SQL Data Manipolation Language

Istruzioni di base

DML

La manipolazione dei dati in SQL avviene principalmente tramite 4 parole chiave:

- **SELECT**: Permette di recuperare i dati dal db eseguendo interrogazioni (query).Ha numerosi parametri che permettono di eseguire query anche molto complesse. Per questo motivo si parla anche di DQL (Data Query Language).
- **INSERT**: Permette di inserire nuovi dati nel database. Può anche utilizzare il risultato di una query di SELECT.
- **DELETE**: Permette di cancellare dati esistenti.
- **UPDATE**: Permette di modificare uno o più campi dei dati esistenti

DELETE e UPDATE fanno uso di condizioni avanzate per determinare quali dati cancellare/modificare. Sono le stesse condizioni utilizzate da SELECT

DML – Selezione dei record

La selezione di record da una o più tabelle (interrogazione / query) avviene tramite il costrutto SELECT.

SELECT ha numerosi parametri che permettono la realizzazione di interrogazioni anche molto complesse. Nella sua forma più semplice è:

```
SELECT <elenco di campi>
FROM <elenco di tabelle>
WHERE <condizioni>
```

L'istruzione lavora sulle tabelle specificate nella clausola FROM e sulle sole righe che soddisfano la condizione impostata nella clausola WHERE. Vengono restituiti solamente i campi specificati nella clausola SELECT.

La clausola WHERE non è obbligatoria. In tal caso vengono restituite tutte le righe.

DML – Selezione dei record

Esempio:

```
SELECT Name, Sex
FROM AthletesFull
WHERE NOC = 'ITA'
```

Se vogliamo tutti i campi possiamo evitare di scriverli tutti, utilizzando l'operatore *

```
SELECT *
FROM AthletesFull
WHERE NOC = 'ITA'
```

L'operatore * restituisce i campi nello stesso ordine con cui sono stati definiti nella tabella.

DML – Selezione dei record senza duplicati

Come detto in precedenza, le tabelle e le query SQL permettono righe duplicate. Cosa non possibile con le relazioni.

Per fare in modo che il risultato di una query non comprenda i valori duplicati utilizziamo la parola chiave DISTINCT:

```
SELECT DISTINCT Name, Sex
FROM AthletesFull
WHERE NOC = 'ITA'
```

Porre sempre ATTENZIONE alla correttezza dei dati restituiti.
In questo caso: a fronte di atleti differenti ma con stesso nome voglio una riga sola o più righe?

DML – Espressioni in SELECT

L'elenco dei campi inserito nella clausola SELECT può anche contenere espressioni, utili per restituire valori differenti rispetto a quanto memorizzato nel database.

```
SELECT DISTINCT Name, Sex, Height*0.0328084, Weight*2.20462
FROM AthletesFull
WHERE NOC = 'ITA'
```

I campi calcolati tramite espressioni non hanno un nome significativo. Possiamo assegnarlo manualmente.

DML – Rinominare le colonne

E’ possibile associare un nome arbitrario alle colonne inserite nella clausola SELECT. Utile per dare nomi più significativi e/o per dare un nome alle colonne calcolate.

Ciò è possibile tramite l’operatore AS posto dopo la colonna a cui si riferisce.

```
SELECT DISTINCT Name, Sex, Height*0.0328084 AS HeightFt,
Weight*2.20462 AS WeightLb

FROM AthletesFull

WHERE NOC = 'ITA'
```

DML – Clausola WHERE

Nella clausola WHERE possiamo costruire espressioni complesse per determinare quali righe della tabella devono essere comprese nel risultato della query.

Gli operatori di base sono quelli propri dell'algebra booleana: =, <>, <, <=, >, >=, AND, OR, NOT

E' possibile anche l'utilizzo delle parentesi tonde e degli operatori algebrici + - * /

```
SELECT *
FROM AthletesFull
WHERE NOC = 'ITA' AND Year = 2016
```

```
SELECT *
FROM AthletesFull
WHERE NOC = 'ITA' AND Year >= 1980 AND Year < 2000
```

DML – Clausola WHERE

Esistono anche altri operatori per migliorare l'espressività delle query. Vedremo:

- BETWEEN
- IN
- LIKE
- IS / IS NOT NULL

DML – Clausola WHERE

BETWEEN permette di specificare un range di valori per restituire tutti le righe che ne fanno parte (o che non ne fanno parte).

