SQL Data Manipolation Language

Lavorare su più tabelle

DML – Lavorare su più tabelle

Molto spesso i dati non sono tutti in una sola tabella, anche a fronte della necessità di normalizzazione.

Sono pertanto necessari metodi per poter accedere a più tabelle creando query che possono operare sui dati completi.

Sql mette a disposizione diverse modalità. Vedremo:

- Prodotto cartesiano
- Join
- Subquery
- Operatori insiemistici (union)

DML – Tabelle di riferimento

Negli esempi che seguono prenderemo in considerazione un semplice database composto da due tabelle:

Dipendenti

CodDip	Cognome	Nome	Ruolo	Stipendio	Sede
1	Rossi	Mario	Medico	5000	S01
2	Bianchi	Lucia	Infermiere	1500	NULL
3	Verdi	Carlo	Amministrativo	1500	S01

Sedi

CodSede	Città
S01	Roma
S02	Milano

DML – Prodotto cartesiano

Il prodotto cartesiano di due tabelle genera un set di dati nel quale ogni riga della prima tabella è combinata con ogni riga della seconda tabella. Sul set di dati così generato è possibile effettuare tutte le operazioni permesse su singola tabella (seleziona campi, WHERE, ORDER BY, GROUP BY, HAVING….). Il prodotto cartesiano si effettua specificando tutte le tabelle nella clausola FROM, separate da virgola

```
SELECT * FROM Dipendenti, Sedi
```

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.CodSede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	S01	Roma
1	Rossi	Mario	Medico	5000	S01	S02	Milano
2	Bianchi	Lucia	Infermiere	1500	NULL	S01	Roma
2	Bianchi	Lucia	Infermiere	1500	NULL	S02	Milano
3	Verdi	Carlo	Amministrativo	1500	S01	S01	Roma
3	Verdi	Carlo	Amministrativo	1500	S01	S02	Milano

DML – Pseudonimi

Nella selezione dei campi, quando non si usa *, si deve indicare anche il nome della tabella a cui ci si riferisce. Per esempio

```
SELECT Dipendenti.CodDip, Dipendenti.Stipendio, Sedi.Città
FROM Dipendenti, Sedi
```

Dipendenti.CodDip	Dipendenti.Stipendio	Sedi.Città
1	5000	Roma
1	5000	Milano
2	1500	Roma
2	1500	Milano
3	1500	Roma
3	1500	Milano

DML – Pseudonimi

E' inoltre possibile l'utilizzo di * solamente su alcune tabelle. Per esempio:

```
SELECT Dipendenti.*, Sedi.Città
FROM Dipendenti, Sedi
```

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	Roma
1	Rossi	Mario	Medico	5000	S01	Milano
2	Bianchi	Lucia	Infermiere	1500	NULL	Roma
2	Bianchi	Lucia	Infermiere	1500	NULL	Milano
3	Verdi	Carlo	Amministrativo	1500	S01	Roma
3	Verdi	Carlo	Amministrativo	1500	S02	Milano

DML – Pseudonimi

Per rendere più compatta la scrittura delle query è possibili usare pseudonimi al posto dei nome completi delle tabelle. Per esempio:

```
SELECT D.CodDip, D.Stipendio, S.Città
FROM Dipendenti D, Sedi S
```

NB: Rimane sempre possibile il renaming delle colonne tramite l'operatore AS.

DML – Join

Molto spesso l'obiettivo è quello di ottenere un sottoinsieme delle righe generate con il prodotto cartesiano. Si può agire con la clausola WHERE. Per esempio:

```
SELECT Dipendenti.*, Sedi.Città
FROM Dipendenti D, Sedi S
WHERE D.Sede = S.CodSede
```

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	Roma
3	Verdi	Carlo	Amministrativo	1500	S01	Roma

Essendo casi molto frequenti il linguaggio SQL mette a disposizione una famiglia di operatori denominati JOIN, che permettono di effettuare anche operazioni non ottenibili tramite WHERE.

DML – Join

Gli operatori di JOIN permettono di generare un set di dati ottenuto combinando, in modi differenti in base alla tipologia di join utilizzata, i dati provenienti da due tabelle.

Alcune tipologie di Join sono ottenibili anche con prodotto cartesiano + Where. Altri invece sono ottenibili solamente con il Join.

