ad Elena di esercitare tutti i privilegi previsti dal modello sulle relazioni `Video` e `Film`, unitamente alla possibilità di delegare ad altri tali privilegi, facendo diventare quindi Elena un ulteriore amministratore delle due relazioni, in aggiunta al loro proprietario Luca. Questo consente ad Elena di effettuare con successo l'ultimo comando `GRANT`, concedendo a Barbara il privilegio di inserire tuple ed effettuare interrogazioni sulla relazione `Film`. □

Dal momento che il modello del System R permette di concedere privilegi con grant option, è possibile che un utente riceva lo stesso privilegio sulla stessa relazione da utenti diversi. Ad esempio, con riferimento all'Esempio 9.7, Barbara riceve due volte il privilegio `select` sulla relazione `Film`, una volta da Luca con grant option e una volta da Elena senza grant option. Come vedremo nel Paragrafo 9.3.3.5, questa possibilità ha ripercussioni sull'operazione di revoca dei privilegi.

Inoltre, il fatto che i privilegi possano essere concessi con grant option fa sì che i privilegi che ogni utente possiede possano essere classificati in due categorie: *privilegi delegabili*, cioè quelli concessi all'utente con grant option, e *privilegi non delegabili*, cioè quelli che sono concessi senza grant option. Ai fini del controllo dell'accesso, entrambe le tipologie di privilegi sono rilevanti, mentre, per decidere l'esito di un comando `GRANT` dovremo prendere in considerazione solo i privilegi delegabili (vedi Paragrafo 9.3.3.4).

9.3.3.2 Comando `REVOKE`

I privilegi concessi possono essere successivamente revocati tramite il comando `REVOKE`, la cui sintassi è la seguente:

```
REVOKE {<lista privilegi> | ALL [PRIVILEGES]}
ON <nome relazione>
FROM {<lista utenti> | PUBLIC};
```

dove:

- `<lista privilegi>` indica l'insieme di privilegi oggetto del comando di revoca. Le parole chiave `ALL` (o `ALL PRIVILEGES`) indicano che tutti i privilegi

precedentemente concessi sulla relazione oggetto del comando dall'utente che esegue il comando sono revocati.

- `<nome relazione>` indica il nome della relazione su cui si vogliono revocare i privilegi.
- `<lista utenti>` indica l'insieme degli utenti a cui il comando si applica. La parola chiave `PUBLIC` consente di revocare i privilegi indicati nel comando a tutti gli utenti a cui erano stati precedentemente concessi sulla relazione oggetto del comando dall'utemte che esegue il comando `REVOKE`.

Un utente può revocare solo i privilegi da lui stesso concessi tramite un precedente comando `GRANT`. Analogamente al comando `GRANT`, è possibile revocare più privilegi con un unico comando `REVOKE`, ed un unico comando `REVOKE` può essere utilizzato per revocare gli stessi privilegi sulla stessa relazione ad utenti diversi. Chiaramente la revoca di un privilegio implica la revoca anche della grant option.

Esempio 9.8 Consideriamo i comandi `GRANT` dell'Esempio 9.7 ed i seguenti comandi `REVOKE` eseguiti da Luca:

```
REVOKE update, insert ON Video FROM elena;
REVOKE update ON Clienti FROM marco;
REVOKE select ON Film FROM giovanna;
```

Con il primo comando viene revocato ad Elena il privilegio di inserire e modificare tuple nella relazione `Video`. Il secondo comando revoca a Marco il diritto di effettuare modifiche sull'attributo `telefono` delle tuple della relazione `Clienti`. Infine, il terzo comando revoca a Giovanna la possibilità di effettuare interrogazioni sulla relazione `Film`. □

Il fatto che il modello di controllo dell'accesso del System R permetta di concedere privilegi con grant option implica che la revoca di un privilegio ad un utente non necessariamente comporta la perdita di quel privilegio da parte dell'utente in questione. Infatti, un utente può continuare ad esercitare un privilegio anche dopo un'operazione di revoca nel caso in cui lo abbia ricevuto da altre fonti *indipendenti* dal soggetto che effettua la revoca. L'esempio seguente chiarisce meglio il concetto.

Esempio 9.9 Consideriamo la sequenza di comandi `GRANT` dell'Esempio 9.7 ed il seguente comando `REVOKE`:

```
luca: REVOKE select ON Film FROM barbara, giovanna;
```

Dopo l'esecuzione del comando di revoca da parte di Luca, Giovanna non può più effettuare interrogazioni sulla relazione `Film`, mentre Barbara mantiene ancora tale privilegio, grazie all'autorizzazione concessale da Elena. Barbara non potrà però più concedere a terzi il privilegio `select` sulla relazione `Film`. □

9.3.3.3 I cataloghi `Sysauth` *e* `Syscolauth`

Il DBMS System R, in analogia a quanto accade anche oggi nei DBMS commerciali, utilizza una rappresentazione uniforme per dati ed autorizzazioni. Le informazioni sull'insieme di autorizzazioni correntemente presenti nel sistema sono memorizzate in due cataloghi di sistema (vedi Capitolo 7) di nome `Sysauth` e `Syscolauth`.

Il catalogo `Sysauth` contiene informazioni sui privilegi che ogni utente ha sulle relazioni della base di dati, mentre il catalogo `Syscolauth` serve per la gestione del privilegio `update`. Lo schema di `Sysauth` è costituito dai seguenti attributi:

- `utente`, è l'identificatore dell'utente a cui sono concessi i privilegi.
- `nomeRel`, indica il nome della relazione su cui sono concessi i privilegi.
- `tipo` $\in$ {`R`,`V`}, indica se l'oggetto dell'autorizzazione è una relazione di base (`tipo`='`R`') o una vista (`tipo`='`V`').
- `select`, indica se l'utente ha il privilegio di effettuare interrogazioni sulla relazione considerata. Tale colonna contiene un *timestamp*, che denota il tempo in cui il privilegio è stato concesso; il valore 0 indica che il relativo privilegio non è stato concesso. Esistono colonne analoghe per ogni privilegio previsto dal modello.
- `grantor`, è l'identificatore dell'utente che ha concesso l'autorizzazione.
- `grantopt` $\in$ {`Y`,`N`}, indica se i privilegi sono delegabili (`grantopt ='Y'`) o meno (`grantopt ='N'`).

Le informazioni sul timestamp sono fondamentali per la corretta implementazione dell'operazione di revoca dei privilegi (vedi Paragrafo 9.3.3.5). Il timestamp può essere sia un semplice contatore sia indicare un tempo reale. L'importante è che soddisfi le seguenti proprietà: *(i)* sia monotonicamente crescente; *(ii)* non esistano due comandi `GRANT` con lo stesso timestamp. Privilegi concessi con lo stesso comando `GRANT` hanno lo stesso timestamp. Quando un utente crea una relazione, gli vengono concessi tutti i privilegi sulla relazione con grant option. Il loro timestamp sarà quello del comando `CREATE TABLE`.