BETWEEN ha lo stesso effetto di >= … AND <= …

```
SELECT *
FROM AthletesFull
WHERE NOC = 'ITA' AND Year BETWEEN 1980 AND 1999
```

Per ottenere le righe non comprese nell'intervallo si utilizza NOT BETWEEN

L'operatore funziona anche con stringhe. In tal caso l'ordinamento è quello alfabetico.

DML – Clausola WHERE

IN permette di specificare un elenco di valori per restituire tutti le righe che ne fanno parte (o che non ne fanno parte).

IN ha lo stesso effetto di = … OR = … OR = …

L'elenco di valori va espresso tra parentesi tonde, separati da virgola.

```
SELECT *

FROM AthletesFull

WHERE NOC IN ('ITA','USA')
```

Per ottenere le righe non comprese nell'elenco si utilizza NOT IN

DML – Clausola WHERE

LIKE permette di effettuare ricerche su parti di un campo testuale. Tramite caratteri speciali è possibile determinare dei "pattern" di ricerca sui campi:

- ? Un singolo carattere
- % Una stringa di 0 o più caratteri

```
SELECT *
FROM AthletesFull
WHERE Name LIKE '%Phelps%'
```

Per ottenere le righe non soddisfano il pattern di ricerca si utilizza NOT LIKE

DML – Clausola WHERE

Tutti gli operatori visti in precedenza IGNORANO i campi con valore NULL.

Per considerarli, o per escluderli, utilizziamo rispettivamente gli operatori IS NULL e IS NOT NULL

```
SELECT *
FROM AthletesFull
WHERE Medal IS NOT NULL
```

Esercizio: Query di Selezione / 1

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

1. Selezione di tutti gli atleti italiani che hanno gareggiato nel nuoto (Swimming) nelle olimpiadi del 1952
2. Dettagliare meglio la query precedente per restituire solo gli atleti di sesso maschile che hanno gareggiato nel 100m style libero (usare like)
3. Scoprire se i giochi del 1998 erano estivi o invernali. La query deve restituire 1 riga e 1 colonna.
4. Selezionare tutti gli atleti che hanno vinto una medaglia ad una età compresa tra 13 e 15 anni
5. Recuperare tutte le edizioni dei giochi svolte a Los Angeles o a London

Esercizio: Query di Selezione / 1 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

1. Selezione di tutti gli atleti italiani che hanno gareggiato nel nuoto (Swimming) nelle olimpiadi del 1952

```
USE Olympics;

SELECT *

FROM AthletesFull

WHERE NOC = 'ITA' AND Sport = 'Swimming' AND Year = 1952
```

Esercizio: Query di Selezione / 1 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

2. Dettagliare meglio la query precedente per restituire solo gli atleti di sesso maschile che hanno gareggiato nel 100m style libero (usare like)

```
USE Olympics;

SELECT *

FROM AthletesFull

WHERE NOC = 'ITA' AND Sport = 'Swimming' AND Year = 1952 AND Sex='M' AND Event LIKE
'%100%metres%free%'
```

Esercizio: Query di Selezione / 1 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

3. Scoprire se i giochi del 1998 erano estivi o invernali. La query deve restituire 1 riga e 1 colonna.

```
USE Olympics;

SELECT DISTINCT Season

FROM AthletesFull

WHERE Year = 1998
```

Esercizio: Query di Selezione / 1 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

4. Selezionare tutti gli atleti che hanno vinto una medaglia ad una età compresa tra 13 e 15 anni

```
USE Olympics;

SELECT DISTINCT IdAthlete, Name, Age

FROM AthletesFull

WHERE Age BETWEEN 13 AND 15 AND Medal IS NOT NULL
```

Esercizio: Query di Selezione / 1 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

5. Recuperare tutte le edizioni dei giochi svolte a Los Angeles o a London

```
USE Olympics;

SELECT DISTINCT Games,City

FROM AthletesFull

WHERE City IN ('London','Los Angeles')
```

DML – Ordinare i risultati

L'ordine con cui i risultati vengono restituiti non è fisso. Dipende da come sono organizzati i dati fisici sul database.

Per imporre un determinato ordinamento si utilizza la clausola ORDER BY seguita da un elenco di campi su sui ordinare.

L'ordinamento avverrà sul primo campo inserito e, a parità di questo sul secondo, e così via.

