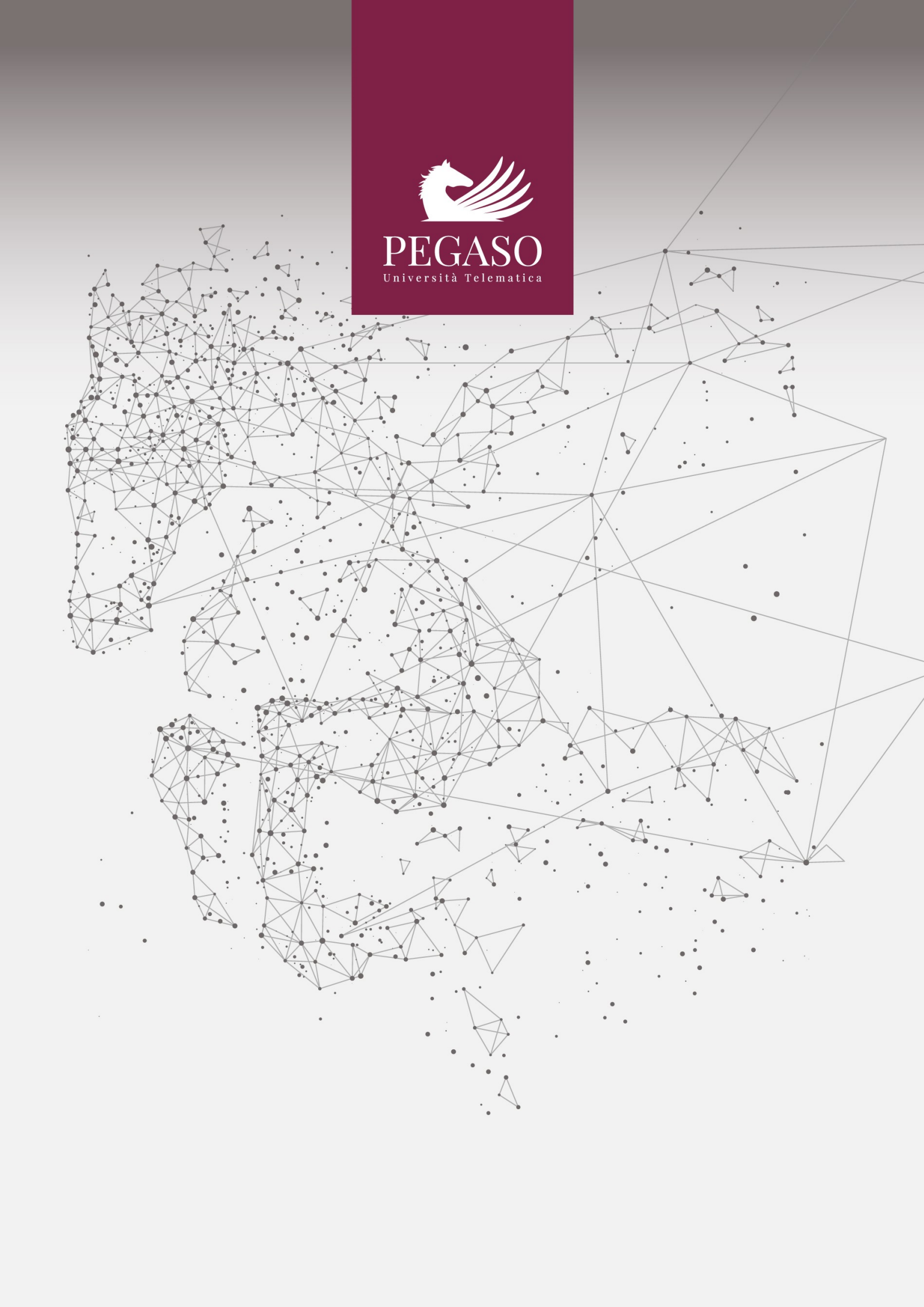
PEGASO
Università Telematica

Indice

1. Introduzione

In questa unità didattica di apprendimento, si studiano le viste ovvero quelle particolari tipologie di relazioni che mostrano agli utenti i dati a seconda di come vengono richiesti. In particolare, si spiegano le varie tipologie di viste possibile, quelle permanenti sul database, Viste Materializzate e quelle residenti in memoria centrale in modalità virtuale. Alcuni esempi aiutano a chiarire i concetti nei vari capitoli della presente dispensa.