Esempio 9.10 La Figura 9.3 mostra un esempio di catalogo `Sysauth`, relativamente ai privilegi `select`, `insert` ed `update` ed alle relazioni `Clienti`, `Video` e `Film`, che riflette lo stato delle autorizzazioni dopo l'esecuzione dei comandi `GRANT` dell'Esempio 9.7. I timestamp sono stati assegnati arbitrariamente, tenendo conto delle considerazioni esposte in precedenza. Le prime tre entrate del catalogo sono automaticamente inserite dal sistema al momento della creazione delle relazioni `Clienti`, `Video` e `Film` da parte di Luca. Il timestamp coincide con quello del relativo comando `CREATE TABLE`. □

Come abbiamo visto, il catalogo `Sysauth` mantiene solo informazioni di tipo "qualitativo" sui privilegi `update` che ogni utente può esercitare, in quanto registra

utente	nomeRel	tipo	select	insert	update	grantor	grantopt
luca	Clienti	R	20	20	20		Y
luca	Video	R	22	22	22		Y
luca	Film	R	25	25	25		Y
marco	Clienti	R	0	0	30	luca	N
barbara	Film	R	32	0	0	luca	Y
giovanna	Film	R	32	0	0	luca	Y
matteo	Film	R	35	0	0	giovanna	N
elena	Video	R	40	40	40	luca	Y
elena	Film	R	40	40	40	luca	Y
barbara	Film	R	47	47	0	elena	N

Figura 9.3: Catalogo `Sysauth`

solo il fatto che un utente possa esercitare o meno il privilegio `update` su una certa relazione, ma non tiene traccia delle colonne specifiche su cui tale privilegio può essere esercitato. Tali informazioni sono mantenute nel catalogo `Syscolauth` che contiene una tupla: `(utente,nomeRel,colonna,grantor,grantopt)` per ogni colonna della relazione `nomeRel` su cui l'utente identificato da `utente` può esercitare il privilegio `update`.

9.3.3.4 Concessione di privilegi

Quando un utente richiede l'esecuzione di un comando `GRANT`, il meccanismo di controllo dell'accesso legge i cataloghi `Sysauth` e `Syscolauth` per determinare se il richiedente possiede o meno il diritto di concedere i privilegi specificati nel comando. Per effettuare tale verifica, l'insieme dei privilegi delegabili che l'utente possiede è intersecato con l'insieme dei privilegi specificati nel comando `GRANT`. Sono possibili tre risultati. Se l'intersezione è vuota, il comando non viene eseguito; se l'intersezione coincide con i privilegi presenti nel comando, il comando viene eseguito totalmente e tutti i privilegi specificati vengono quindi concessi; altrimenti, il comando viene eseguito parzialmente, solo per i privilegi contenuti nell'intersezione. Un utente ha un privilegio delegabile su una relazione se ha creato la relazione oppure ha ottenuto il privilegio con grant option da un qualsiasi grantor.

Esempio 9.11 Consideriamo il catalogo `Sysauth` illustrato nella Figura 9.3 ed i seguenti comandi `GRANT`:

```
marco: GRANT update(telefono) On Clienti TO roberto;
luca: GRANT delete ON Clienti TO giovanni, anna;
barbara: GRANT select, insert ON Film TO alessandro;
```

Il primo comando non viene eseguito, in quanto Marco ha il privilegio `update` sull'attributo `telefono` ma senza grant option. Il secondo comando viene eseguito, in quanto Luca è il proprietario della relazione `Clienti`. Infine, il terzo comando

viene parzialmente eseguito; Barbara possiede il privilegio `select` sulla relazione `Film` con grant option, mentre può esercitare il privilegio `insert` ma senza concederlo a terzi. Il risultato dell'esecuzione del comando è quindi la concessione ad Alessandro del solo privilegio `select`. □

9.3.3.5 Revoca ricorsiva

Un problema di notevole interesse è quello della semantica da attribuire all'operazione di revoca dei diritti delegabili, di quei diritti cioè concessi con grant option. Il modello di controllo dell'accesso del System R adotta la cosidetta *revoca ricorsiva* (o *a cascata* o *basata su timestamp*), in base alla quale un'operazione di revoca del privilegio p concesso 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 privilegio p sulla relazione *rel*, se u_1 non ha ottenuto tale privilegio da fonti indipendenti, ma anche di modificare l'insieme di autorizzazioni nel sistema portandolo in uno stato equivalente a quello in cui si sarebbe trovato se u_2 non avesse mai concesso ad u_1 il privilegio p sulla relazione *rel*. Pertanto, 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 mai ricevuto il privilegio revocato.

La semantica della revoca ricorsiva può essere formalmente definita come segue.

Definizione 9.2 (Revoca ricorsiva) *Siano $G_1, \ldots, G_n$ una sequenza di comandi* `GRANT` *di un singolo privilegio sulla stessa relazione, tali che se $i < j$, $1 \leq i, j \leq n$, allora il comando G_i è eseguito prima del comando G_j. Sia R_i il comando che revoca il privilegio concesso con G_i. La semantica della revoca ricorsiva impone che l'insieme di autorizzazioni nel sistema sistema dopo l'esecuzione della sequenza di comandi:*

$G_1, \ldots, G_n, R_i$

sia identico allo stato raggiunto dopo l'esecuzione della sequenza di comandi:

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

Implementare l'operazione di revoca ricorsiva non è banale. Infatti, per decidere se un privilegio deve essere ricorsivamente revocato è necessario determinare se il privilegio non avrebbe potuto essere concesso qualora il privilegio revocato non fosse mai esistito. Per caratterizzare più precisamente il problema, è opportuno dare una rappresentazione a grafo dello stato delle autorizzazioni rispetto ad un dato privilegio p ed una data relazione *rel*. Il grafo, chiamato *grafo delle autorizzazioni*, contiene un nodo per ogni utente che possiede il privilegio p sulla relazione *rel*. Esiste un arco dal nodo u_1 al nodo u_2 se l'utente u_1 ha concesso p ad u_2 su *rel*. L'arco è etichettato con il timestamp del privilegio e con la lettera 'g' se il privilegio è stato concesso con grant option ed è quindi delegabile. Il grafo contiene sempre un nodo relativo al creatore della relazione, in quanto egli è l'unico che all'inizio può concedere privilegi sulla relazione stessa. La Figura 9.4(a) illustra il grafo delle autorizzazioni relativo al privilegio `select` e alla relazione `Film`, corrispondente al catalogo `Sysauth` della Figura 9.3.

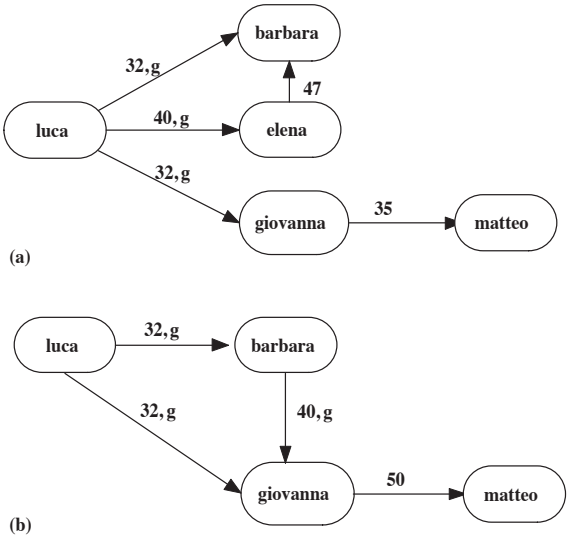

Figura 9.4: Grafi delle autorizzazioni

Esempio 9.12 Con riferimento alla Figura 9.4(a), se Luca revocasse a Giovanna il privilegio `select` sulla relazione `Film` anche il privilegio concesso da Giovanna a Matteo sarebbe revocato in quanto non avrebbe mai potuto essere concesso se Luca non avesse concesso a Giovanna il privilegio per cui richiede la revoca. □

Le informazioni sul timestamp sono fondamentali per implementare correttamente la revoca ricorsiva, come dimostra l'esempio seguente.