L'ordinamento di default e crescente (ASC). Per imporre l'ordinamento decrescente si utilizza la parola chiave DESC

```
SELECT *
FROM AthletesFull
WHERE Medal IS NOT NULL
ORDER BY Year, Age DESC
```

DML – Ordinare i risultati

I parametri inseriti in ORDER BY possono essere elencati per nome (come già visto) o per "posizione" nell'elenco di selezione. Questa modalità è possibile solamente se i parametri sono specificati in maniera esplicita e non si utilizza SELECT *

```
SELECT IdAthlete,Name,Year,Age
FROM AthletesFull
WHERE Medal IS NOT NULL
ORDER BY 3, Age DESC
```

La clausola ORDER BY così scritta è equivalente a:

```
ORDER BY Year, Age DESC
```

DML – Limitare il numero dei risultati

Talvolta è comodo limitare il numero di risultati restituiti rispetto a quando previsto dalla clausola WHERE. Può essere utilizzato, per esempio, a fini di debug, per verificare che la query che stiamo scrivendo sia corretta.

Tramite la clausola TOP(n), inserita nella clausola di SELECT, è possibile specificare il numero massimo di righe da restituire.

```
SELECT TOP(10) *
FROM AthletesFull
WHERE Medal IS NOT NULL
```

DML – Limitare il numero dei risultati

E' possibile limitare il numero di risultati dopo averne "scartati alcuni". E' utile per realizzare versioni paginate (soprattutto quando le query sono eseguite da codice).

```
SELECT *

FROM AthletesFull

WHERE Medal IS NOT NULL

ORDER BY Id

OFFSET 10 ROWS

FETCH NEXT 10 ROWS ONLY
```

La query restituisce 10 righe a partire dalla 11°. NB: Order by è obbligatorio e non può essere combinata con TOP

Esercizio: Query di Selezione / 2

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

1. Selezione i 10 atleti più giovani ad aver vinto una medaglia d'oro
2. Selezione i 10 atleti più anziani ad aver vinto una medaglia d'oro
3. Selezionare lo sport e l'evento a cui ha partecipato l'atleta più anziano, nei giochi dal 2000 in poi
4. Recuperare l'elenco delle città che hanno ospitato i giochi, in ordine alfabetico
5. Ottenere l'elenco dei partecipanti (solo il nome), in ordine alfabetico e suddivisi per anno e nazione

Esercizio: Query di Selezione / 2 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

1. Selezione i 10 atleti più giovani ad aver vinto una medaglia d'oro

```
USE Olympics;

SELECT DISTINCT TOP(10) IdAthlete, Name, Age, NOC, Sport

FROM AthletesFull

WHERE Medal = 'Gold' AND Age IS NOT NULL

ORDER BY Age
```

Esercizio: Query di Selezione / 2 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

2. Selezione i 10 atleti più anziani ad aver vinto una medaglia d'oro

```
USE Olympics;

SELECT TOP(10) DISTINCT IdAthlete, Name, Age, NOC, Sport

FROM AthletesFull

WHERE Medal = 'Gold' AND Age IS NOT NULL

ORDER BY Age DESC
```

Esercizio: Query di Selezione / 2 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

3. Selezionare lo sport e l'evento a cui ha partecipato l'atleta più anziano, nei giochi dal 2000 in poi

```
USE Olympics;

SELECT  TOP(1) Sport, Event

FROM AthletesFull

WHERE Year>=2000 AND Age IS NOT NULL

ORDER BY Age DESC, Name
```

Esercizio: Query di Selezione / 2 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

4. Recuperare l'elenco delle città che hanno ospitato i giochi, in ordine alfabetico

```
USE Olympics;

SELECT DISTINCT City

FROM AthletesFull

ORDER BY City
```

Esercizio: Query di Selezione / 2 - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere le seguenti query di selezione:

5. Ottenere l'elenco dei partecipanti (solo il nome), in ordine alfabetico e suddivisi per anno e nazione

```
USE Olympics;

SELECT Name

FROM AthletesFull

ORDER BY Year, NOC, Name
```

DML – Inserimento di righe

L'inserimento di righe all'interno di una tabella esistente avviene tramite il comando INSERT INTO, specificando l'elenco dei valori da inserire in ogni campo della nuova riga.

```
INSERT INTO Nations(Id, Name, NOC) VALUES(1,'Italy','ITA')
```