2. Le viste: introduzione

Tra i livelli di astrazione di una base di dati c'è il livello esterno, o vista: quello più vicino all'utente *finale* del database, che corrisponde al modo di vedere i dati da parte dell'utente stesso.

Se il database è relazionale, leggere un insieme di dati avente un significato potrebbe essere complesso, perché potrebbe richiedere eccessivi Join fra tabelle, a discapito quindi dell'efficienza; con una vista è invece possibile semplificare molto la stesura di interrogazioni (query) che leggono le informazioni.

Un altro scopo delle viste potrebbe essere semplificare o potenziare la gestione dei permessi. Ad esempio si potrebbe creare una query che legge solo alcuni dati da una tabella, per poi assegnare il permesso in lettura ad un certo utente sulla vista, ma non sulla tabella di base. In questo modo l'utente non vedrà i dati che non vengono estratti dalla vista[1].

Le viste permettono di analizzare, semplificare e personalizzare la visualizzazione del database per ogni utente. Da un punto di vista fisico esistono diversi modi per implementare una vista. Generalmente i DBMS rielaborano le interrogazioni sulle viste in modo che agiscano sulle tabelle che fanno parte della vista stessa. Una vista può essere composta da una o più tabelle; alcuni DBMS, come MySQL, consentono anche di basare la vista su un'espressione relazionale che non coinvolge alcuna tabella.

La vista è una tabella virtuale contenente dati provenienti da altre tabelle del Data Base e può essere funzione di altre viste purché esista un ordinamento in grado di guidare il calcolo delle relazioni derivate. Le viste possono essere aggiornabili, cioè è possibile eseguire su di esse comandi DML. Non tutti i DBMS supportano però questa possibilità e comunque pongono dei limiti e prevedono opzioni di controllo (cfr. in seguito). Poiché una vista non è altro che un'interfaccia su una o più tabelle, esse andranno comunque a modificare le tabelle sottostanti.

Non tutte le viste sono aggiornabili e scrivibili: dipende dove sono scritti i dati. Se anche tutti i dati contenuti nella vista sono scritti fisicamente nelle tabelle, qualora la vista coinvolga diverse tabelle è necessario che tra esse vi sia una relazione a uno a uno.

[1] https://it.wikipedia.org/wiki/Vista_(basi_di_dati)

Di solito non è una buona idea consentire agli utenti finali di fare qualsiasi cosa sulle tabelle (e quindi si utilizzano delle viste), per i seguenti motivi[2]:

- Sicurezza.
- Prevenzione di errori.
- Rendere i dati comprensibili. I nomi delle tabelle e delle colonne sono spesso lunghi e non parlanti. Una vista e le sue colonne possono avere nomi molto più significativi.
- Performance.

[2] http://databasemaster.it/perche-usare-le-viste/

3. Viste: una prima suddivisione

Come si è detto, può risultare utile mettere a disposizione degli utenti rappresentazioni diverse degli stessi dati. Nel modello relazionale, la tecnica prevista a questo scopo è quella delle *relazioni derivate*, relazioni il cui contenuto è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni). Le Relazioni base invece sono quelle relazioni con contenuto autonomo.

Ciascun utente interagisce quindi con lo schema esterno, rappresentato nei livelli di un database di Figura 1. Infatti, ogni utente vede solo:

- ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto;
- ciò che è autorizzato a vedere (autorizzazioni).

Ciò si può ottenere attraverso:

- Strumenti di programmazione:
    - si può semplificare la scrittura di interrogazioni: espressioni complesse e sotto-espressioni ripetute.
- Utilizzo di programmi esistenti su schemi ristrutturati.

