Esercitazione 4

Temi: ragionamenti vari e query complesse su un database parecchio strutturato.

L'archivio musicale

Proviamo a realizzare un database più complesso. Il database conterrà 5 tabelle variamente collegate fra loro, contenenti dati relativi ad un archivio musicale.

SPECIFICA: vogliamo essere in grado di memorizzare i dati di ciascun *album* nel nostro archivio; ogni album è prodotto da un *produttore* ed interpretato da un *interprete*; ciascun interprete e ciascun produttore sono associati ad una *nazione*. Infine, ogni *album* contiene un insieme di brani musicali (la tabella si chiama *brano*, coerentemente con le altre).

PROGETTAZIONE: strutturiamo il nostro database come segue. Le tabelle sono queste:

- *nazione(nome)*
- *interprete(nome,nazione)*
- *produttore(nome,indirizzo,nazione)*
- *album(idalbum,titolo,anno,produttore,prezzo,interprete)*
- *brano(titolo,album,posizione,durata)*

Al solito, le chiavi sono sottolineate; imponiamo i relativi vincoli di chiave (unicità, non-nullità) su ciascuna tabella. Notate come la progettazione rifletta le esigenze "intuitive" della specifica; ad esempio,

- la chiave della tabella *album* è la coppia {*titolo,interprete*}, dal momento che presumibilmente non esistono due album al mondo con lo stesso titolo, interpretati dallo stesso interprete. Notate che possono bensì esistere due album con lo stesso titolo ma di due interpreti differenti (quanti gruppi al mondo hanno pubblicato "The best of" oppure "Greatest hits"...) oppure due album con titoli diversi dello stesso interprete (e ci mancherebbe altro);
- la chiave della tabella *brano* è la coppia {*album,posizione*}, dal momento che ogni brano è univocamente individuato data la sua posizione in un certo album. Questo non pregiudica il fatto che possono esserci brani con lo stesso titolo interpretati da autori diversi, naturalmente.

Inoltre, nella tabella *album*, imponiamo che *idalbum* sia univoco e indicizzato, di fatto rendendolo un'ulteriore chiave, indipendente da {*titolo,interprete*} – una scelta intelligente consiste nell'usare il tipo dati *contatore* di Access.

Le *relationship* fra tabelle sono le seguenti:

- *nazioni.nome* è in relazione uno-a-molti con *interpreti.nazione*
- *nazioni.nome* è in relazione uno-a-molti con *produttori.nazione*
- *produttori.nome* è in relazione uno-a-molti con *album.produttore*
- *interpreti.nome* è in relazione uno-a-molti con *album.interprete*
- *album.idalbum* è in relazione uno-a-molti con *brano.album*

Imporre i vincoli di integrità referenziale fra tabelle quando è possibile e verificare che, per quanto ci interessa, non vi siano ridondanze e/o anomalie. Anche queste relationship riflettono le esigenze del progettista: le nazioni, sia degli interpreti che dei produttori, sono indicizzate in una tabella a parte; ogni album ha uno ed un solo produttore e uno ed un solo interprete; ed infine, ad ogni album corrispondono uno o più brani.

INTERROGAZIONE: Una volta popolate decentemente le tabelle del database, provate a realizzare le seguenti query (in ordine crescente di difficoltà):

1. *(query su più tabelle)* quali album di artisti italiani sono prodotti da produttori italiani?
2. *(query su più tabelle)* in generale, quali album sono prodotti nel paese dell'artista?
3. *(query a somma)* qual è il valore totale dell'archivio musicale?
4. *(query a somma con raggruppamento)* e per nazionalità dell'interprete? per esempio: quanto vale tutta la musica italiana che c'è nell'archivio?
5. *(prodotto cartesiano)* quali brani appaiono in più di un disco suonati da artisti diversi (cover)?
6. *(prodotto cartesiano)* e quali invece suonati dallo stesso, per esempio perché riappaiono in una compilation?
7. *(due query in cascata)* chi interpreta il brano più lungo dell'archivio?
8. *(tre query in cascata)* e chi l'*album* più lungo dell'archivio?

Soluzioni e commenti

Facciamo appunto riferimento al database `Esercitazione4.mdb`.

Tabelle e chiavi

Le chiavi vengono facilmente imposte col solito meccanismo di Access, che prevede unicità e non-nullità, sia che la chiave sia su un attributo solo, sia che essa sia su più attributi, come è il caso della tabella *brano*; il fatto che {*album*,*posizione*} sia chiave di *brano* significa che possono bensì esserci due righe con lo stesso *album*, o con lo stesso *posizione*, ma *non* due righe con la stessa coppia {*album*,*posizione*}.

Relazioni fra tabelle

I vincoli di integrità referenziale vengono imposti all'atto della creazione/modifica di una relationship cliccando sui relativi bottoni. Access riconosce automaticamente il "tipo" di relationship (uno-a-uno, uno-a-molti o indefinita) a seconda che entrambi, uno o nessuno dei campi di partenza siano univoci.
Ad esempio, la relazione fra *produttore.nome* e *album.produttore* è immediatamente riconosciuta come uno-a-molti perché *produttore.nome*, essendo chiave della tabella *produttore*, è univoca.
Provate a creare una relazione, per dire, da *produttore.indirizzo* a *album.produttore* e vedrete che il sistema riconosce la relazione come indeterminata, perché né il primo, né il secondo campo sono univoci.