La lista degli attributi può essere omessa. In tal caso viene considerato l'ordine di creazione.

```
INSERT INTO Nations VALUES(1,'Italy','ITA')
```

Gli attributi non compresi nella lista assumono il valore di default (se previsto), oppure il valore NULL (se ammesso).

```
INSERT INTO Nations(Name, NOC) VALUES('Italy','ITA')
```

DML – Inserimento di righe

L'inserimento contemporaneo di più righe avviene combinando INSERT INTO con SELECT, ovvero inserendo nella tabella il risultato di un'altra query.

```
INSERT INTO Nations(NOC)

SELECT DISTINCT NOC

FROM AthletesFull
```

I nomi dei campi della query di SELECT possono anche essere differenti rispetto ai nomi dei campi della tabella di destinazione. L'importante è che abbiano un tipo di dato compatibile: il primo campo della query di select deve essere compatibile con il primo campo della query di insert…

DML – Inserimento di righe

INSERT INTO pretende che la tabella in cui inserire le nuove righe sia esistente. Se non esiste va precedentemente creata con CREATE TABLE.

Per effettuare le due operazioni contemporaneamente si utilizza SELECT INTO

```
SELECT DISTINCT NOC
INTO Nations
FROM AthletesFull
```

La nuova tabella viene creata deducendo nomi e tipi dalla query di select. Non vengono creati vincoli di nessun tipo, che pertanto vanno integrati manualmente.

SELECT INTO genera errore se la tabella esiste.

DML – Cancellazione di righe

La cancellazione avviene tramite l'istruzione DELETE nella quale si specifica la tabella su cui operare e una clausola WHERE tramite la quale selezionare le righe da cancellare.

```
DELETE FROM Nations

WHERE NOC IS NULL
```

La clausola WHERE è identica a quanto giù visto per SELECT.

La cancellazione di una riga potrebbe portare alla violazione di qualche vincolo di integrità referenziale. Vedremo più avanti come gestire questi casi.

DML – Aggiornamento di righe

L'aggiornamento avviene tramite l'istruzione UPDATE nella quale si specifica la tabella su cui operare, una clausola WHERE tramite la quale selezionare le righe da aggiornare, e i valori aggiornati da assegnare a uno o più campi.

```
UPDATE AthletesFull

SET Medal = NULL

WHERE Medal NOT IN ('Gold','Silver','Bronze')
```

La clausola WHERE è identica a quanto già visto per SELECT.

Anche l'aggiornamento di una riga potrebbe portare alla violazione di qualche vincolo di integrità referenziale.

Esercizio: Query di Inserimento

Con riferimento alla tabella AthletesFull del db Olympics, e alle tabelle normalizzate precedentemente create:

1. Scrivere le query di inserimento che, leggendo i dati dalla tabella AthletesFull, riempie le tabelle normalizzate.

NB: Per il momento lavorare solamente sulle tabelle normalizzate che non contengono chiavi esterne. Le altre le affronteremo più avanti.

Esercizio: Query di Inserimento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, e alle tabelle normalizzate precedentemente create:

1. Scrivere le query di inserimento che, leggendo i dati dalla tabella AthletesFull, riempie le tabelle normalizzate.

```
USE Olympics

DELETE FROM Events

INSERT INTO Events(Event,Sport)

SELECT DISTINCT Event,Sport FROM AthletesFull
```

Esercizio: Query di Inserimento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, e alle tabelle normalizzate precedentemente create:

1. Scrivere le query di inserimento che, leggendo i dati dalla tabella AthletesFull, riempie le tabelle normalizzate.

```
USE Olympics;

DELETE FROM Games;

INSERT INTO Games(Games, Year, Season, City)

SELECT DISTINCT Games, Year, Season, City FROM AthletesFull
```

Esercizio: Query di Inserimento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, e alle tabelle normalizzate precedentemente create:

1. Scrivere le query di inserimento che, leggendo i dati dalla tabella AthletesFull, riempie le tabelle normalizzate.

```
USE Olympics;

DELETE FROM Athletes;

INSERT INTO Athletes(IdAthlete, Name, Sex, Height, Weight)

SELECT DISTINCT IdAthlete, Name, Sex, Height, Weight FROM AthletesFull
```

SQL Data Manipolation Language

Raggruppamenti

DML – Raggruppamenti

Le istruzioni di SELECT viste in precedenza permettono di recuperare informazioni che si riferiscono a singole righe.