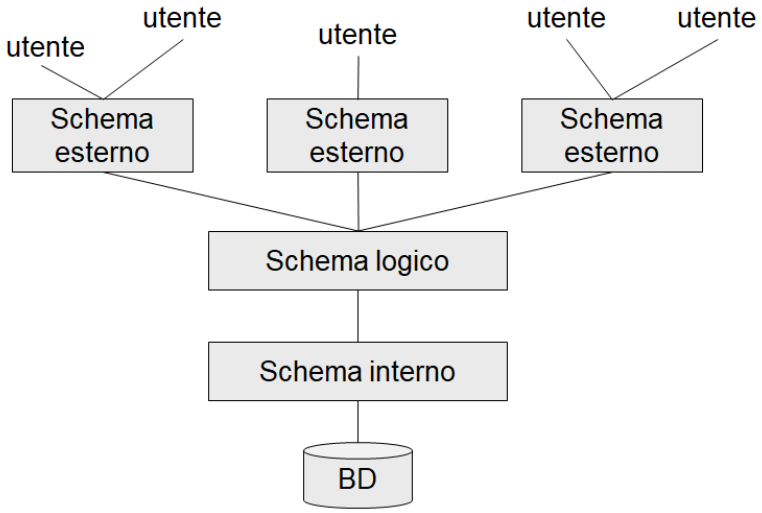

Figura 1: i livelli di un database. L'utente interagisce con lo schema esterno.

Esempio

Si supponga di avere le due relazioni di Figura 2 e di voler avere una relazione che chiamiamo Supervisione, con ogni impiegato il capo. In termini relazionali:

$$\textbf{Supervisione} = \pi_{\textbf{Impiegato, Capo}} (\textbf{Afferenza} \bowtie \textbf{Direzione})$$

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Direzione

Reparto	Capo
A	Mori
B	Bruni

Figura 2: database di esempio formato da 2 relazioni.

Tale relazione può essere costruita attraverso due passaggi poiché non esiste già memorizzata bel database.

Il primo passaggio prevede la seguente operazione relazionale che produce la prima relazione R:

R = Afferenza⋈ Direzione

R= Afferenza ⋈ Direzione ⟹

Impiegato	Reparto	Capo
Rossi	A	Mori
Neri	B	Bruni
Bianchi	B	Bruni

Figura 3: prima relazione.

Secondo passaggio: effettuiamo la proiezione di R ottenendo la relazione finale illustrata in Figura 4.

Supervisione = $\pi_{Impiegato,Capo}$ (R)

Impiegato	Capo
Rossi	Mori
Neri	Bruni
Bianchi	Bruni

Figura 4: relazione Supervisione.

In questo modo abbiamo costruito la relazione con le informazioni che l'utente voleva vedere. Il punto è dove si trova questa tabella, se memorizzata nel database oppure formata a richiesta e messa a disposizione in modo virtuale.

4. Viste: tipologie

In questa sezione si illustrano le varie tipologie di viste:

- VISTA VIRTUALE: calcolata ogni volta che serve. Approccio che consiste nel riscrivere le interrogazioni che usano viste in termini di interrogazioni che usano tabelle base. Ha il vantaggio di non dover salvare i dati delle tabelle virtuali associate alle viste. D'altronde tale approccio risulta inefficiente quando vi sono molte query che si riferiscono alla medesima vista;
- VISTA MATERIALIZZATA: archiviata in modo permanente. Soluzione che consiste nel materializzare le viste, cioè nel calcolare le relative tabelle virtuali, e usare queste tabelle per risolvere le interrogazioni che le usano. Risolve i problemi di efficienza ma ha lo svantaggio di dover mantenere la materializzazione della vista e aggiornarla quando le tabelle di base su cui la vista è definita vengono modificate. Se la vista viene materializzata è necessario ricalcolarne il contenuto ogni volta che le relazioni da cui dipende vengono aggiornate (i dati ivi contenuti vengono aggiornati automaticamente a intervalli regolari dal DBMS. Usate di solito per applicazioni di datawarehousing). La Figura 5 mostra un esempio di uso del modello di vista materializzata per generare un riepilogo delle vendite. I dati delle tabelle ordini, ordinati e clienti vengono combinati per generare una vista che contiene il valore totale delle vendite per ogni prodotto nella categoria Electronics, insieme al conteggio del numero di clienti che hanno acquistato ogni articolo.

