

Algebra e calcolo relazionale

#algebra-relazionale

#procedurali

#ridenominazione

#selezione

#proiezione

#join

#viste

I linguaggi possono essere distinti in:

- **dichiarativi**, specificano le proprietà del risultato("che cosa")
 - calcolo relazionale
 - SQL
 - Query By Example (QBE)
- **procedurali**, specificano le modalità di generazione del risultato ("come")
 - **algebra relazionale**

Algebra relazionale

Insieme di operatori:

- su relazioni
- che producono relazioni
- possono essere composti

Con l'algebra relazionale lavoriamo con tabelle/relazioni e applichiamo operatori sulle stesse per produrre altre tabelle.

Operatori insiemistici

Le relazioni sono degli **insiemi**, con risultati relazioni.

Posso fare l'unione \cup di 2 relazioni con n -uple di entrambe? Sì, a condizione che le 2 relazioni siano definite sullo stesso insieme di attributi (non posso fare $15 \cup 5$).

- **unione** \cup , unisce gli attributi delle tabelle, il risultato è un insieme di n -uple (relazione), i duplicati vengono eliminati
- **intersezione** \cap , con le n -uple uguali tra entrambe le relazioni
- **differenza** $-$

Ridenominazione

Operatore monadico (su una tabella) che **modifica lo schema**, non l'istanza, cambiando il nome di 1 o più attributi.

☰ Ridenominare 2 tabelle

L'unione tra 2 tabelle con attributi "Madre" e "Padre" non è possibile siccome il nome degli attributi è diverso, possiamo tuttavia ridenominare questi

$REN_{genitore \leftarrow padre}(Paternità) \cup REN_{genitore \leftarrow madre}(Maternità)$

Selezione

Operatore monadico (su una sola tabella) che produce un risultato con lo stesso schema dell'operando e contiene una **selezione** delle n -uple che soddisfano un **predicato** (VERO o FALSO).

Semplicemente: prende una *condizione* e ritorna i risultati soddisfacenti la condizione, contenuti nella tabella *operando*.

$$SEL_{Condizione}(Operando)$$

dove *Condizione* è una espressione booleana

≡ **Impiegati che guadagnano più di 50**

$SEL_{\text{stipendio} > 50}(\text{Impiegati})$

≡ **Impiegati che guadagnano più di 50 e lavorano a 'Milano'**

$SEL_{\text{stipendio} > 50 \text{ AND } \text{filiale} = \text{'Milano'}}(\text{Impiegati})$

Proiezione

Decomposizione verticale, operatore ortogonale.

Anche lui operatore monadico, parametrico.

Semplicemente: prende una *lista di attributi* riguardante a una tabella (*operando*) e restituisce solo quelli specificati.

$$PROJ_{ListaAttributi}(Operando)$$

≡ **Cognome e filiale di tutti gli impiegati**

$PROJ_{\text{cognome}, \text{nome}}(\text{Impiegati})$

Una proiezione contiene al più tante *n*-uple quante l'operando e può contenerne di meno.

Se *X* è una superchiave di *R*, allora $PROJ_X(R)$ contiene esattamente tante *n*-uple quante *R*.

Possiamo usare selezione e proiezione insieme, per restituire risultati di una selezione per delle colonne specifiche solo del SELECT:

≡ **Matricola e cognome degli impiegati che guadagnano più di 50**

$PROJ_{\text{matricola}, \text{cognome}}(SEL_{\text{stipendio} > 50}(\text{Impiegati}))$

Non possiamo correlare informazioni presenti in relazioni diverse, nè informazioni in *n*-upla diverse di una stessa relazione.

JOIN

Permette di correlare dati in relazioni diverse.

Cardinalita':

- il join di R_1 e R_2 contiene un numero di *n*-uple:
 - compreso fra 0 e il prodotto di $|R_1|$ e $|R_2|$

- se coinvolge una chiave di R_2 allora il numero di n -uple è:
 - compreso fra 0 e $|R_1|$
- se il join coinvolge una chiave di R_2 e vincolo d'integrità referenziale, allora il numero di n -uple è
 - pari a $|R_1|$

$R_1 \text{ JOIN } R_2$ e' una relazione su $X_1 X_2$ (intesa come unione):

$$\{t \text{ su } X_1 X_2 \mid \text{esistono } t_1 \in R_1 \wedge t_2 \in R_2 \text{ con } t[X_1] = t_1 \wedge t[X_2] = t_2\}$$

Per ogni riga che si trova nella tabella di sinistra, guardiamo quante di righe hanno un attributo in comune con la tabella di destra e uniamo nel caso in cui questa incidenza esista.

JOIN NATURALE

Immaginiamo di avere una due tabelle e volessimo unire le due, seguendo un criterio: `numero` deve essere contenuto in entrambe.

Possiamo farlo con il **join naturale** dove i miei attributi coincidono su un attributo. Noi non dobbiamo fare nulla, il join e' automatico se l'attributo comune esiste.