Ci sono molti casi nei quali è necessario recuperare informazioni riepilogative di gruppi di righe.

Per esempio:

- Il numero di atleti italiani che hanno partecipato alle olimpiadi di Londra del 2012.
- Il numero di olimpiadi disputate in ogni città

Per soddisfare queste necessità SQL mette a disposizione due strumenti:

- Funzioni di aggregazione
- Clausola GROUP BY

DML – Funzioni di Aggregazione

Le funzioni di aggregazione sono funzioni, principalmente matematiche, che lavorano su una intera colonna (o su un gruppo, lo vedremo tra poco) e restituiscono un unico valore anziché tutte le righe previste dalla clausola WHERE.

```
SELECT SUM(Stipendio) As StipendioMensileTotale
FROM Dipendenti
WHERE Ruolo = 'Medico'
```

La query di esempio genera una sola riga di risultato, con un solo campo, contenente la somma del campo Stipendio calcolata su tutte le righe che soddisfano la clausola WHERE.

DML – Funzioni di Aggregazione

Le funzioni di aggregazione possono anche lavorare su espressioni applicate a campi

```
SELECT SUM(Stipendio*12) As StipendioTotale
FROM Dipendenti
WHERE Ruolo = 'Medico'
```

DML – Funzioni di Aggregazione

Le principali funzioni di aggregazione sono:

- MIN → Minimo
- MAX → Massimo
- SUM → Somma
- AVG → Media aritmetica
- COUNT → Conteggio di righe

Solamente COUNT considera i valori NULL, tutte le altre funzioni li ignorano.

Min e Max funzionano anche su campi testuali e considerano l'ordinamento alfabetico.

Sum e Avg funzionano solo su campi numerici, mentre Count funziona su qualsiasi tipo di dato.

DML – Funzioni di Aggregazione

All'interno della stessa query posso inserire più funzioni di aggregazione. Verranno calcolate in modo indipendente l'una dall'altra

```
SELECT SUM(Stipendio),MIN(Stipendio),MAX(Stipendio),AVG(Stipendio),
COUNT(Stipendio)
FROM Dipendenti
WHERE Ruolo = 'Medico'
```

La query calcola la somma degli stipendi, lo stipendio minimo, massimo e medio dei dipendenti con ruolo Medico. Calcola inoltre il numero di righe su cui i risultati sono calcolati.

Non possono essere inseriti campi che non siano inseriti in funzioni di aggregazione

DML – Funzioni di Aggregazione - COUNT

La funzione COUNT è l'unica che considera i valori NULL. E' possibile comunque specificare se considerarli oppure no:

- COUNT(*) → I valori NULL vengono considerati
- COUNT(NomeCampo) → I valori NULL non vengono considerati

```
SELECT COUNT(*) FROM AthletesFull
```
→ Numero totale di record nella tabella

```
SELECT COUNT(Medal) FROM AthletesFull
```
→ Numero di record per i quali Medal non è null. E' pertanto equivalente a:

```
SELECT COUNT(*) FROM AthletesFull WHERE Medal IS NOT NULL
```

DML – Funzioni di Aggregazione - DISTINCT

E' possibile fare in modo che le funzioni di aggregazione lavorino su valori distinti. E' sufficiente inserire la parola chiave DISTINCT all'interno della funzione:

```
SELECT COUNT(DISTINCT Sport) FROM AthletesFull
```

La query restituisce il numero totale di sport mai apparsi alle olimpiadi

DML – CAST

Le funzioni di aggregazione determinano in automatico il tipo del risultato:

- SUM su colonna INT → Risultati di tipo INT
- SUM su colonna FLOAT → Risultato FLOAT
- AVG su colonna INT → Risultato INT

Per impostare in modo arbitrario il tipo di dato si utilizza la funzione CAST, specificando il tipo desiderato.

```
SELECT CAST(COUNT(DISTINCT Sport) AS FLOAT) FROM AthletesFull
```

La funzione CAST è utilizzabile anche nella clausola SELECT per modificare il tipo di dato delle colonne calcolate.