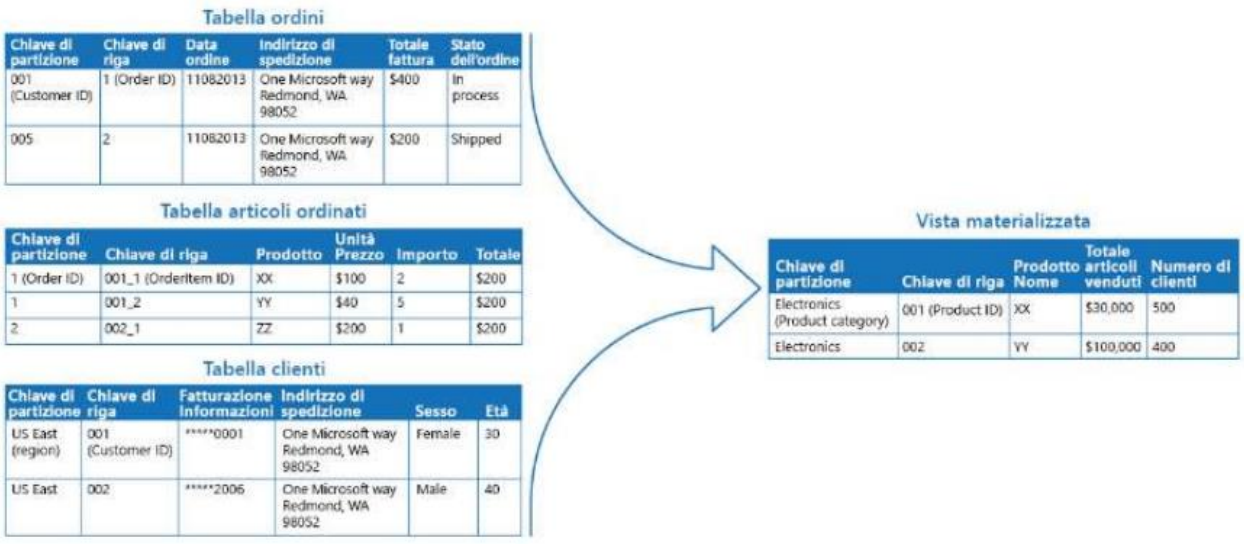

Figura 5: esempio di vista materializzata.

5. Viste: utilità

In questo paragrafo riprendiamo i concetti espressi a pag. 2 estendendoli alla luce di quanto appreso nel paragrafo precedente. Riassumendo, le viste:

- Consentono di personalizzare l'interfaccia utente e di semplificare la scrittura di interrogazioni complesse;
- Possono essere usate come meccanismo di controllo degli accessi. Quindi facilitano la gestione della privatezza dei dati (una delle qualità offerte da un DBMS) e potenziano la gestione dei permessi consentendo di definire una porzione del database accessibile ad un particolare gruppo di utenti. (Ad esempio si potrebbe creare una query che legge solo alcuni dati da una tabella per poi assegnare il permesso in lettura ad un certo utente sulla vista, ma non sulla tabella di base. In questo modo l'utente non vedrà i dati che non vengono estratti dalla vista.);
- Permettono di memorizzare nel DBMS interrogazioni complesse condivise;
- Sono utili per rendere l'interfaccia delle applicazioni indipendente dallo schema logico (INDIPENDENZA LOGICA) e compatibile con versioni precedenti con cui emulare una tabella il cui schema è stato modificato;
- Estendono il potere espressivo del linguaggio SQL (illustrato nelle prossime UDA), consentendo un annidamento delle interrogazioni più sofisticato oppure la realizzazione di campi calcolati che non dipendono totalmente dai dati presenti nel database, facendo fronte a modifiche dello schema logico che comporterebbero una ricompilazione dei programmi applicativi.