Esempio 9.13 Consideriamo il grafo delle autorizzazioni della Figura 9.4(b) relativo ad una sequenza di comandi `GRANT` per il privilegio `select` sulla relazione `Film` e supponiamo che Luca revochi a Giovanna il privilegio `select` sulla relazione `Film`. Questo non comporta la revoca in cascata del privilegio `select` che Giovanna ha concesso a Matteo, in quanto tale privilegio avrebbe potuto essere concesso da Giovanna al tempo 50 anche se Giovanna non lo avesse ricevuto da Luca al tempo 32, in virtù del comando `GRANT` effettuato da Barbara al tempo 40. Una situazione diversa si avrebbe se Barbara avesse concesso a Giovanna il privilegio `select` sulla relazione `Film` ad un tempo superiore al tempo 50. In questo caso, la revoca effettuata da Luca comporterebbe anche la revoca del privilegio concesso da Giovanna a Matteo in quanto al tempo 50 Giovanna non avrebbe potuto concedere il privilegio `select` a Matteo se non lo avesse ricevuto da Luca. □

La procedura che implementa la revoca ricorsiva è presentata nella Figura 9.5. La procedura riceve in ingresso una richiesta di revoca (*revoker,privilegio, relazione,revokee*), dove *revoker* è l'utente che richiede la revoca, mentre *revokee*

Procedura *revoca_ricorsiva*(*revoker,privilegio,relazione,revokee*)
`Begin`

1. Seleziona da `Sysauth` le tuple con `utente`=*revokee*, `nomeRel`=*relazione*, `grantor`=*revoker*
2. Poni a zero il valore dell'attributo `privilegio` nelle tuple selezionate, oppure cancella le tuple se tutte le entrate relative ai privilegi sono uguali a zero
3. `If privilegio=update Then`
   Cancella le tuple con `utente`=*revokee*, `nomeRel`=*relazione* e `grantor`=*revoker* da `Syscolauth`
   `EndIf`
4. *min_tm* := `CURRENT_TIME`
5. `For` ogni tupla in `Sysauth` con `utente`=*revokee*, `nomeRel`=*relazione* e `grantopt=Y Do`
   `If privilegio` $\neq 0$ e `privilegio` $<$ *min_tm* `Then` *min_tm* := `privilegio`
   `EndFor`
6. `For` ogni utente u tale che esiste in `Sysauth` una tupla con `utente`=u, `nomeRel`=*relazione* e `grantor`=*revokee* `Do`
   `If privilegio` $<$ *min_tm* `Then`
   revoca_ricorsiva(*revokee,privilegio,relazione,u*)
   `EndIf`
   `EndFor`

`End`

Figura 9.5: Procedura per la revoca ricorsiva

è l'utente a cui il privilegio viene revocato. Tale procedura modifica i cataloghi `Sysauth` e `Syscolauth` per eliminare tutti i privilegi che, in base alla semantica della revoca ricorsiva, devono essere revocati in seguito all'esecuzione del comando di revoca. La procedura è una procedura ricorsiva di cui illustriamo il funzionamento tramite un esempio. Supponiamo che l'utente A revochi il privilegio p all'utente B sulla relazione rel. I passi (1) e (2) della procedura *revoca_ricorsiva* aggiornano il catalogo `Sysauth` per eliminare il privilegio p concesso da A a B su rel. Se il privilegio revocato è `update` è inoltre necessario aggiornare il catalogo `Syscolauth` (passo (3)). Per determinare se le autorizzazioni per p concesse da B ad altri utenti debbano essere ricorsivamente revocate, la procedura determina, interrogando opportunamente il catalogo `Sysauth`, il tempo minimo, se diverso da zero, a cui p è stato concesso con grant option a B da utenti diversi da A (passi (4) e (5)). Un'autorizzazione per p concessa da B ad altri utenti è ricorsivamente revocata se il tempo in cui è stata specificata è maggiore del valore calcolato al passo precedente. Il procedimento è ripetuto per ogni utente i cui privilegi sono stati modificati durante l'esecuzione della procedura (passo (6)), fintantoché non sono necessarie ulteriori modifiche.

La procedura appena illustrata opera correttamente anche in presenza di cicli nel grafo delle autorizzazioni (vedi Esercizio 9.3 alla fine di questo capitolo). L'esempio seguente mostra il funzionamento della procedura di revoca.

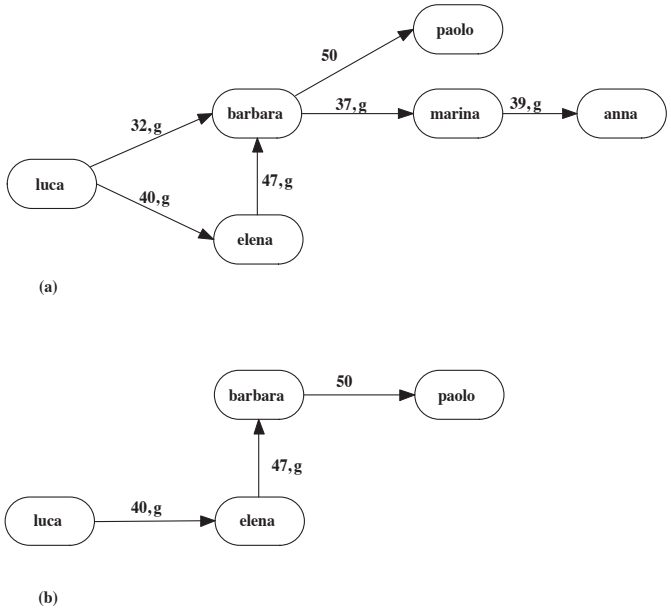

Figura 9.6: Autorizzazioni: (a) stato iniziale; (b) stato dopo la revoca di Luca a Barbara

Esempio 9.14 Consideriamo il grafo delle autorizzazioni della Figura 9.6(a) relativo al privilegio `select` sulla relazione `Film` ed il relativo catalogo `Sysauth` riportato nella Figura 9.7. Supponiamo che Luca revochi a Barbara il privilegio `select`. La procedura della Figura 9.5 opera come segue:

1. Cancella dal catalogo `Sysauth` la tupla `(barbara,Film,R,32,luca,Y)`.
2. Calcola il minimo tra i timestamp dei privilegi `select` rimasti a Barbara con grant option sulla relazione `Film` dopo l'operazione di revoca: $min\_tm = 47$.
3. Considera i privilegi `select` sulla relazione `Film` concessi da Barbara ad altri utenti e confronta il loro timestamp con il minimo calcolato al passo precedente:
   - l'autorizzazione concessa da Barbara a Paolo ha timestamp $50 > min\_tm$: l'autorizzazione non viene revocata;
   - l'autorizzazione concessa da Barbara a Marina ha timestamp $37 < min\_tm$: la procedura viene ricorsivamente richiamata per eliminare tale autorizzazione.

I passi (1), (2) e (3) vengono eseguiti per ogni utente i cui privilegi sono stati modificati in seguito all'operazione di revoca. Come conseguenza, anche l'auto-

utente	nomeRel	tipo	select	grantor	grantopt
luca	Film	R	25		Y
barbara	Film	R	32	luca	Y
marina	Film	R	37	barbara	Y
anna	Film	R	39	marina	Y
elena	Film	R	40	luca	Y
barbara	Film	R	47	elena	Y
paolo	Film	R	50	barbara	N

Figura 9.7: Catalogo `Sysauth` per le autorizzazioni della Figura 9.6(a)

rizzazione concessa da Marina ad Anna viene ricorsivamente revocata. Il grafo risultante è mostrato nella Figura 9.6(b). □

La revoca ricorsiva, sebbene abbia una semantica ben definita e sia stata alla base di quasi tutti i modelli di controllo dell'accesso dei DBMS commerciali, è stata anche oggetto di critiche in quanto può portare alla cancellazione di un eccessivo numero di autorizzazioni. Infatti, spesso le autorizzazioni che un utente possiede dipendono dal ruolo che l'utente ricopre all'interno dell'organizzazione. Se un utente cambia ruolo, ad esempio in seguito ad una promozione, può essere utile revocare all'utente le autorizzazioni che possedeva, ma non quelle che egli aveva concesso ad altri utenti. Per tali motivi, sono state proposte dalla comunità scientifica semantiche alternative a quella della revoca ricorsiva. Ad esempio, è stato introdotto un nuovo tipo di revoca, chiamata *revoca senza cascata*, in base alla quale, quando un utente revoca un privilegio ad un altro utente, le autorizzazioni concesse da quest'ultimo in virtù del privilegio revocato non sono revocate ricorsivamente ma vengono rispecificate come se fossero state concesse dall'utente che ha richiesto l'operazione di revoca.

