

# 29. Il comando select e l'algebra relazionale

## parte terza

database

## il comando select e l'algebra relazionale

Abbiamo visto dunque che SQL gestisce database relazionali ed è in grado di simulare il funzionamento dei comandi dell'algebra relazionale.

Occorre comunque mettere in evidenza un'importante distinzione tra il comportamento di SQL e le regole del modello relazionale.

SQL consente ad una tabella prodotta da un'operazione di **SELECT** di avere due o più tuple identiche in tutti i valori dei loro attributi (ciò può avvenire ad es. se la lista dei campi non comprende una chiave candidata); in generale quindi **una tabella prodotta da SQL non è un insieme di tuple**, dal momento che in un insieme per definizione non possono esserci due elementi uguali, **piuttosto si tratta di un *multinsieme*** (talvolta chiamato *sacca* o *bag*) **di tuple**.

## il comando select e l'algebra relazionale

Per questo motivo DBMS come *MySQL* (basati sull'uso di SQL) consentono anche la realizzazione di tabelle senza la definizione di una chiave primaria, permettendo in questo modo l'eventuale presenza di tuple uguali nella tabella stessa.

Esiste comunque il predicato **DISTINCT** che insieme col **SELECT** permette eliminare dalla tabella risultante tutte le eventuali righe ripetute, ottenendo così una relazione corretta dal punto di vista della sua definizione insiemistica.

# il comando select e l'algebra relazionale

La seguente query seleziona i codici dei dipartimenti registrati nella tabella *Personale* riportando ogni codice una sola volta:

```
SELECT DISTINCT CodDip  
FROM Personale;
```

Senza specificare il predicato **DISTINCT** ogni codice prodotto sarebbe stato ripetuto tante volte quanti sono i dipendenti che lavorano in quel dipartimento.

SQL ammette anche il predicato **ALL** che non va specificato (in quanto di default) e mostra tutte le righe che soddisfano le condizioni imposte dalla clausola **WHERE**.

# il comando select e l'algebra relazionale

Estrarre le città dei dipendenti con qualifica di tecnico, facendo comparire ogni città al più una volta sola:










```
SELECT DISTINCT Qualifica, Città
FROM Personale
WHERE Qualifica = 'Tecnico';
```

**NB:** Senza l'uso del predicato **DISTINCT**, sarebbero potute comparire più righe uguali, là dove ci fosse più di un tecnico in una stessa città; ad es. nel nostro caso sono presenti due tecnici in Catanzaro

+ Opzioni

	Qualifica	Città
<input type="checkbox"/>  Modifica  Copia  Elimina	Tecnico	Catanzaro

+ Opzioni

	Qualifica	Città
<input type="checkbox"/>  Modifica  Copia  Elimina	Tecnico	Catanzaro
<input type="checkbox"/>  Modifica  Copia  Elimina	Tecnico	Catanzaro
<input type="checkbox"/>  Modifica  Copia  Elimina	Tecnico	Catanzaro


# il comando select e l'algebra relazionale

Le righe che risultano da un'interrogazione vengono restituite secondo un ordinamento determinato automaticamente dal sistema a meno che non si specifichi esplicitamente un criterio di ordinamento.

Per ottenere la matricola, il cognome e lo stipendio degli impiegati del dipartimento D2 in ordine (crescente) di matricola, si deve formulare la seguente query:

```
SELECT Matricola, Cognome, Qualifica  
FROM Personale  
WHERE CodDip='D2'  
ORDER BY Matricola;
```

La clausola **ORDER BY** è in genere l'ultimo elemento di un comando SQL.



**NB:** Negli ordinamenti, il valore NULL compare all'inizio delle sequenze crescenti e alla fine delle sequenze decrescenti.

# il comando select e l'algebra relazionale

Il criterio di ordinamento può essere espresso su più campi in ordine crescente («**ASC**») oppure decrescente («**DESC**»); se quest'ultima opzione non viene esplicitata, come **ordinamento predefinito** viene assunto quello **crescente**.

Ad es. "**ORDER BY** stipendio **DESC**, nominativo" fornirà i dati ordinati dallo stipendio più grande al più piccolo e a parità di esso, i nominativi in ordine alfabetico.

Per ottenere la matricola, il cognome e lo stipendio degli impiegati del dipartimento D1, in ordine ascendente di *Città* e, nell'ambito della stessa città, discendente di *Stipendio*, si ricorre alla seguente interrogazione:

```
SELECT Città, Matricola, Cognome, Stipendio  
FROM Personale  
WHERE CodDip='D1'  
ORDER BY Città, Stipendio DESC;
```

## il comando select e l'algebra relazionale

Un altro operatore reso disponibile dal linguaggio SQL per la formulazione delle condizioni è il **BETWEEN**; esso permette di confrontare il valore di un campo con un intervallo di valori nella forma:

`<campo> BETWEEN <valore_iniziale> AND <valore_finale>`

equivalente a:

`<campo> >= <valore_iniziale> AND <campo> <= <valore_finale>`

**NB:** Ovviamente, è anche possibile anteporre al **BETWEEN** l'operatore booleano **NOT** per valutare la condizione opposta, cioè controllare se il valore non rientra nell'intervallo specificato.