JOIN NATURALE

numero	voto
1	25
2	13
3	27
4	28

numero	candidato
1	mario rossi
2	nicola russo
3	mario bianchi
4	remo neri

numero	candidato	voto
1	mario rossi	25
2	nicola russo	13
3	mario bianchi	27
4	remo neri	28

Produce un risultato:

- sull'unione degli attributi degli operandi
- con n -uple costruite ciascuna a partire da una n -upla di ognuno degli operandi

JOIN COMPLETO

Ogni n -upla contribuisce al risultato. Nessuna viene eliminata.

Tuttavia se non troviamo attributi uguali, il join diventa **incompleto**.

JOIN COMPLETO tuttavia vuoto

impiegato	reparto
Rossi	A
Neri	B
Binachi	B

reparto	capo
B	Mori
C	Bruni

impiegato	reparto	capo
-----------	---------	------

JOIN ESTERNO

Estende con *valori NULL* le n -uple che verrebbero tagliate fuori da un join interno, si può fare sulla sinistra, destra o completo: tutte le n -uple dell'argomento di sinistra vengono prese e per gli argomenti di destra, se non ci sono, vanno a NULL (*outer left join*).

- *sinistro* mantiene tutte le n -uple del primo operando, estendendo con NULL se necessario;
- *destro* del secondo operando;
- *completo* su entrambi gli operandi

JOIN LEFT con le tabelle di prima

impiegato	reparto
Rossi	A
Neri	B
Binachi	B

reparto	capo
B	Mori
C	Bruni

impiegati JOIN_{LEFT} reparti

impiegato	reparto	capo
neri	B	mori
bianchi	B	mori
rossi	A	NULL

impiegati JOIN_{RIGHT} reparti

impiegato	reparto	capo
neri	B	mori
bianchi	B	mori

impiegato	reparto	capo
NULL	C	bruni

impiegati JOIN_{FULL} reparti

impiegato	reparto	capo
neri	B	mori
bianchi	B	mori
rossi	A	NULL
NULL	C	bruni

JOIN E PROIEZIONI

Se prendessimo due tabelle e facessimo INNER JOIN (JOIN NATURALE), con una successiva PROIEZIONE, non e' detto che si ritorni alla tabella originale. Quando il JOIN non e' completo, allora accade.

$$\text{PROJ}_{X_1}(R_1 \text{ JOIN } R_2) \subseteq R_1$$

Se facessimo l'operazione inversa (prima due PROIEZIONI e poi il JOIN), otterremmo piu' n -uple di quelle di partenza.

$$(\text{PROJ}_{X_1}(R)) \text{ JOIN } (\text{PROJ}_{X_2}(R)) \supseteq R$$

PRODOTTO CARTESIANO

Sarebbe un JOIN NATURALE su relazioni senza attributi in comune.

Contiene sempre un numero di n -uple pari al prodotto delle cardinalita' degli operandi (tutte combinabili).

Impiegati		Reparti	
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B		

Impiegati JOIN Reparti			
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Rossi	A	B	Bruni
Neri	B	A	Mori
Neri	B	B	Bruni
Bianchi	B	A	Mori
Bianchi	B	B	Bruni

Di solito viene susseguito con un SELECT se vogliamo dargli un senso:

$$\text{SEL}_{\text{condizione}}(R_1 \text{ JOIN } R_2)$$

L'operazione viene chiamata *theta-join*, JOIN con condizione:

$$R_1 \text{ JOIN}_{\text{condizione}} R_2$$

Se l'operazione di confronto (condizione) nel theta-join e' sempre l'uguaglianza (=) allora si parla di *equi-join*:

Impiegati		Reparti	
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B		

Impiegati JOIN _{Reparto=Codice} Reparti			
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B	B	Bruni

VISTE

Sono rappresentazioni dei dati per *schema esterno*.

- relazioni derivate, cui contenuto è funzione di altre relazioni;
- relazioni di base, a contenuto autonomo.

Ci sono 2 tipi di relazioni derivate:

- viste materializzate, funzionano molto bene fintanto che le relazioni rimangono costanti nel tempo, ovvero non cambiano troppo frequentemente (che non vedremo);
- relazioni virtuali (viste), supportate da tutti i DBMS, un'interrogazione su una vista viene eseguita "ricalcolando" la vista;

Rimpiazzare pezzi grossi in un nome che mi dà significato, aiuta nella comprensione delle interrogazioni da farsi. Nello schema esterno ogni utente vede solo:

- ciò che gli interessa;
- ciò che è autorizzato a vedere.

Capi := Imp

PROJ_{Imp.Matr, Imp.Nome, Imp.Stip, Capi.Matr, Capi.Nome, Capi.Stip}

(**SEL**_{Imp.Stip > Capi.Stip} (

Capi JOIN_{Capi.Matr = Capo} (**Sup JOIN**_{Imp = Imp.Matr} **Imp**)))

up to: 10-05