Anche lo standard SQL ha recepito questa linea di tendenza. Infatti, a partire dallo standard SQL:1999, la revoca può essere richiesta *con o senza cascata*. La revoca senza cascata prevede che un'operazione di revoca non venga eseguita se comporta la cancellazione di altre autorizzazioni, oltre a quella oggetto del comando. La revoca con cascata, invece, è una revoca ricorsiva, in cui però non sono considerati i timestamp delle autorizzazioni (vedi Paragrafo 9.4 per ulteriori dettagli).

9.3.3.6 Autorizzazioni su viste

Le viste sono un importante meccanismo attraverso cui è possibile fornire forme più sofisticate di controllo dell'accesso, rispetto a quelle illustrate fino ad ora. Ad esempio, possono essere frequenti le situazioni in cui non tutti gli attributi di una relazione hanno gli stessi requisiti di protezione. La sintassi del comando `GRANT` discussa nel Paragrafo 9.3.3.1 non consente di concedere privilegi solo su alcuni attributi di una relazione, ad eccezione del privilegio `update`. Inoltre, per

la sintassi del comando `GRANT`, non è possibile autorizzare un utente a vedere solo alcune colonne di una relazione (ad esempio quelle relative al nome e cognome dei clienti). Questa politica per il controllo dell'accesso può essere implementata, nel modello del System R, mediante l'uso di viste. In questo caso, è sufficiente definire una vista come proiezione sugli attributi su cui vogliamo concedere i privilegi (`nome` e `cognome`) ed autorizzare l'accesso alla vista invece che alla relazione di base. Inoltre, le viste permettono di concedere privilegi *statistici*. Ad esempio, un utente potrebbe non essere autorizzato a vedere i titoli dei film noleggiati da un cliente, ma solo il numero di noleggi da lui effettuati. Tramite le viste, è possibile soddisfare questo requisito di sicurezza: basta definire una vista che computa il numero di noleggi effettuati da ogni cliente e concedere all'utente l'accesso alla vista invece che alle relazioni di base. Infine, le viste consentono di realizzare il cosidetto *controllo dell'accesso in base al contenuto*, ovvero permettono di autorizzare l'accesso solo a specifiche tuple di una relazione, sulla base dei valori dei loro attributi. Ad esempio, se vogliamo autorizzare un utente a vedere solo le tuple della relazione `Film` relative a commedie, è sufficiente definire una vista che seleziona dalla relazione `Film` le tuple che soddisfano tale condizione e concedere all'utente il privilegio `select` sulla vista, invece che sulla relazione di base.

L'utilizzo delle viste comporta quindi notevoli vantaggi rispetto alla tipologia di politiche che possiamo implementare. È però opportuno chiarire alcuni aspetti connessi al loro corretto utilizzo.

In primo luogo, un utente può creare una vista solo se ha il privilegio `select` sulle relazioni/viste su cui è definita. Un'altra importante questione riguarda i privilegi che l'utente che crea una vista può esercitare sulla vista stessa. In precedenza, abbiamo visto come l'utente che crea una relazione di base possa esercitare su di essa tutti i privilegi previsti dal modello del System R con facoltà di delega. Per le viste, invece, i privilegi che l'utente che le definisce ha su di esse dipendono da due fattori: *(i)* le autorizzazioni che l'utente possiede sulle relazioni/viste su cui la vista è definita;*(ii)* la semantica della vista, ovvero la sua definizione in termini delle relazioni o viste componenti.

Rispetto al punto *(i)*, se una vista è definita su una singola relazione (o vista), i privilegi che l'utente che crea la vista ha su di essa sono gli stessi che ha sulla relazione o vista componente. I privilegi sulla vista saranno delegabili o meno, a seconda di come sono definiti per la relazione o vista componente. Nel caso in cui la vista sia definita su più relazioni (o viste), i privilegi che l'utente che crea la vista ha su di essa sono ottenuti intersecando i privilegi che l'utente ha su tutte le relazioni (o viste) componenti. Inoltre, un privilegio sulla vista è delegabile solo se il creatore della vista ha il diritto di delegare tale privilegio su tutte le relazioni o viste componenti. Per quanto concerne il punto *(ii)*, abbiamo visto nel Capitolo 3 come SQL ponga alcune restrizioni sulle operazioni che possono essere effettuate su una vista. Ad esempio, una vista che computa delle statistiche non può essere aggiornata. Queste restrizioni si riflettono anche sulle autorizzazioni che un utente che crea una vista ha sulla vista stessa. Le autorizzazioni che un utente ha sulle relazioni/viste su cui la vista è definita potranno essere esercitate anche sulla vista

solo se l'interrogazione di definizione della vista lo consente.

Esempio 9.15 Con riferimento al catalogo `Sysauth` della Figura 9.3, supponiamo che Barbara esegua il comando:

```
CREATE VIEW Commedie AS
SELECT * FROM Film
WHERE genere = 'commedia';
```

L'esecuzione di tale comando viene autorizzata in quanto Barbara ha il privilegio `select` su `Film`. Questo è l'unico privilegio che Barbara può esercitare sulla vista appena creata, in quanto è l'unico che possiede sulla relazione `Film`. Inoltre, Barbara può concedere a terzi tale privilegio, avendolo ricevuto da Luca con grant option. Supponiamo ora che Elena crei la vista `NumNoleggi`, definita come segue:

```
CREATE VIEW NumNoleggi AS
SELECT codCli, COUNT(*) AS NumNol
FROM Noleggi
GROUP_BY codCli;
```

In base al catalogo `Sysauth` della Figura 9.3 Elena ha sulla relazione `Film` i privilegi `select`, `insert` ed `update`. Per le considerazioni viste in precedenza, questi privilegi dovrebbero essere esercitabili da Elena anche sulla vista a meno che l'interrogazione di definizione della vista sia tale che alcuni di essi non siano esercitabili. Le restrizioni discusse nel Capitolo 3, fanno sì che i privilegi `update` ed `insert` non siano esercitabili sulla vista. Elena può quindi esercitare sulla vista solo il privilegio `select`. Inoltre, siccome possedeva tale privilegio con grant option sulla relazione `Film` potrà concedere a terzi l'autorizzazione di selezionare tuple dalla vista `NumNoleggi`. □

Quando un utente crea una vista, il catalogo `Sysauth` viene aggiornato con due tuple a lui relative, corrispondenti ai privilegi, delegabili e non, che l'utente può esercitare. I privilegi sono determinati in base ai punti *(i)* e *(ii)* illustrati in precedenza. Il timestamp associato ai privilegi è quello del relativo comando `CREATE VIEW`.

La concessione di privilegi su una vista è molto simile a quella su relazioni di base: i privilegi che un utente può concedere ad altri su una vista sono quelli che possiede con grant option. Le operazioni di revoca sono, invece, più complicate, in quanto è necessario stabilire cosa succede ad una vista se un privilegio `select` su una delle relazioni su cui è definita viene revocato. L'approccio seguito dal System R, in accordo alla semantica della revoca ricorsiva, è quello di cancellare la vista se il privilegio revocato era l'unico utile per la sua definizione.