# il comando select e l'algebra relazionale

Volendo estrarre tutti i dati dei dipendenti nati tra il 1960 e il 1969 si può usare:

```
SELECT *  
FROM Personale  
WHERE DataNasc BETWEEN '1960-01-01' AND '1969-12-31';
```

oppure utilizzando la funzione **YEAR()** che estrae l'anno da un campo di tipo data:

```
WHERE YEAR(DataNasc) BETWEEN 1960 AND 1969;
```

**NB:** Le costanti data usate per delimitare l'intervallo temporale sono state espresse come stringhe nel formato AAAA-MM-GG (quattro cifre per l'anno, due per il mese e due per il giorno separate dal simbolo «-»), mentre nel caso dell'anno i valori di confronto sono stati espressi come numeri interi.

# il comando select e l'algebra relazionale

Il risultato di una query è sempre una tabella virtuale che non viene memorizzata nel database. È comunque possibile (se si hanno i necessari privilegi come utente del DBMS) memorizzare la tabella ottenuta nel database come nuova tabella:

```
CREATE TABLE DipendentiAnni60  
SELECT *  
FROM Personale  
WHERE DataNasc BETWEEN '1960-01-01' AND '1969-12-31';
```

oppure utilizzando la funzione `YEAR()` che estrae l'anno da un campo di tipo data:

La query mostrata estrae tutti i dipendenti nati negli anni 60 (tra il 1 gennaio 1960 ed il 31 dicembre 1969) e crea una nuova tabella `DipendentiAnni60` contenente il risultato di questa query.

# il comando select e l'algebra relazionale

Un'ultima considerazione la facciamo a proposito dell'operazione di *join*.

Sappiamo che il comando **SELECT** può operare su più tabelle, se ne indichiamo i nomi (separati da virgole) dopo la parola chiave **FROM**.

Si realizza di fatto l'operazione di congiunzione (**JOIN**) delle tabelle, quando la clausola **WHERE** impone l'uguaglianza dei valori dei campi che si corrispondono nelle tabelle coinvolte.

L'esempio sul quale stiamo lavorando presenta un'associazione 1:N tra le tabelle *Dipartimenti* e *Personale*, associazione che si concretizza sull'uguaglianza dei valori dell'attributo *CodDip* comune alle due tabelle, chiave primaria in *Dipartimenti* e chiave esterna in *Personale*.

# il comando select e l'algebra relazionale

La query seguente, già vista prima, mostra ad es. un elenco con i cognomi dei nostri dipendenti e le città in cui questi lavorano; partendo dal prodotto cartesiano tra le tabelle *Dipartimenti* e *Personale*, la clausola **WHERE** permette di selezionare le sole righe per le quali corrispondono i valori dei campi *CodDip*.

Realizza il prodotto cartesiano tra le tabelle *Dipartimenti* e *Personale*.

```
SELECT P.Cognome, P.Nome, D.Citta
FROM Dipartimenti AS D, Personale AS P
WHERE D.CodDip = P.CodDip;
```

Seleziona le righe del prodotto cartesiano tra le tabelle *Dipartimenti* e *Personale* per le quali corrispondono i valori dei campi *CodDip*, realizzando di fatto una join tra di esse.

+ Opzioni

Cognome	Nome	Citta
Rossi	Mario	Catanzaro
Bianchi	Carlo	Catanzaro
Verdi	Giovanni	Cosenza
Verdi	Franco	Cosenza
Rossi	Carlo	Cosenza
Segni	Carlo	Cosenza
Palmi	Pedi	Cosenza
Gialli	Lorenzo	Reggio Calabria
Rosati	Paola	Reggio Calabria
Franco	Marco	Reggio Calabria
Tedesco	Matteo	Catanzaro

# il comando select e l'algebra relazionale

Un'azienda di articoli da ferramenta registra gli ordinativi dei suoi clienti in Calabria.

Ciascun cliente viene identificato da un codice formato da due parti, l'indicazione della provincia di appartenenza e un numero progressivo, per cui la chiave primaria della tabella *Cliente* è formata da due attributi: *Prov* e *Codice* (ogni codice può ripetersi in province diverse, ma la coppia codice + provincia è univoca per ciascun cliente).

Vediamo un possibile modello relazionale:

La chiave primaria della tabella *Cliente* è formata da due campi: *Codice* + *Prov*.