Esercizio: Funzioni di aggregazione

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

1. Quante città hanno ospitato giochi olimpici?
2. Quanti atleti hanno vinto la medaglia d'oro?
3. Qual è l'età media dei partecipanti?
4. Qual è l'età media dei medagliati?
5. Qual è l'età dell'atleta medagliato più giovane? (farla con aggregazione)
6. Quanti atleti italiani hanno Rossi nel cognome? (anche De Rossi, ma non Rossini ne Grossi)
7. Qual è l'altezza media degli atleti di basket maschile? E della ginnastica femminile?

Esercizio: Funzioni di aggregazione - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

1. Quante città hanno ospitato giochi olimpici?

```
SELECT COUNT(DISTINCT City) FROM AthletesFull
```

2. Quanti atleti hanno vinto la medaglia d'oro?

```
SELECT COUNT(DISTINCT IdAthlete) FROM AthletesFull WHERE Medal = 'Gold'
```

3. Qual è l'età media dei partecipanti?

```
SELECT AVG(CAST(Age AS FLOAT)) FROM AthletesFull
```

4. Qual è l'età media dei medagliati?

```
SELECT AVG(CAST(Age AS FLOAT)) FROM AthletesFull WHERE Medal IS NOT NULL
```

Esercizio: Funzioni di aggregazione

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

5. Qual è l'età dell'atleta medagliato più giovane? (farla con aggregazione)

```
SELECT MIN(Age) FROM AthletesFull WHERE Medal IS NOT NULL
```

6. Quanti atleti italiani hanno Rossi nel cognome? (anche De Rossi, ma non Rossini ne Grossi)

```
SELECT COUNT(DISTINCT IdAthlete) FROM AthletesFull

WHERE NOC = 'ITA' AND (Name LIKE '% Rossi' OR Name LIKE '% Rossi %' OR Name LIKE 'Rossi %' OR
Name = 'Rossi')
```

7. Qual è l'altezza media degli atleti di basket maschile? E della ginnastica femminile?

```
SELECT AVG(Height) FROM AthletesFull WHERE Sex='M' AND Sport = 'Basketball'

SELECT AVG(Height) FROM AthletesFull WHERE Sex='F' AND Sport = 'Gymnastics'
```

DML – Raggruppamento

Le funzioni di aggregazione viste fino ad ora non sono ancora sufficienti per coprire tutti i casi. Infatti:

- Calcolano il risultato su tutte le righe che soddisfano la clausola WHERE
- Non permettono di inserire campi non aggregati

Per superare questi limiti si utilizza la clausola GROUP BY, seguita da un elenco di campi su cui raggruppare. Per esempio:

```
SELECT NOC,COUNT(*) FROM AthletesFull GROUP BY NOC
```

In questa query viene restituito, per ogni nazione, il numero di atleti che hanno preso parte alle olimpiadi. Le righe vengono raggruppate a parità di campo NOC. Per ogni gruppo di righe viene calcolata la funzione di aggregazione COUNT.

I campi inseriti nella clausola GROUP BY possono anche essere inseriti in SELECT. Essendo identici all'interno del singolo gruppo non possono dare origine ad ambiguità.

DML – Raggruppamento

Una query con group by ritorna tante righe quanti sono i gruppi generati. E' possibile effettuare un ulteriore filtro per restituire solamente le righe che soddisfano determinate condizioni.

Per ottenere questo risultato si utilizza la clausola HAVING, nella quale si possono utilizzare le stesse modalità previste per WHERE potendo però inserire solamente campi aggregati e funzioni di aggregazione.

```
SELECT NOC,COUNT(Medal) FROM AthletesFull
WHERE Year = 2016
GROUP BY NOC
HAVING COUNT(Medal)>5
```

La query restituisce le nazioni che, nel 2016, hanno vinto più di 5 medaglie, riportandone anche il numero totale.

DML – Raggruppamento – Having e Where

Le clausole HAVING e WHERE hanno una funzionalità simile ma restano comunque differenti:

- WHERE → Filtra le righe prima del raggruppamento
- HAVING → Filtra le righe raggruppate dopo il raggruppamento

Il filtro per campo di raggruppamento può essere fatto sia su WHERE che su HAVING. Le due sintassi che seguono sono equivalenti:

```
SELECT NOC,COUNT(*)
FROM AthletesFull
GROUP BY NOC
HAVING NOC = 'ITA'
```

```
SELECT NOC,COUNT(*)
FROM AthletesFull
WHERE NOC = 'ITA'
GROUP BY NOC
```

DML – Raggruppamento e ordinamento

L'ordinamento avviene con la solita clausola ORDER BY ma avviene dopo il raggruppamento e pertanto la clausola va scritta dopo GROUP BY e HAVING.