Esempio 9.16 Consideriamo le viste dell'Esempio 9.15 e il catalogo `Sysauth` riportato nella Figura 9.3 e supponiamo che Luca revochi ad Elena il privilegio `select` sulla relazione `Noleggio`. In questo caso, la vista `NumNoleggi` viene a sua

volta cancellata in quanto il privilegio `select` che Luca aveva concesso ad Elena era per lei l'unico utile per creare la vista. Viceversa, se Luca revocasse a Barbara il privilegio `select` su `Film`, la decisione se cancellare o meno la vista `Commedie` creata da Barbara dipenderebbe dal timestamp del comando `CREATE VIEW`. Se il timestamp è maggiore di 47 la vista non è cancellata in quanto Barbara ha ricevuto da Elena il privilegio `select` su `Film` al tempo 47. Invece, nel caso in cui il comando `CREATE VIEW` sia stato eseguito da Barbara prima del tempo 47, la vista viene ricorsivamente revocata. □

9.3.4 Modelli basati su ruoli

Una delle più rilevanti estensioni proposte ai modelli di controllo dell'accesso fino ad ora illustrati è l'introduzione del concetto di *ruolo*, brevemente descritto nel Paragrafo 9.2.1.2. Nel *controllo dell'accesso basato su ruoli* (*RBAC – Role Based Access Control*), i ruoli rappresentano delle funzioni che gli utenti ricoprono all'interno dell'organizzazione o azienda in cui operano (sono esempi di ruolo per il dominio della videoteca `commesso`, `cliente` e `direttoreVideoteca`). I privilegi sono concessi ai ruoli invece che ai singoli utenti. Le autorizzazioni specificate per un ruolo sono quelle necessarie per esercitare le funzioni connesse al ruolo stesso. Gli utenti sono abilitati a ricoprire uno o più ruoli, in base alle mansioni che devono svolgere. L'abilitazione a ricoprire un ruolo implica l'acquisizione di tutte le autorizzazioni ad esso connesse.

Il controllo dell'accesso basato su ruoli ha suscitato subito un notevole interesse, dimostrato dai numerosi modelli di controllo dell'accesso proposti da gruppi di ricerca e dal fatto che quasi tutti i DBMS commerciali e lo standard SQL forniscono tale funzionalità, seppure con delle differenze. Per supplire a questa mancanza di standardizzazione e rendere più agevole lo sviluppo di meccanismi per il controllo dell'accesso basati su ruoli, è stato definito uno standard NIST (*National Institute of Standards and Technology*) che introduce un modello di riferimento. Lo standard è stato concepito in maniera modulare ed è costituito da tre componenti: *Modello base* (*Core RBAC*), *Modello gerarchico* (*Hierarchical RBAC*), *Modello con vincoli* (*Constrained RBAC*). Il modello base definisce i requisiti minimi per realizzare il controllo dell'accesso basato su ruoli. Gli altri due modelli, non dipendenti l'uno dall'altro, aggiungono al modello base, rispettivamente, la strutturazione gerarchica dei ruoli e la possibilità di specificare vincoli sull'attivazione e sull'assegnazione dei ruoli. L'idea alla base di questa organizzazione modulare dello standard è che non esiste un unico modello adatto a tutti gli ambienti ed i domini applicativi. In questo modo, è possibile conformarsi allo standard realizzando una sola o più componenti, a seconda delle specifiche esigenze. Nel seguito, illustreremo brevemente ognuna delle componenti dello standard.

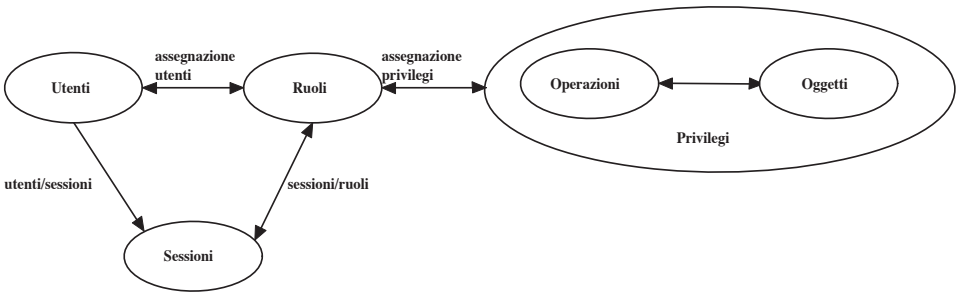

Figura 9.8: Modello RBAC base

9.3.4.1 Modello base

Il modello base si compone di quattro componenti principali: *Utenti*, *Ruoli*, *Privilegi* e *Sessioni*. Le prime due componenti sono già state illustrate in precedenza. I *Privilegi* indicano i diritti d'accesso esercitabili sugli oggetti del sistema. Gli elementi dell'insieme *Privilegi* sono coppie (obj, op), dove obj è un oggetto ed op è un'operazione. Infine, il concetto di sessione è stato introdotto perché un utente, quando effettua il log-in al sistema, può attivare un sottoinsieme dei ruoli che è abilitato a ricoprire. Una sessione stabilisce quindi la corrispondenza tra un utente ed i ruoli attivi durante la sua connessione al sistema. Il modello base stabilisce inoltre le relazioni che intercorrono tra le componenti appena introdotte, illustrate graficamente nella Figura 9.8, dove le doppie frecce indicano relazioni molti a molti.

Dall'analisi della Figura 9.8 è evidente come il modello base preveda che un utente possa essere assegnato a più ruoli, mentre un ruolo può essere ricoperto da più utenti (tutti quelli che ricoprono la funzione ad esso associata). Le autorizzazioni sono specificate per i ruoli e non per utenti singoli (infatti non esiste alcuna freccia che collega utenti a privilegi). Ruoli e privilegi sono anch'essi legati da una relazione molti a molti. Infine, ogni sessione mette in corrispondenza un utente con l'insieme dei ruoli attivi per quella sessione. Ogni sessione è associata ad un singolo utente, mentre un utente può attivare diverse sessioni (e diversi ruoli in ognuna di esse).

9.3.4.2 Modello gerarchico

Il modello base non prevede alcuna strutturazione gerarchica dell'insieme dei ruoli (per questo è anche conosciuto come "*Modello flat*"). Il modello gerarchico aggiunge al modello base la possibilità di strutturare i ruoli in gerarchie. Questa funzionalità pemette di modellare al meglio tutte quelle realtà dove le funzioni aziendali od organizzative sono strutturate in una gerarchia che riflette diversi livelli di autorità e responsabilità. Una gerarchia sui ruoli, di cui un esempio è illustrato nella Figura 9.9, definisce quindi un ordinamento parziale tra di essi, in

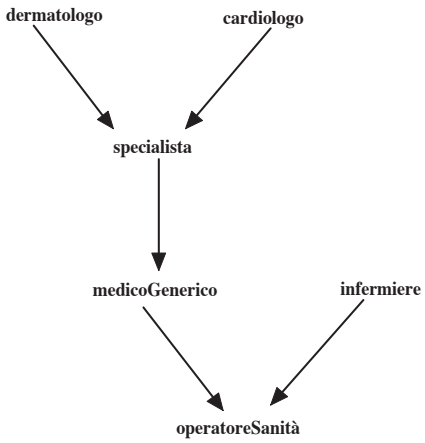

Figura 9.9: Gerarchia dei ruoli

quanto non tutti i ruoli sono necessariamente in relazione gerarchica. Tale gerarchia induce un'ereditarietà dei privilegi tra ruoli nella gerarchia e stabilisce una relazione tra gli utenti abilitati a ricoprire i vari ruoli.

Più precisamente, il modello gerarchico introduce una relazione d'ordine parziale sull'insieme dei ruoli, chiamata *relazione di ereditarietà* e denotata con $\geq$, tale per cui, dati due ruoli r_1 ed r_2, $r_1 \geq r_2$[4] implica che r_1 eredita tutti i privilegi assegnati ad r_2 e tutti gli utenti associati ad r_1 sono anche utenti associati ad r_2. Questo principio di ereditarietà è motivato dal fatto che, in linea di principio, un ruolo dovrebbe poter effettuare sugli oggetti del sistema tutte le operazioni che possono essere effettuate da ruoli corrispondenti a funzioni più in basso nella gerarchia. Naturalmente, possono esistere casi in cui tale principio generale può non essere conveniente. Pensiamo ad esempio ad un sistema per la sorveglianza di un impianto critico, dove alcuni addetti molto specializzati sono capaci di compiere delle operazioni che i loro responsabili non sono in grado di effettuare. In queste situazioni, basta configurare la gerarchia dei ruoli in maniera tale da non mettere in relazione i due ruoli per cui il principio di ereditarietà non deve valere.