*Cliente* (*Codice*, *Prov*, *Nominativo*, *Indirizzo*, *Città*, *Telefono*);

*Ordinativi* (*Codice*, *ProvCliente*, *CodCliente*, *Descrizione*, *Data*, );

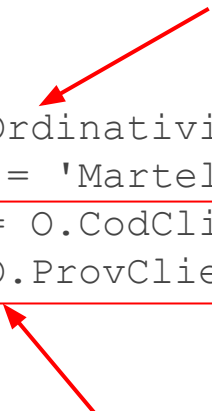
Codice del cliente e provincia vengono riportati nella tabella *Ordinativi* come chiave esterna.

# il comando select e l'algebra relazionale

Volendo ottenere un elenco dei clienti che hanno ordinato un martello, dobbiamo prima eseguire il join delle tabelle sull'uguaglianza dei campi provincia e codice cliente, poi la selezione per descrizione del prodotto = martello:

**NB:** anche se non c'è possibilità di ambiguità, si è scelto comunque di indicare l'alias delle tabelle coinvolte.

```
SELECT C.Nominativo  
FROM Cliente AS C, Ordinativi AS O  
WHERE O.Descrizione = 'Martello'  
AND C.Codice = O.CodCliente  
AND C.Prov = O.ProvCliente;
```



La condizione di join verifica l'uguaglianza dei valori in entrambi i campi (codice cliente e provincia) chiave primaria in Cliente e chiave esterna in Ordinativi

# il comando select e l'algebra relazionale

Riprendiamo adesso un esempio visto in precedenza:

*Dipartimenti* (CodDip, Nome, Indirizzo, Città);

*Personale* (Matricola, CodDip, Cognome, Nome, DataNasc, Qualifica, Stipendio, Ufficio, Città);

Partendo dallo stipendio annuale (campo *Stipendio*), abbiamo ottenuto cognome, nome e stipendio mensile dei dipendenti con qualifica di programmatore riportati nella tabella *Personale*, tramite la query seguente:

```
SELECT Cognome, Nome, Stipendio/12 AS Mensile
FROM Personale
WHERE Qualifica = 'Programmatore'
```

Possiamo notare che gli importi mensili presentano sei cifre decimali, mentre sarebbe meglio averne solo due.

				Cognome	Nome	Mensile			
<input type="checkbox"/>		Modifica		Copia		Elimina	Bianchi	Carlo	2700.000000
<input type="checkbox"/>		Modifica		Copia		Elimina	Franco	Marco	2500.000000
<input type="checkbox"/>		Modifica		Copia		Elimina	Tedesco	Matteo	2666.666667
<input type="checkbox"/>		Modifica		Copia		Elimina	Vacchi	Marco	2916.666667
<input type="checkbox"/>		Modifica		Copia		Elimina	Segni	Carlo	2400.000000
<input type="checkbox"/>		Modifica		Copia		Elimina	Ponsi	Mario	3166.666667

# il comando select e l'algebra relazionale

Tramite la funzione `ROUND()` possiamo arrotondare un numero decimale in modo da presentare un numero prefissato di cifre dopo la virgola:

```
SELECT Cognome, Nome, ROUND(Stipendio/12, 2) AS Mensile
FROM Personale
WHERE Qualifica = 'Programmatore'
```

La funzione `ROUND()` presenta la seguente sintassi:

`ROUND(numero, decimali)`

Numero da arrotondare

Opzionale: numero di cifre decimali; se assente la funzione restituisce il numero intero senza cifre decimali.

+ Opzioni				Cognome	Nome	Mensile
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
<input type="checkbox"/>		Modifica		Copia		Elimina
				Bianchi	Carlo	2700.00
				Franco	Marco	2500.00
				Tedesco	Matteo	2666.67
				Vacchi	Marco	2916.67
				Segni	Carlo	2400.00
				Ponsi	Mario	3166.67










# il comando select e l'algebra relazionale

Un'altra funzione utile è la `FLOOR()` la quale restituisce il numero intero più grande che risulta minore o uguale al numero fornito come argomento.

Ad es. per ottenere un elenco con matricola, cognome ed età dei dipendenti afferenti al dipartimento D1, possiamo sfruttare la query seguente:

```
SELECT Matricola, Cognome,  
        FLOOR(DATEDIFF(CURDATE(), Personale.DataNasc)/365) AS Età  
FROM Personale  
WHERE CodDip = 'D1'
```

+ Opzioni

					Matricola	Cognome	Età		
<input type="checkbox"/>		Modifica		Copia		Elimina	AB001	Rossi	33
<input type="checkbox"/>		Modifica		Copia		Elimina	AB002	Bianchi	29

# il comando select e l'algebra relazionale

Altre funzioni utili:

`CEIL(numero)` : restituisce il numero intero più piccolo che risulta maggiore o uguale al numero fornito come argomento;

`TRUNCATE(numero, decimali)` : restituisce il numero fornito come argomento con tante cifre decimali quante sono quelle specificate dal secondo argomento, senza alcun arrotondamento.