Sul set di dati così generato è possibile effettuare tutte le operazioni permesse su singola tabella (seleziona campi, WHERE, ORDER BY, GROUP BY, HAVING….).

Vedremo:

- Inner Join (e self Join)
- Left Join
- Right Join
- Full outer Join

DML – Inner Join

L'operatore di Inner Join prevede di specificare una condizione di uguaglianza tra attributi delle due tabelle e genera un set di dati nel quale sono incluse solamente le righe nelle quali l'uguaglianza è soddisfatta.

```
SELECT *
FROM Dipendenti D INNER JOIN Sedi S ON D.Sede = S.CodSede
```

È equivalente a:

```
SELECT *
FROM Dipendenti D, Sedi S
WHERE D.Sede = S.CodSede
```

NB: Le righe con valore NULL sui campi di Join non vengono considerati (NULL non è un valore)

DML – Self Join

L'inner Join può essere fatto anche sulla stessa tabella. In tal caso si chiama Self Join e richiede, necessariamente l'utilizzo di pseudonimi.

Si consideri per esempio una tabella con le relazioni Genitore-Figlio e le seguente query:

Genitore	Figlio
Luca	Anna
Maria	Alberto
Giorgio	Luca
Michele	Maria

```
SELECT G1.Genitore A Nonno, G2.Figlio As Nipote
FROM Genitori G1 INNER JOIN Genitori G2 ON
G1.Figlio = G2.Genitore
```

Nonno	Nipote
Giorgio	Anna
Michele	Alberto

DML – Left Join

Il Left Join prevede che il set dei risultati contenga sempre e comunque tutte le righe della tabella di "sinistra", ovvero della prima tabella inserita nel predicato di join.

Per le righe nelle quali la condizione di join è soddisfatta tutto funziona come in un INNER JOIN.

Le righe della tabella di sinistra per le quali non è soddisfatto il predicato di join vengono comunque incluse nel set dei risultati, inserendo NULL in corrispondenza dei campi della tabella di destra.

```
SELECT * FROM Dipendenti D LEFT JOIN Sedi S ON D.Sede = S.CodSede
```

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.CodSede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	S01	Roma
2	Bianchi	Lucia	Infermiere	1500	NULL	NULL	NULL
3	Verdi	Carlo	Amministrativo	1500	S01	S01	Roma

DML – Right Join

Il Right Join è duale al Left Join. Prevede che il set dei risultati contenga sempre e comunque tutte le righe della tabella di "destra", ovvero della seconda tabella inserita nel predicato di join.

Per le righe nelle quali la condizione di join è soddisfatta tutto funziona come in un INNER JOIN.

Le righe della tabella di destra, per le quali non è soddisfatto il predicato di join, vengono comunque incluse nel set dei risultati, inserendo NULL in corrispondenza dei campi della tabella di sinistra.

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.CodSede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	S01	Roma
3	Verdi	Carlo	Amministrativo	1500	S01	S01	Roma
NULL	NULL	NULL	NULL	NULL	NULL	S02	Milano

DML – Full outer Join

Il Full outer Join combina le caratteristiche del right e del left join. Prevede che il set dei risultati contenga sempre e comunque tutte le righe della tabella di "destra" e tutte le righe della tabella di "sinistra", combinando quelle che soddisfano la condizione di join e inserendo NULL nei campi per i quali non c'è corrispondenza.

```
SELECT * FROM Dipendenti D FULL OUTER JOIN Sedi S ON D.Sede =
S.CodSede
```

Dipendenti.CodDip	Dipendenti.Cognome	Dipendenti.Nome	Dipendenti.Ruolo	Dipendenti.Stipendio	Dipendenti.Sede	Sedi.CodSede	Sedi.Città
1	Rossi	Mario	Medico	5000	S01	S01	Roma
2	Bianchi	Lucia	Infermiere	1500	NULL	NULL	NULL
3	Verdi	Carlo	Amministrativo	1500	S01	S01	Roma
NULL	NULL	NULL	NULL	NULL	NULL	S02	Milano

DML – Predicato di Join

Il predicato di Join può contenere espressioni multiple, in AND tra di loro, per specificare correlazioni multi-campo.