9.3.4.3 Modello con vincoli

Il modello con vincoli aggiunge al modello base la possibilità di specificare vincoli. In realtà, lo standard attuale considera solo una tipologia di vincolo, nota come *separazione delle mansioni* (*SoD - Separation of Duties*), mentre altre categorie di vincoli, quali ad esempio vincoli sul tempo di attivazione dei ruoli, non sono al momento considerate nello standard. La separazione delle mansioni è un vincolo

[4] r_1 è detto anche *ruolo senior*, mentre r_2 *ruolo junior*.

di sicurezza molto importante che impone delle restrizioni sull'insieme dei ruoli esercitabili o attivabili da un certo utente, per prevenire situazioni che potrebbero portare un utente ad avere troppo potere senza un adeguato controllo. I vincoli per la separazione delle mansioni possono essere classificati in due categorie: *vincoli statici* e *vincoli dinamici*. I vincoli statici stabiliscono delle relazioni di mutua esclusione tra ruoli assegnabili ad un certo utente. Ad esempio, la politica aziendale potrebbe impedire che un utente sia assegnato contemporaneamente al ruolo `segretarioAmministrativo` e `revisoreConti`, in quanto il secondo esercita una funzione di controllo sul primo. Nel modello con vincoli questa nozione di ruoli incompatibili è generalizzata a più di due ruoli. Ad esempio, per ragioni di garanzia, potremmo imporre che ogni utente non possa ricoprire più di due ruoli tra quelli che corrispondono a funzioni direttive all'interno di un'organizzazione. Più precisamente, un vincolo di separazione delle mansioni di tipo statico è formalizzato come una coppia (RS, n), dove RS è un sottoinsieme dei ruoli previsti dal modello ed n è un numero naturale. Tale vincolo stabilisce che nessun utente può essere abilitato a ricoprire un numero di ruoli maggiore o uguale ad n tra quelli in RS. Se i ruoli sono strutturati in gerarchia, i vincoli di separazione delle mansioni vengono propagati lungo la gerarchia. Ad esempio, con riferimento alla Figura 9.9, se per il ruolo `specialista` è specificato un vincolo di separazione delle mansioni, questo vale anche per i ruoli senior `cardiologo` e `dermatologo`.

Abbiamo visto come i vincoli statici di separazione delle mansioni impongano delle restrizioni sull'insieme di ruoli che possono essere ricoperti da un utente. Ci sono però situazioni in cui un utente potrebbe essere autorizzato ad attivare due ruoli separatamente, perché questo non causa alcun problema di conflitto di interessi, mentre non dovrebbe essere autorizzato ad attivarli entrambi nella stessa sessione. Ad esempio, un utente potrebbe essere autorizzato ad attivare il ruolo `cassiere` e `supervisoreCassiere` in due momenti diversi, ma non contemporaneamente nella stessa sessione (in quanto il principio di separazione delle mansioni impone che le funzioni operative e di controllo siano mantenute separate). I vincoli dinamici di separazione delle mansioni consentono di imporre delle restrizioni sull'insieme dei ruoli che un utente può attivare in una sessione. Formalmente, sono modellati come i vincoli statici, cioè come una coppia (RS, n), dove RS è un sottoinsieme dei ruoli previsti dal modello ed n è un numero naturale. La semantica è però diversa in quanto un vincolo dinamico impone che nessun utente possa attivare contemporaneamente un numero di ruoli maggiore o uguale ad n tra quelli in RS.

9.4 Controllo dell'accesso in SQL

SQL è basato, per quanto riguarda il controllo dell'accesso, sul modello del System R illustrato nel Paragrafo 9.3.3, però con alcune importanti differenze ed estensioni. Tra le più rilevanti differenze vi è il supporto per i ruoli, una diversa semantica per l'operazione di revoca e l'aggiunta di nuovi privilegi ed oggetti di autorizzazione. Nel seguito, illustreremo solo le caratteristiche fondamentali del controllo

dell'accesso in SQL, focalizzandoci sulle differenze rispetto al modello di controllo dell'accesso del System R.

La prima differenza con il modello di controllo dell'accesso del System R è la possibilità di definire ruoli. I ruoli possono essere creati con il comando `CREATE ROLE <nome ruolo>` ed eliminati con il comando `DROP ROLE <nome ruolo>`, dove `<nome ruolo>` indica il nome del ruolo che vogliamo creare o cancellare, rispettivamente. Altre differenze riguardano i comandi `GRANT` e `REVOKE`, oggetto dei successivi paragrafi.

9.4.1 Comando `GRANT`

Il comando `GRANT` del System R è stato esteso con la possibilità di concedere privilegi non solo ad utenti ma anche a ruoli. Inoltre, il comando è stato esteso con la possibilità di autorizzare un utente non solo all'esercizio di privilegi ma anche a ricoprire uno o più ruoli. Altre estensioni, come ad esempio l'insieme di privilegi ed oggetti previsti dal comando, riflettono le estensioni apportate al modello relazionale nel corso degli anni (discusse brevemente nel Capitolo 1). Alcune di tali estensioni saranno illustrate in dettaglio nei Capitoli 10 e 11 a cui rimandiamo il lettore per una piena comprensione degli argomenti nel seguito illustrati.

Più precisamente, il comando `GRANT` in SQL ha una duplice sintassi, a seconda che venga utilizzato per concedere privilegi (ad utenti o ruoli) o per abilitare utenti/ruoli a ricoprire ruoli. La prima forma ha la sintassi di seguito illustrata:

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

dove:

- `<lista privilegi>` indica l'insieme dei privilegi concessi con il comando `GRANT`. La parola chiave `ALL PRIVILEGES` indica tutti i privilegi previsti dal modello.
- `<nome oggetto>` indica il nome dell'oggetto della base di dati su cui sono concessi i privilegi. In alcuni casi, è inoltre necessario specificare un qualificatore dell'oggetto (`<qualificatore oggetto>`) prima dell'oggetto stesso (ad esempio usiamo la parola chiave `TYPE` se l'oggetto dell'autorizzazione è un tipo definito dall'utente).[5]
- `<lista utenti>` indica l'insieme degli utenti a cui vengono concessi i privilegi.
- `<lista ruoli>` indica l'insieme dei ruoli a cui vengono concessi i privilegi. La parola chiave `PUBLIC` consente di specificare le autorizzazioni implicate dal comando per tutti gli utenti/ruoli del sistema.

[5] I tipi definiti dall'utente verranno illustrati nel Capitolo 10.

- La clausola opzionale `WITH GRANT OPTION` consente la delega dell'amministrazione dei privilegi oggetto del comando, in analogia al corrispondente comando nel modello del System R (vedi Paragrafo 9.3.3.1).

- La clausola opzionale `WITH HIERARCHY OPTION` può essere specificata solo per il privilegio `select` e consente, nel caso di relazioni legate da una gerarchia di ereditarietà (vedi Capitolo 10 per una trattazione di questi argomenti), di propagare il privilegio a tutte le sotto-tabelle della tabella riferita nel comando `GRANT`.

Vediamo ora quali sono le principali estensioni all'insieme dei privilegi ed oggetti di autorizzazione rispetto a quelli previsti dal modello del System R. Innanzitutto i privilegi `select` e `insert`, in aggiunta al privilegio `update`, possono essere concessi selettivamente solo su alcune colonne di una relazione (vale la stessa sintassi vista nel Paragrafo 9.3.3.1 per il privilegio `update`). Altri privilegi previsti dallo standard, dovuti alle estensioni apportate al modello relazionale nel corso degli anni, sono: `references`, che permette di utilizzare una colonna in un vincolo o asserzione, `trigger`, che permette di specificare trigger che operano su una certa relazione,[6] `under`, che permette la creazione di sotto-tipi o sotto-tabelle,[7] `usage`, che permette di utilizzare un oggetto dello schema (ad esempio un tipo) nella definizione di un altro oggetto, ed `execute`, che permette l'esecuzione di una procedura o funzione. SQL pone alcune restrizioni sui privilegi specificabili in relazione alla tipologia di oggetto. Ad esempio, i privilegi `usage` ed `execute` non possono essere specificati per relazioni, mentre `trigger` può essere specificato solo per relazioni di base.