6. Viste: operazioni

In questo paragrafo illustriamo le principali operazioni che si possono effettuare sulle viste. Iniziamo dicendo che le operazioni sono eseguite sostituendo alla vista la sua definizione. Riprendiamo l'esempio della relazione *Supervisione* dell'esempio precedente e vediamone come viene eseguita l'operazione di selezione per sapere tutti i dipendenti che hanno per capo il sig. Mori. Ovviamente per costruire tale vista, si parte dalle relazioni base Afferenza e Direzione per costruire la relazione Supervisione:

$$\sigma_{Capo = 'Mori'} \textbf{(Supervisione)}$$

viene eseguita come:

$$\sigma_{Capo = 'Mori'}(\pi_{Impiegato,Capo} \textbf{(Afferenza} \bowtie \textbf{Direzione))}$$

Partendo dal calcolo precedente, otteniamo le trasformazioni illustrate in Figura 6.

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Verdi	A

Direzione

Reparto	Capo
A	Mori
B	Bruni
C	Bruni

Supervisione

Impiegato	Capo
Rossi	Mori
Neri	Bruni
Verdi	Mori

Figura 6: primo passaggio.

Successivamente applicando l'operatore di selezione otteniamo Ottenendo come risultato la relazione di Figura 7. Quindi abbiamo creato una vista a partire da due relazioni base: Afferenza e Direzione e applicando diversi operatori relazionali.

Impiegato	Capo
Rossi	Mori

Figura 7: relazione finale.

Date la vista *Supervisione*, vogliamo inserire, nella vista, il fatto che Lupi ha come capo Bruni, oppure che Belli ha come capo Falchi; come facciamo? Il problema è che, come sappiamo, la vista permette di costruire una relazione che non è memorizzata nel database (non è cioè una vista materializzata). Quindi si tratta di aggiornare una vista. Ciò si ottiene con le seguenti regole:

- Modificare le relazioni di base in modo che la vista, "ricalcolata" rispecchi l'aggiornamento;
- L'aggiornamento sulle relazioni di base corrispondente a quello specificato sulla vista deve essere univoco. In generale però non è univoco!

  Ne consegue che ben pochi aggiornamenti sono ammissibili sulle viste.

7. Conclusioni

Le viste possono essere aggiornabili, cioè è possibile eseguire su di esse comandi DML. Non tutti i DBMS però supportano questa possibilità. Poiché una vista non è altro che un'interfaccia su una o più tabelle, le modifiche andranno a variare le tabelle sottostanti. Non tutte le viste sono aggiornabili e scrivibili. Si parla di viste "updatable" (sulle quali cioè si può eseguire UPDATE e DELETE) solo quando il DBMS è in grado di stabilire una mappatura inversa tra i record presenti nella vista e quelli nelle tabelle base. Si parla di viste "*insertable*" (sulle quali si può eseguire INSERT) quando il DBMS è in grado di inserire il record nella tabella corretta. Ad esempio non è "insertable" una vista che mostra i valori massimi di una certa tabella o che raggruppa i record con una clausola di gruppo (cfr. prossima uda), perché tali dati non sono fisicamente scritti su una qualche tabella, ma rielaborati tramite una query. Se anche tutti i dati contenuti nella vista sono scritti fisicamente nelle tabelle, qualora la vista coinvolga diverse tabelle è necessario che tra esse vi sia una relazione a uno a uno.

Bibliografia

- Atzeni P., Ceri S., Fraternali P., Paraboschi S., Torlone R. (2018). Basi di Dati. McGraw-Hill Education.
- Batini C., Lenzerini M. (1988). Basi di Dati. In Cioffi G. and Falzone V. (Eds). Calderini. Seconda Edizione.