Supponiamo che la normalizzazione avesse spostato le medaglie in una tabella dedicata

`Medals(IdAthlete, IdGame, IdEvent, Medal)`

```
SELECT * FROM Partecipations P

INNER JOIN Medals M ON M.IdAthlete = P.IdAthlete AND M.IdGame =
P.IdGame AND M.IdEvent = P.IdEvent
```

DML – Join Multipli

E' possibile concatenare più istruzioni di Join, anche di tipo diverso, per recuperare dati da un numero arbitrario di tabelle. E' sufficiente specificare più operatori di Join:

```
SELECT * FROM Tabella1 T1
INNER JOIN Tabella2 T2 ON T1.Id = T2.Id
INNER JOIN Tabella3 T3 ON T3.Id = T2.Id
LEFT JOIN Tabella4 T4 ON T4.Id = T1.Id
...
```

DML – Join in query di inserimento

Il join può essere utilizzato anche in query di inserimento, cancellazione e modifica.

L'inserimento, la cancellazione e la modifica avvengono comunque su una tabella alla volta.
Il Join è utilizzato per determinare i dati da inserire, da cancellare o da modificare.

Nelle query di INSERT il join può essere utilizzato nella query che recupera i dati da inserire.

```
INSERT INTO Tabella3
SELECT * FROM Tabella1 T1
INNER JOIN Tabella2 T2 ON T1.Id = T2.Id
WHERE condizione
```

DML – Join in query di cancellazione

Nelle query di cancellazione il join è utilizzato per determinare le righe da cancellare. La cancellazione avviene però sempre e solamente su una tabella alla volta.

```
DELETE T1
FROM Tabella1 T1
INNER JOIN Tabella2 T2 ON T1.Id = T2.Id
WHERE condizione
```

DML – Join in query di update

Il join nelle istruzioni di update è spesso utilizzato per aggiornare i campi di una tabella con i valori presenti in un'altra tabella.

```
UPDATE Tabella1 T1
INNER JOIN Tabella2 T2 ON T1.Id = T2.Id
SET T1.CampoX = T2.CampoY
WHERE condizione
```

Esercizio: Join

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Popolare le tabelle che sono ancora vuote (quelle con chiave esterna)
2. Creare una query di selezione che ricostruisca la tabella AthletesFull da cui siamo partiti (tranne campo Id).

Ricostruire alcune query delle sezioni precedenti

1. Qual è l'età media dei medagliati?
2. Recuperare i 10 atleti che hanno vinto più medaglie d'oro, calcolandone il numero ed ordinando in modo decrescente
3. Stilare la classifica degli sport che hanno assegnato più medaglie

Esercizio: Join

Inoltre:

1. Recuperare tutti i nomi degli atleti che hanno partecipato all'edizione del 2012
2. Per ogni edizione calcolare il numero totale di medaglie assegnate
3. Quale evento ha assegnato più medaglie nella storia dei giochi?

Esercizio: Join - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Popolare le tabelle che sono ancora vuote (quelle con chiave esterna)

```
INSERT INTO Partecipations(Id, IdAthlete, Age, NOC, IdGame, IdEvent)

SELECT DISTINCT A.Id, A.IdAthlete, A.Age, A.NOC, G.Id AS IdGame, E.Id As IdEvent

FROM AthletesFull A INNER JOIN Games G On A.Games = G.Games AND A.Year=G.Year AND A.Season =
G.Season

INNER JOIN Events E On A.Sport = E.Sport AND A.Event = E.Event
```

Esercizio: Join - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

2. Creare una query di selezione che ricostruisca la tabella AthletesFull da cui siamo partiti (tranne campo Id).

```
SELECT A.IdAthlete, A.Name, A.Sex, P.Age, A.Height, A.Weight, P.Noc, G.Games, G.Year, G.Season,
G.City, E.Sport, E.Event, P.Medal

FROM Partecipations P

INNER JOIN Athletes A ON P.IdAthlete = A.IdAthlete

INNER JOIN Events E ON P.IdEvent = E.Id

INNER JOIN Games G ON P.IdGame = G.Id
```

Esercizio: Join - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

Ricostruire alcune query delle sezioni precedenti:

1. Qual è l'età media dei medagliati?

```
SELECT AVG(Age)
FROM Partecipations P
WHERE Medal IS NOT NULL
```

Esercizio: Join - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

Ricostruire alcune query delle sezioni precedenti:

2. Recuperare i 10 atleti che hanno vinto più medaglie d'oro, calcolandone il numero ed ordinando in modo decrescente

```
SELECT TOP(10) A.IdAthlete, A.Name, COUNT(Medal) As Medals
FROM Athletes A
INNER JOIN Partecipations M ON A.IdAthlete = P.IdAthlete
WHERE P.Medal = 'Gold'
GROUP BY A.IdAthlete, A.Name
ORDER BY COUNT(Medal) DESC
```

