Esercitazione 3

Temi: query su più tabelle (join).

1. Query su più tabelle, ovvero: il join

Supponiamo di voler memorizzare più dati per ciascun fornitore, per esempio la *citta* e l'*indirizzo*. Si potrebbero aggiungere due campi alla tabella *merce*:

merce(nome,varieta,quantita,fornitore,cittaFornitore,indirizzoFornitore,tipo)

...ma questo pone dei problemi. Innanzitutto la struttura della tabella comincia ad essere assai complicata; in secondo, e più importante, luogo, c'è **ridondanza** nei dati. Immaginate di avere 100 tipi diversi di merce forniti tutti dal fornitore Terzi, via dell'Acquedotto Romano 6/canc., Verona. Avremmo questo dato ripetuto per 100 volte nella nostra tabella! Il problema è che stiamo mescolando in una sola tabella dati relativi a due entità ben distinte fra loro: le merci ed i fornitori.
Molto meglio avere *due* tabelle:

merce(nome,varieta,quantita,fornitore,tipo)

fornitore(nome,indirizzo,citta)

...e poi stabilire una relazione uno-a-molti fra *fornitore.nome* e *merce.fornitore*. In sostanza la tabella *fornitore* elenca tutti i fornitori una volta sola (grazie al fatto che {*nome*} è chiave in *fornitore*); e tutte le volte che un dato del fornitore relativo ad una merce è richiesto, dovremo fare una query incrociata, cioè su più tabelle. L'operazione coinvolta è il join.
Provate a realizzare le seguenti query:

1. il solo prodotto cartesiano fra le due tabelle. Quante colonne, e quante righe ha?
2. qual è l'indirizzo del fornitore di mele (di qualsiasi *varieta* di mele)?
3. da quali città arriva la verdura (usate il campo *tipo*)?
4. cosa arriva da Savona?
5. e quale merce arriva da Savona *oppure* da Genova? Visualizzare solo *nome* e *varieta* della merce.

1.1 Concetti associati

Due tabelle A e B sono legate da una **relationship** quando i valori di un campo di A sono legati a quelli di un campo di B; quando uno dei due campi è chiave (cioè è univoco) si parla di **relazione uno-a-molti**; se lo sono entrambi, si parla di **relazione uno-a-uno**[1].
Mettiamo vi sia una relazione uno-a-molti fra A.alfa e B.beta; allora, tutti i valori che appaiono in B.beta *devono* apparire in A.alfa (ma non necessariamente viceversa); la tabella A funge da "indice di riferimento" per la tabella B.

[1] Non confondete questo tipo di "relazioni", che preferisco indicare col termine inglese *relationship*, con gli oggetti matematici detti relazioni, che sono sottoinsiemi del prodotto cartesiano di insiemi.

Se provate a inserire una valore in B.beta che non appare in A.alfa, Access si blocca. Provate per esempio a inserire una merce il cui fornitore (*merce.fonitore*) non appaia nella tabella dei fornitori (*fornitore.nome*)...
È possibile inoltre forzare vincoli di integrità fra tabelle, qualora esista una relationship fra loro:

- **vincoli di aggiornamento a cascata**: se cambio *fornitore.nome*, tutti i relativi campi *merce.fornitore* verranno automaticamente cambiati (provare per credere!);
- **vincoli di cancellazione a cascata**: se tento di cancellare un *fornitore.nome*, Access si ferma e mi chiede cosa deve fare delle merci fornite da quel fornitore.

Le relationship sono realizzate mediante il **join**. Il join è: **prodotto cartesiano** più **selezione**. Creiamo un'unica grossa tabella con tutti i dati di entrambe le tabelle coinvolte (cioè il prodotto cartesiano) e poi selezioniamo proprio le righe che verificano il vincolo della relationship: A.alfa=B.beta.

1.2 Soluzioni

1. il prodotto cartesiano in SQL si indica semplicemente con la virgola:
   ```
   SELECT *
   FROM fornitore,merce;
   ```
   Il numero di colonne è la somma delle colonne delle due tabelle coinvolte: 5+3=8, e notate che siccome in entrambe le tabelle c'è un campo *nome*, esso appare col nome della tabella davanti: *merce.nome* e *fornitore.nome*. Tutti gli altri appaiono semplicemente col loro nome.
   Il numero di righe è il prodotto delle righe delle due tabelle coinvolte: 9*4=36... come da definizione algebrica!
2. ```
   SELECT fornitore.indirizzo
   FROM fornitore INNER JOIN merce
          ON fornitore.nome = merce.fornitore
   WHERE merce.nome=mela;
   ```
   Il che equivale, in algebra, a
   $\pi_{fornitore.indirizzo}(\sigma_{merce.nome=mela}(merce \bowtie_{\substack{fornitore.nome= \\ merce.fornitore}} fornitore))$
   Notate che, usando l'interfaccia grafica di Access, le cose si semplificano assai; in effetti il solo fatto che due tabelle appaiano nel pannello grigio, e che sia visualizzata una relazione uno-a-molti fra di loro, forza l'uso della join in modo "trasparente".
3. ```
   SELECT fornitore.citta
   FROM fornitore INNER JOIN merce
            ON fornitore.nome = merce.fornitore
   WHERE merce.tipo=verdura;
   ```
4. ```
   SELECT merce.*
   FROM fornitore INNER JOIN merce
             ON fornitore.nome = merce.fornitore
   WHERE fornitore.citta=savona;
   ```
5. come sopra, ma con un pizzico di algebra Booleana (l'uso dell'operatore OR):
   ```
   SELECT fornitore.citta, merce.nome, merce.varieta
   FROM fornitore INNER JOIN merce
   ```
   ```

```
          ON fornitore.nome = merce.fornitore
WHERE fornitore.citta=savona OR fornitore.citta=genova;
```