Provate ora a rendere *produttore.indirizzo* univoca in *produttore*, e vedrete che la relazione verrà riconosciuta come uno-a-molti. Il vantaggio delle uno-a-molti è che Access può mantenere i vincoli di integrità automaticamente, se richiesto (e noi lo richiediamo sempre).
L'unica difficoltà è magari che *molti brani fanno parte di un album*, per cui ci vuole una relazione uno-a-molti fra album e brani. Però la chiave di un album è, manifestamente, la {*titolo,interprete*} (assumiamo che nessun artista abbia mai pubblicato due album con lo stesso nome), e infatti abbiamo richiesto proprio quella coppia come chiave primaria.
In Access, stabilire una relationship su attributi multipli è piuttosto farraginoso; per cui, per semplificare le cose, decidiamo di avere *due* chiavi indipendenti nella tabella *album*: una è {*idalbum*} e l'altra è {*titolo,interprete*}. La chiave a singolo attributo {*idalbum*} ci permetterà di indicizzare l'album di ciascun brano in maniera rapida (usando un solo attributo).

Query

Innanzitutto non fatevi spaventare dal fatto che praticamente tutte le query utilizzano il join: ce lo dobbiamo aspettare dal momento che il database comprende un bel po' di tabelle diverse e ognuna di esse è coinvolta in una relationship con qualche altra tabella. Inoltre, abbiamo qui casi di join nidificati, ovvero di query che coinvolgono *tre* tabelle contemporaneamente.

1. quali album di artisti italiani sono prodotti da produttori italiani?
   Usando l'interfaccia grafica di Access è facile: date le tabelle *album*, *produttori* e *interprete* si richiede che i campi *nazione* di *produttore* e *interprete* siano uguali a "italia". L'SQL non è così immediato, invece, poiché la query richiede la presenza di tre tabelle e quindi abbiamo un join dentro un altro join (join nidificato).
2. in generale, quali album sono prodotti nel paese dell'artista?
   Idem come sopra, ma chiediamo che i campi *nazione* di *produttore* e *interprete* siano uguali fra loro. Confrontate l'SQL di questa con quella di sopra: questo è leggermente più semplice.
3. qual è il valore totale dell'archivio musicale?
   Facilissima: si fa la somma del campo *album.prezzo*.
4. e per nazionalità dell'interprete? per esempio: quanto vale tutta la musica italiana che c'è nell'archivio?
   Come sopra, ma aggiungiamo un raggruppamento su *interprete.nazione* di ciascun *album*. L'SQL è leggermente più complesso: c'è un join di mezzo.
5. quali brani appaiono in più di un disco suonati da artisti diversi (cover)?
   Questa e un po' più diffcile. Dovendo confrontare dati relativi alla stessa entità, in questo caso *brano.interprete* di due brani aventi lo stesso *titolo*, creiamo dapprima una join fra *brano.album* e *album.idalbum*, da cui risaliamo all'*interprete*. Quindi facciamo una copia sia di *brano* che di *album*, e rifacciamo lo stesso join fra queste due nuove tabelle; infine, facciamo il prodotto cartesiano delle due tabelle ottenute con i join, chiedendo che *brano.titolo* sia uguale nelle due tabelle, e che però l'interprete sia differente. Provare per credere. L'SQL di questa query è piuttosto complesso: studiatelo attentamente!
6. e quali invece suonati dallo stesso, per esempio perché riappaiono in una compilation?
   Come sopra, ma chiediamo che i due *brano.interprete* siano uguali, che i due *brano.titolo* siano pure uguali, e che invece gli *album.titolo* siano differenti.

7. chi interpreta il brano più lungo dell'archivio?
   Facciamo una query che calcoli il massimo di *brano.durata* – per questo fate sì che Access mostri la riga "totali" nella struttura della query (funzioni aggregate). Ogni query è l'applicazione di uno o più operatori relazionali ad un insieme di tabelle (relazioni), quindi bene o male è essa stessa una tabella, e può venire indirizzata usandone il nome ed il nome dei suoi attributi. Supponiamo di aver chiamato questa query *query7a*, allora possiamo, per risolvere il problema, costruire una seconda query che selezioni il *brano* avente *durata* pari alla massima *durata* trovata, cioè *query7a.MaxDidurata*. Dato il nome del *brano* posso risalire all'*album* cui appartiene, e di lì all'*interprete*.
8. e chi interpreta l'*album* più lungo dell'archivio?
   Leggermente più complessa: stavolta mi serve una query che calcoli la durata totale di ogni singolo *album*, la otteniamo chiedendo la somma di *brano.durata* ma raggruppando i brani per *album.* Chiamiamo questa query *query8a.* Quindi mi serve un'altra query che estragga il massimo da *query8a.SommaDidurata,* chiamiamola *query8b*; infine una terza query che estragga la riga di *query8a* per cui *SommaDidurata* è pari a *query8b.MaxDiSommaDidurata*. Dato il nome dell'*album* posso facilmente risalire all'interprete.