Esercizio: Join - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

Ricostruire alcune query delle sezioni precedenti:

3. Stilare la classifica degli sport che hanno assegnato più medaglie

```
SELECT E.Sport, COUNT(Medal) As Medals
FROM Events E
INNER JOIN Partecipations P ON E.Id = P.IdEvent
GROUP BY E.Sport
ORDER BY COUNT(Medal) DESC
```

Esercizio: Join - Soluzioni

Inoltre:

1. Recuperare tutti i nomi degli atleti che hanno partecipato all'edizione del 2012

```
SELECT DISTINCT A.IdAthlete, A.Name

FROM Athletes A

INNER JOIN Partecipations P ON A.IdAthlete = P.IdAthlete

INNER JOIN Games G ON P.IdGame = G.Id

WHERE G.Year = 2012

ORDER BY A.Name
```

Esercizio: Join - Soluzioni

Inoltre:

2. Per ogni edizione calcolare il numero totale di medaglie assegnate

```
SELECT G.Games, COUNT(P.Medal) AS Medals

FROM Games G

INNER JOIN Partecipations P ON G.Id = P.IdGame

GROUP BY G.Games

ORDER BY G.Games
```

Esercizio: Join - Soluzioni

Inoltre:

3. Quale evento ha assegnato più medaglie nella storia dei giochi?

```
SELECT TOP(1) E.Event, COUNT(P.Medal) AS Medals
FROM Events E
INNER JOIN Partecipations P ON E.Id = P.IdEvent
GROUP BY E.Event
ORDER BY COUNT(P.Medal) DESC
```

DML – Subquery

In SQL è possibile esprimere le condizioni della clausola di WHERE facendo uso del risultato di altre query. Tali query sono dette subquery, query innestate o query annidate.

Per esempio

```
SELECT * FROM Dipendenti
WHERE Sede IN (SELECT CodSede from Sedi)
```

Le query vengono eseguite partendo da quelle più interne (più innestate) per poi passare, via via, a quelle più esterne.

Una volta terminata l'esecuzione di una sottoquery, questa viene sostituita dal suo risultato e l'elaborazione procede.

DML – Subquery

```
SELECT * FROM Dipendenti
WHERE Sede IN (SELECT CodSede from Sedi)
```

La query riportata nell'esempio è equivalente alla query:

```
SELECT * FROM Dipendenti
WHERE Sede IN ('S01','S02')
```

Con l'unica differenza che la prima versione, quella con la subquery, funziona anche se viene inserita una nuova riga nella tabella Sedi.

DML – Subquery

Nella maggior parte dei casi le sottoquery devono restituire una sola colonna e un numero di righe che dipende dall'operatore utilizzato nella clausola WHERE. Per esempio:

- Operatore IN → La sottoquery deve restituire una colonna e un numero qualsiasi di righe
- Operatori matematici → La sottoquery deve restituire una colonna e una riga (valore scalare)

In caso contrario la query non verrà eseguita e genererà un errore. **Le query che seguono, per esempio, sono errate:**

```
SELECT * FROM Dipendenti

WHERE Sede IN (SELECT * from Sedi)
```

```
SELECT * FROM Dipendenti

WHERE Stipendio = (SELECT Stipendio from Dipendenti ORDER BY 1 )
```

DML – Subquery - EXISTS

Utilizzando le subquery possiamo sfruttare anche l'operatore EXISTS che permette di verificare se il risultato di una subquery restituisce almeno una riga.

```
SELECT * FROM Athletes
WHERE EXISTS (SELECT * FROM Partecipations WHERE Medal IS NOT NULL)
```

Se la sottoquery restituisce una o più righe allora la query esterna avrà luogo. E' anche possibile utilizzare NOT EXISTS per verificare se la query iterna non restituisce righe.

Questo utilizzo di EXISTS non è particolarmente interessante in quanto la query esterna restituisce sempre lo stesso risultato, indipendentemente dalla query esterna.