Anche SQL, come il modello del System R, prevede l'amministrazione decentralizzata dei privilegi mediante ownership. Inoltre, il proprietario di un oggetto è l'unico abilitato ad esercitare su di esso i privilegi `drop` (cancellazione) e `alter` (modifica).

Come abbiamo detto in precedenza, il comando `GRANT` ha in SQL una duplice sintassi per consentire, oltre alla concessione di privilegi, anche di abilitare utenti/ruoli a ricoprire dei ruoli. In questa seconda accezione la sintassi è la seguente:

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

dove:

- `<lista ruoli concessi>` indica l'insieme dei ruoli concessi con il comando `GRANT`.

- `<lista utenti>` indica l'insieme degli utenti per cui vengono abilitati i ruoli concessi con il comando.

[6] I trigger saranno oggetto del Capitolo 11.
[7] Vedi il Capitolo 10 per ulteriori dettagli.

- `<lista ruoli>` indica l'insieme dei ruoli per cui vengono abilitati i ruoli concessi con il comando. La parola chiave `PUBLIC` consente di abilitare i ruoli specificati nel comando per tutti gli utenti/ruoli del sistema.

- la clausola opzionale `WITH ADMIN OPTION` è l'analogo per i ruoli della grant option; quando si è abilitati a ricoprire un ruolo con admin option non solo vengono ereditate tutte le autorizzazioni specificate per quel ruolo, ma è anche possibile concedere a terzi la possibilità di ricoprire il ruolo.

Notiamo che la sintassi del comando `GRANT` permette di abilitare un ruolo a ricoprirne un altro (ereditando quindi le sue autorizzazioni). Questo è il modo con cui SQL fornisce implicitamente la possibilità di strutturare i ruoli in gerarchia (vedi Paragrafo 9.3.4.2).

Esempio 9.17 I seguenti sono esempi di comandi `GRANT`:

```
GRANT usage ON TYPE indirizzo TO giovanna WITH GRANT OPTION;
GRANT execute ON aggiornaClienti TO elena;
GRANT select(nome, cognome), references(codCli) ON Clienti TO marco;
GRANT delete, update ON Clienti TO direttoreVideoteca;
GRANT direttoreVideoteca TO roberto WITH ADMIN OPTION;
```

Il primo comando consente a Giovanna di utilizzare il tipo `indirizzo` nella definizione di altri oggetti dello schema e di concedere a terzi tale privilegio. Il secondo comando consente ad Elena di eseguire la procedura `aggiornaClienti`, mentre il terzo comando concede a Marco la possibilità di formulare interrogazioni sugli attributi `nome` e `cognome` della relazione `Clienti` e di specificare vincoli che coinvolgono l'attributo `codCli` della stessa relazione. Gli ultimi due comandi, invece, coinvolgono ruoli. Il primo assegna al ruolo `direttoreVideoteca` il privilegio di cancellare e modificare tuple della relazione `Clienti`, mentre il secondo assegna a Roberto il ruolo `direttoreVideoteca` e la facoltà di abilitare terzi a ricoprire questo ruolo. □

9.4.2 Comando `REVOKE`

Anche per quanto riguarda l'operazione di revoca lo standard SQL prevede alcune importanti novità rispetto al modello del System R. Come per il comando `GRANT` anche per il comando `REVOKE` vi è una duplice sintassi, a seconda che lo utilizziamo per la revoca di privilegi o di ruoli. Iniziamo, in primo luogo, a vedere la sintassi del comando per la revoca di privilegi:

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

dove:

- le clausole opzionali `GRANT OPTION FOR` e `HIERARCHY OPTION FOR` servono per revocare la sola grant o hierarchy option, mantenendo il diritto ad esercitare i privilegi oggetto del comando di revoca.
- `<lista privilegi>` indica l'insieme di privilegi oggetto del comando di revoca.
- `<nome oggetto>` indica il nome dell'oggetto della base di dati su cui sono revocati i privilegi. In alcuni casi è inoltre necessario specificare un qualificatore dell'oggetto (`<qualificatore oggetto>`) prima dell'oggetto stesso, in analogia a quanto visto per il comando `GRANT`.
- `<lista utenti>` indica l'insieme degli utenti a cui sono revocati i privilegi.
- `<lista ruoli>` indica l'insieme dei ruoli a cui sono revocati i privilegi.
- Le clausole `RESTRICT` e `CASCADE` servono per gestire l'operazione di revoca ed il loro significato verrà precisato in seguito.

Il comando per revocare l'abilitazione a ricoprire ruoli ha la seguente sintassi:

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

dove:

- la clausola opzionale `ADMIN OPTION FOR` serve per revocare la sola admin option, mantenendo il diritto a ricoprire i ruoli oggetto del comando di revoca.
- `<lista ruoli revocati>` indica l'insieme di ruoli revocati con il comando.
- `<lista utenti>` indica l'insieme degli utenti a cui viene revocata l'abilitazione a ricoprire i ruoli oggetto della revoca.
- `<lista ruoli>` indica l'insieme dei ruoli a cui viene revocata l'abilitazione a ricoprire i ruoli oggetto della revoca.
- le clausole `RESTRICT` e `CASCADE` servono per gestire l'operazione di revoca ed il loro significato verrà precisato in seguito.

La prima differenza con l'analogo comando del System R è che il comando `REVOKE` in SQL può essere utilizzato per revocare sia privilegi sia autorizzazioni a ricoprire un ruolo. Inoltre, vi è la possibilità di revocare solo la grant o l'admin o la hierarchy option, senza revocare il corrispondente privilegio.

Esempio 9.18 Consideriamo i comandi `GRANT` dell'Esempio 9.17 ed i seguenti comandi `REVOKE`:

```
REVOKE GRANT OPTION FOR usage ON TYPE indirizzo FROM giovanna;
REVOKE delete ON Clienti FROM direttoreVideoteca;
REVOKE direttoreVideoteca FROM roberto;
```

Il primo comando è un esempio di revoca della sola grant option. L'effetto è che Giovanna può ancora utilizzare il tipo `indirizzo` nella specifica di altri oggetti dello schema, ma non può più concedere a terzi questo privilegio. Con il secondo comando viene revocato al ruolo `direttoreVideoteca` (e quindi implicitamente anche a tutti gli utenti abilitati a ricoprirlo) il privilegio di cancellare tuple dalla relazione `Clienti`. Infine, con il terzo comando viene revocata a Roberto l'abilitazione a ricoprire il ruolo `direttoreVideoteca`. □

Una delle differenze più importanti dello standard SQL rispetto al modello di controllo dell'accesso del System R riguarda la semantica da attribuire all'operazione di revoca. Lo standard prevede due possibilità. Se la revoca di un privilegio è richiesta con l'opzione `RESTRICT`, allora l'esecuzione del comando non viene concessa se questo comporta la revoca di altri privilegi, oppure la cancellazione di oggetti dello schema (ad esempio nel caso di viste create grazie al privilegio revocato). Se la revoca invece è richiesta con l'opzione `CASCADE`, viene implementata una revoca, simile alla revoca ricorsiva del System R (vedi Paragrafo 9.3.3.5), ma senza considerare i timestamp. Le autorizzazioni che un utente a cui è stato revocato un privilegio ha specificato non sono ricorsivamente revocate se l'utente ha ricevuto da altre fonti indipendenti il privilegio specificato nel comando `REVOKE` con grant option, indipendentemente dal tempo in cui ha ricevuto il privilegio; sono ricorsivamente revocate, in caso contrario. Il seguente esempio chiarisce le differenze.