Nella clausola ORDER BY è possibile però inserire solamente campi aggregati e funzioni di aggregazione.

```
SELECT NOC,COUNT(Medal) FROM AthletesFull
WHERE Year = 2016
GROUP BY NOC
HAVING COUNT(Medal)>5
ORDER BY NOC ASC, COUNT(*) DESC                    (ORDER BY 1 ASC, 2 DESC)
```

Esercizio: Funzioni di aggregazione con raggruppamento

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

1. Per ogni città calcolare quante edizioni ha ospitato. Ordinare in ordine decrescente.
2. Recuperare i 10 atleti che hanno vinto più medaglie d'oro, calcolandone il numero ed ordinando in modo decrescente.
3. Qual è l'età media dei medagliati per ogni edizione disputata. Ordinare per media crescente.
4. Ordinare le edizioni per numero totale, decrescente, di medaglie vinte dall'Italia.
5. Quale edizione ha visto la partecipazione di più nazioni?
6. Stilare la classifica degli sport che hanno assegnato più medaglie.
7. Stilare l'elenco degli atleti che hanno vinto più di 3 medaglie in almeno due edizioni
8. Stilare l'elenco degli atleti con il maggior tempo tra prima e ultima medaglia vinte.

Esercizio: Funzioni di aggregazione con raggruppamento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

1. Per ogni città calcolare quante edizioni ha ospitato. Ordinare in ordine decrescente.

```
SELECT City, COUNT(DISTINCT Games) AS Editions
FROM AthletesFull GROUP BY City
ORDER BY COUNT(DISTINCT Games) DESC, City
```

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

```
SELECT TOP(10) IdAthlete, Name, COUNT(Medal) AS Golds
FROM AthletesFull WHERE Medal = 'Gold'
GROUP BY IdAthlete, Name
ORDER BY COUNT(Medal) DESC, Name
```

Esercizio: Funzioni di aggregazione con raggruppamento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

3. Qual è l'età media dei medagliati per ogni edizione disputata. Ordinare per media crescente.

```
SELECT Games, AVG(Age) AS AvgAge
FROM AthletesFull WHERE Medal IS NOT NULL
GROUP BY Games
ORDER BY AVG(Age)
```

4. Ordinare le edizioni per numero totale, decrescente, di medaglie vinte dall'Italia.

```
SELECT Games, COUNT(Medal) AS Medals
FROM AthletesFull WHERE NOC = 'ITA'
GROUP BY Games
ORDER BY COUNT(Medal) DESC
```

Esercizio: Funzioni di aggregazione con raggruppamento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

5. Quale edizione ha visto la partecipazione di più nazioni?

```
SELECT TOP(1) Games, COUNT(DISTINCT NOC) AS Nations
FROM AthletesFull
GROUP BY Games
ORDER BY COUNT(DISTINCT NOC) DESC
```

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

```
SELECT Sport, COUNT(Medal) AS Medals
FROM AthletesFull
GROUP BY Sport
ORDER BY COUNT(Medal) DESC
```

Esercizio: Funzioni di aggregazione con raggruppamento - Soluzioni

Con riferimento alla tabella AthletesFull del db Olympics, scrivere query per dare risposta alle seguenti domande:

7. Stilare l'elenco degli atleti che hanno vinto più di 3 medaglie in almeno due edizioni

```
SELECT IdAthlete, Name, COUNT(Medal) AS Medals, COUNT(DISTINCT Games) AS Partecipations
FROM AthletesFull
WHERE Medal IS NOT NULL
GROUP BY IdAthlete, Name
HAVING COUNT(Medal)>3 AND COUNT(DISTINCT Games)>=2
ORDER BY COUNT(Medal) DESC
```

8. Stilare l'elenco degli atleti con il maggior tempo tra prima e ultima medaglia vinte.

```
SELECT IdAthlete, Name, Max(Year) - Min(Year) AS TimeDifference
FROM AthletesFull WHERE Medal IS NOT NULL
GROUP BY IdAthlete, Name
ORDER BY Max(Year) - Min(Year) DESC
```