DML – Subquery - EXISTS

L'utilizzo di EXISTS diventa più interessante nel momento in cui c'è un collegamento tra campi della query esterna e campi della query interna. In tale caso la subquery è detta correlata:

Per esempio:

```
SELECT * FROM Athletes A
WHERE EXISTS (SELECT * FROM Partecipations WHERE Medal IS NOT NULL
AND IdAthlete = A.IdAthlete)
```

In caso di query correlate la sottoquery viene valutata per ogni riga della query esterna, determinando quindi se tale riga deve essere restituita o meno.

DML – Subquery

In molte situazioni la stessa interrogazione può essere realizzata sia con subquery che con join.

Per esempio la query precedente può essere ottenuta anche con un join

```
SELECT DISTINCT A.* FROM
Athletes A INNER JOIN Partecipations P ON A.IdAthlete = P.IdAthlete
WHERE P.Medal IS NOT NULL
```

La forma con query innestata "potrebbe" essere più lenta in fase di esecuzione, in quanto si obbliga il DBMS a seguire un ordine di esecuzione definito da noi (prima le query interne e poi quelle più esterne)

DML – Subquery

Passare da subquery a join e viceversa potrebbe non essere semplice. Per esempio:

```
SELECT * FROM Athletes A

WHERE NOT EXISTS (SELECT * FROM Partecipations WHERE Medal IS NOT
NULL

AND IdAthlete = A.IdAthlete)
```

Si trasforma in

```
SELECT DISTINCT A.* FROM

Athletes A LEFT JOIN Partecipations P ON A.IdAthlete = P.IdAthlete

WHERE P.Medal IS NULL
```

DML – Subquery e aggiornamento dati

Le subquery sono parte integrante delle clausole WHERE e pertanto utilizzabili anche nei comandi di DELETE e UPDATE. Per esempio:

```
DELETE FROM Athletes A
WHERE NOT EXISTS (SELECT * FROM Partecipations WHERE Medal IS NOT NULL
AND IdAthlete = A.IdAthlete)
```

DML – Subquery in FROM

Le subquery possono essere inserite anche nella clausola FROM. Il loro effetto è quello di poter effettuare una query sul risultato di una seconda query (innestata).

```
SELECT COUNT(S.NOCs) FROM (
	SELECT IdAthlete, COUNT(distinct NOC) AS NOCs
	FROM AthletesFull
	GROUP BY IdAthlete
	HAVING COUNT(distinct NOC) > 1
) S
```

Alla subquery deve sempre essere assegnato un nome.

I campi della subquery possono essere utilizzati nelle clausole della query esterna.

DML – Campi calcolati con subquery

Le subquery possono essere utilizzate anche per calcolare singoli campi.

```
select distinct IdAthlete, Name,Sex,
( SELECT COUNT(*) FROM AthletesFull Where IdAthlete = A.IdAthlete AND Medal = 'Gold' ) As Gold,
( SELECT COUNT(*) FROM AthletesFull Where IdAthlete = A.IdAthlete AND Medal = 'Silver' ) As
Silver,
( SELECT COUNT(*) FROM AthletesFull Where IdAthlete = A.IdAthlete AND Medal = 'Bronze' ) As
Bronze
from AthletesFull A
WHERE IdAthlete = 94406
```

Le sottoquery vengono calcolate per ogni riga compatibile con la clausola WHERE (in questo caso prima dell'applicazione di distinct). Nelle sottoquery possono essere utilizzati campi della query esterna.

ATTENZIONE ALLE PRESTAZIONI!

Esercizio: Subquery

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Recuperare tutti i nomi degli atleti che hanno partecipato all'edizione del 2012 (con subquery)
2. Quanti atleti hanno vinto più di 10 medaglie?

Esercizio: Subquery - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Recuperare tutti i nomi degli atleti che hanno partecipato all'edizione del 2012 (con subquery)

```
SELECT A.IdAthlete, A.Name
FROM Athletes A
WHERE A.IdAthlete IN (
        SELECT P.IdAthlete FROM Partecipations P
        WHERE P.IdGame = (
                SELECT TOP(1) Id FROM Games WHERE Year = 2012
        )
)
```

Esercizio: Subquery - Soluzioni

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

2. Quanti atleti hanno vinto più di 10 medaglie?

```
SELECT COUNT(*) FROM
(
        SELECT IdAthlete, COUNT(Medal) As Medals
        FROM Partecipations GROUP BY IdAthlete
        HAVING COUNT(Medal)>10
) Sub
```