Esempio 9.19 Consideriamo l'Esempio della Figura 9.4(b) e supponiamo che Barbara abbia concesso a Giovanna il privilegio `select` su `Film` al tempo 55 invece che 40. Supponiamo ora che Luca revochi a Giovanna il privilegio concessole al tempo 32. Questo non comporterebbe, come invece avviene nel modello del System R, la revoca del privilegio concesso da Giovanna a Matteo, in quanto Giovanna continua ad avere il privilegio `select` con grant option sulla relazione `Film` concessole da Barbara. □

Considerazioni analoghe valgono per l'operazione di revoca dei ruoli.

Note conclusive

Proteggere le informazioni memorizzate in un DBMS è un compito complesso che coinvolge diversi attori: dai servizi del DBMS stesso, alle reti, ai sistemi operativi. In questo capitolo, ci siamo concentrati sul controllo dell'accesso in quanto

i meccanismi che lo realizzano sono per la maggior parte di stretta competenza del DBMS. Inoltre, ci siamo principalmente focalizzati sulla protezione dei dati memorizzati in un DBMS relazionale. Per ragioni di spazio, abbiamo tralasciato una serie di importanti problematiche che hanno dato origine ad una notevole attività di ricerca. Innanzitutto, una cospicua attività di ricerca è stata condotta per definire modelli di controllo dell'accesso per DBMS non relazionali, quali ad esempio DBMS ad oggetti, attivi [FT00] e, più recentemente, per sorgenti dati XML [BS05]. Inoltre, ci sono state numerose proposte volte ad estendere il potere espressivo dei modelli discrezionali più noti, quali ad esempio il modello del System R. Tra le principali proposte, ricordiamo il supporto per autorizzazioni positive e negative, autorizzazioni basate sul contenuto, autorizzazioni forti e deboli, autorizzazioni implicite definite tramite opportune regole, ed autorizzazioni temporali [FT00]. Un'altra rilevante area di ricerca è stata quella connessa allo sviluppo di modelli di controllo dell'accesso di tipo mandatorio. Numerosi modelli sono stati proposti così come prototipi e prodotti commerciali [FT00]. In tale ambito, sono stati anche studiati problemi relativi all'inferenza e protocolli sicuri per il controllo della concorrenza e per le operazioni di recovery [AJG99]. Infine, il problema del controllo dell'accesso è stato investigato anche per nuove applicazioni dei sistemi di gestione dati, quali ad esempio i sistemi di data warehouse e data mining, i DBMS multimediali, i sistemi per la gestione di workflow [FT00], i sistemi informativi geografici e le applicazioni web [FT05].

Note bibliografiche

Per una trattazione approfondita delle tematiche di sicurezza rimandiamo il lettore ai due testi di Bishop [Bis02] e [Bis05]. Il modello di Bell e LaPadula è illustrato in [BL75], mentre per il modello di controllo dell'accesso del System R gli articoli di riferimento sono [Fag76] e [GW76]. [FSG+01] è l'articolo di riferimento per quanto riguarda il controllo dell'accesso basato su ruoli. In tale articolo sono anche illustrate le funzionalità amministrative per la specifica ed il mantenimento delle componenti dei vari modelli previsti dallo standard e per effettuare il controllo dell'accesso, che non sono state trattate in questo capitolo. Infine, per SQL il riferimento è la documentazione relativa allo standard SQL:2003 [ISO03].

Esercizi

9.1 Consideriamo le seguenti classi di accesso:

$\mathtt{c}_1$ = `(TS,{Navy,Nuclear})`
$\mathtt{c}_2$ = `(U,{Navy})`
$\mathtt{c}_3$ = `(S,{Navy})`
$\mathtt{c}_4$ = `(C,{Air Force,Nato})`
$\mathtt{c}_5$ = `(TS,{Navy,Army})`

Determinare:

a. Le relazioni che intercorrono tra tali classi.

b. Gli accessi che soggetti con classe di accesso $\mathtt{c}_i$ possono esercitare su oggetti con classe di accesso $\mathtt{c}_j$, $\forall i, j = 1, \ldots, 5$.

9.2 Consideriamo il modello di controllo dell'accesso del System R e la relazione:
`AcquistoProdotti(numProdotto,prezzoUnitario,data,quantità,nome)`
creata da Federica. Supponiamo che:

- al tempo 10 Federica crei una vista che restituisce per ogni prodotto la media del prezzo unitario;
- al tempo 16 Federica conceda tutti i privilegi su tale vista a Matteo con grant option;
- al tempo 20 Matteo conceda tutti i privilegi sulla vista a Michela con grant option;
- al tempo 28 Federica conceda tutti i privilegi sulla vista a Michela;
- al tempo 30 Federica revochi a Matteo i privilegi che gli aveva precedentemente concesso.

a. Scrivere il comando per la definizione della vista e determinare le operazioni che Federica può eseguire su di essa.

b. Scrivere i comandi per concedere e revocare i privilegi sulla vista elencati in precedenza.

c. Determinare se dopo l'operazione di revoca eseguita da Federica, Michela può accedere alla vista e concedere ad altri utenti privilegi su tale vista.

d. Rispondere al quesito al punto (c) considerando lo standard SQL invece del modello di controllo dell'accesso del System R.

9.3 Consideriamo il modello di controllo dell'accesso del System R ed i seguenti comandi:

`luca,50:`[8] `GRANT select on Film to marco WITH GRANT OPTION;`
`marco,52: GRANT select on Film to giovanna WITH GRANT OPTION;`
`giovanna,55: GRANT select on Film to matteo WITH GRANT OPTION;`
`matteo,57: GRANT select on Film to luca;`
`luca,60: REVOKE select on Film FROM marco;`

a. Costruire il grafo delle autorizzazioni relativo alla sequenza di comandi `GRANT` e le corrispondenti entrate nel catalogo `Sysauth`.

b. Eseguire la procedura della Figura 9.5 e determinare lo stato delle autorizzazioni dopo la sua esecuzione.

c. Descrivere il catalogo `Sysauth` dopo l'operazione di revoca al punto (b).

d. Rispondere ai quesiti ai punti (b) e (c) considerando lo standard SQL invece del modello di controllo dell'accesso del System R.

9.4 Consideriamo il modello basato su ruoli ed i ruoli `direttoreVideoteca`, `commesso`, `magazziniere` e `contabile`, tali che `direttoreVideoteca` $\geq$ `commesso`, `direttoreVideoteca` $\geq$ `magazziniere` e `direttoreVideoteca` $\geq$ `contabile`. Supponiamo che il ruolo `commesso` possa effettuare interrogazioni sulla relazione `Film`, mentre il ruolo `contabile` possa inserire tuple nella relazione `Cliente`.

a. Costruire la gerarchia dei ruoli in base alle relazioni sopra descritte.

b. Determinare le autorizzazioni per il ruolo `direttoreVideoteca`.

c. Specificare un vincolo che vieti agli utenti di ricoprire contemporaneamente i ruoli `direttoreVideoteca` e `magazziniere`.

9.5 Con riferimento alla base di dati della videoteca, scrivere i comandi SQL per:

a. Creare il ruolo `regista`.

b. Assegnare al ruolo `regista` il privilegio di eseguire interrogazioni sulle relazioni `Video` e `Film`.

[8] La notazione `u,t` indica il grantor ed il timestamp del comando `GRANT`.

c. Abilitare Giovanni a ricoprire il ruolo `regista` e a concedere a terzi la possibilità di ricoprire tale ruolo.

d. Creare un ruolo `aiutoRegista` sotto-ruolo del ruolo `regista`.

e. Revocare al ruolo `regista` il privilegio di eseguire interrogazioni sulla relazione `Film`.

f. Revocare a Giovanni la possibilità di concedere ad altri l'abilitazione a ricoprire il ruolo `regista`.