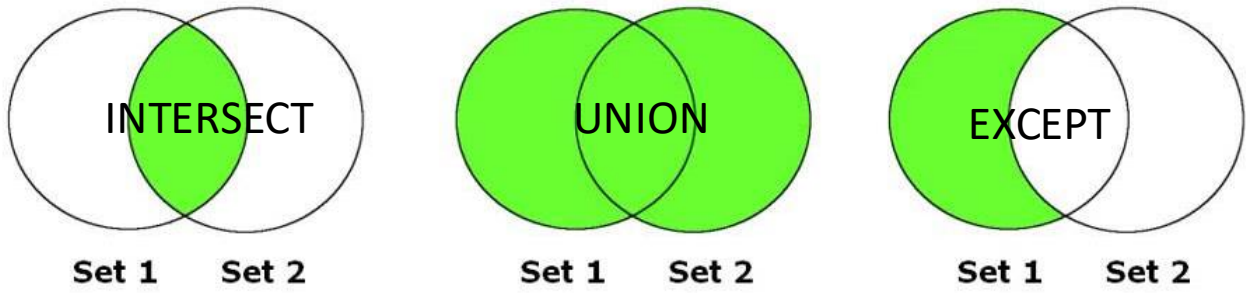
DML – Operatori insiemistici

E' possibile combinare insieme il risultato di due query, ottenendo quindi un unico output.

La combinazione dei risultati avviene tramite tre operatori:

- UNION
- INTERSECT
- EXCEPT

Vengono chiamati operatori insiemistici perché il loro funzionamento deriva esattamente dalla teoria degli insi



DML – Operatori insiemistici - UNION

L'operatore UNION viene inserito "come separatore" tra le due query da unire. Per esempio:

```
SELECT DISTINCT Name,Year
FROM AthletesFull A
WHERE NOC='ITA'
UNION
SELECT DISTINCT Nome,2024 AS Year
FROM Athletes2024 A
WHERE Nation = 'Italy'
```

DML – Operatori insiemistici - UNION

Affinché la UNION abbia successo è necessario che entrambe le query restituiscano lo stesso numero di campi e che questi siano di tipi compatibili.

Il primo campo della prima query deve essere compatibile con il primo campo della seconda query, il secondo campo della prima query deve essere compatibile con il secondo campo della seconda query…

Il comportamento di default di UNION prevede l'eliminazione dei duplicati. Per mantenerli si deve aggiungere l'opzione ALL, utilizzando pertanto il comando UNION ALL

Esercizio: Union

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Recuperare le medaglie vinte da Micheal Phelps (IdAthlete = 93860), divise per oro, argento e bronzo e ordinate per importanza decrescente del metallo (usare union)

IdAthlete	Name	Medal	Number
93860	Michael Phelps	Gold	23
93860	Michael Phelps	Silver	3
93860	Michael Phelps	Bronze	2

Esercizio: Union - Soluzione

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

1. Recuperare le medaglie vinte da Micheal Phelps (IdAthlete = 94406), divise per oro, argento e bronzo e ordinate per importanza decrescente del metallo (usare union e non group by)

```
SELECT A.IdAthlete,A.Name,P.Medal, COUNT(*) As Number
FROM Athletes A
INNER JOIN Partecipations P ON A.IdAthlete = P.IdAthlete
WHERE P.Medal = 'Gold' AND A.IdAthlete = 94406
GROUP BY A.IdAthlete,A.Name,P.Medal
UNION
SELECT A.IdAthlete,A.Name,P.Medal, COUNT(*) As Number
FROM Athletes A
INNER JOIN Partecipations P ON A.IdAthlete = P.IdAthlete
WHERE P.Medal = 'Silver' AND A.IdAthlete = 94406
GROUP BY A.IdAthlete,A.Name,P.Medal … + altro UNION per Bronze
```

Esercizio: Challenge finale

Con riferimento alla tabelle normalizzate dei dati relativi alle olimpiadi:

Recuperare il medagliere per nazioni dell'edizione 2016, ordinato per numero decrescente di ori, di argenti e di bronzi.

ATTENZIONE a conteggiare correttamente gli sport di squadra. Assegnano una medaglia ad ogni atleta della squadra ma nel medagliere per nazioni devono essere conteggiate una sola volta.

NOC	Golds	Silvers	Bronzes	Total
USA	46	37	38	121
GBR	27	23	17	67
…	…	…	…	…

Potete verificare il medagliere completo su Wikipedia