1. LA CHIUSURA DEL LINGUAGGIO SQL E LE QUERY NIDIFICATE; JOIN E SELF-JOIN

QUERY NIDIFICATE

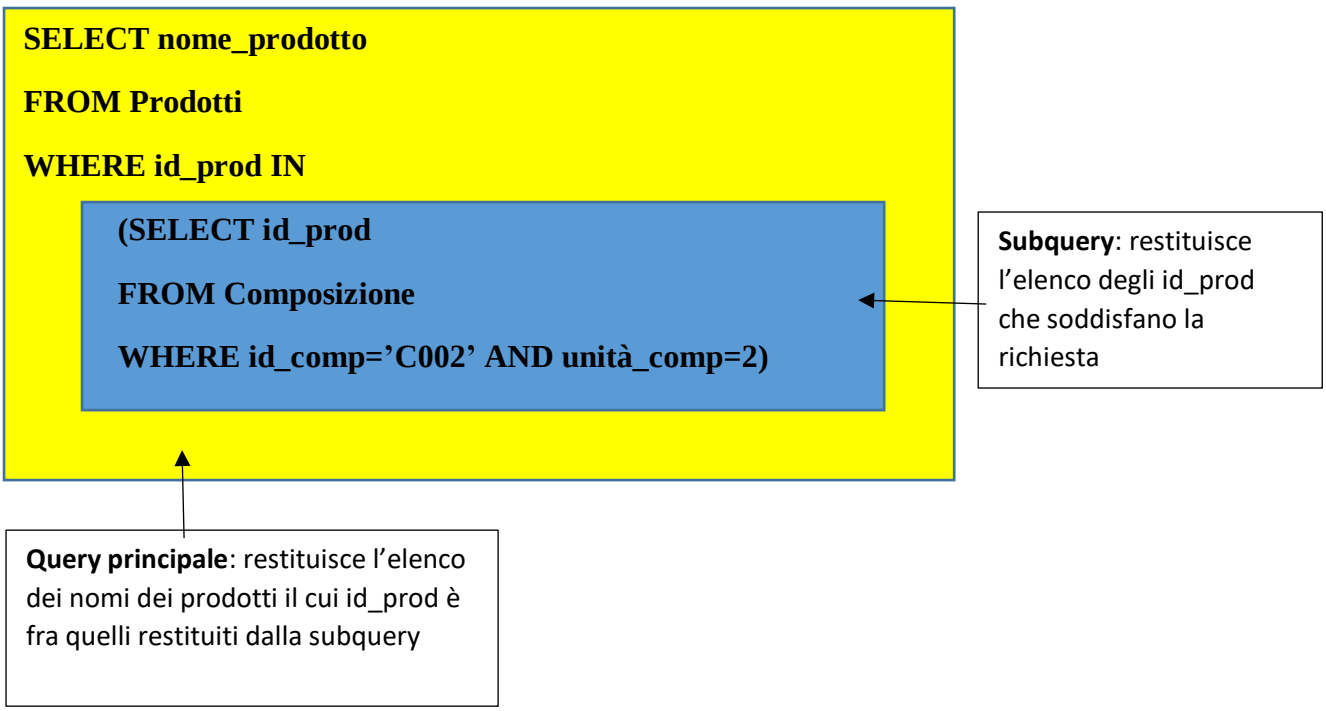
La proprietà di CHIUSURA del linguaggio SQL, ossia il fatto che le operazioni su tabelle danno come risultato altre tabelle, consente di **utilizzare il risultato di una query (chiamata subquery)** (è una tabella con dei valori) **come argomento della clausola WHERE di un'altra query** (con la parola chiave **IN**). In questo modo si possono quindi realizzare delle query nidificate, il cui risultato si **ottiene a partire dalla subquery più interna arrivando alla più esterna.**

Esempio1: Seleziona tutti i dipendenti (matricola e nominativo) che lavorano in provincia di Milano

- prima faccio la select per ottenere i codici dei dipartimenti con sede in provincia di Milano
- Poi inserisco la select nella query più esterna.

SELECT matricola, nominativo FROM personale WHERE id_dip IN (SELECT id_dip from dipartimenti WHERE provincia='MI');

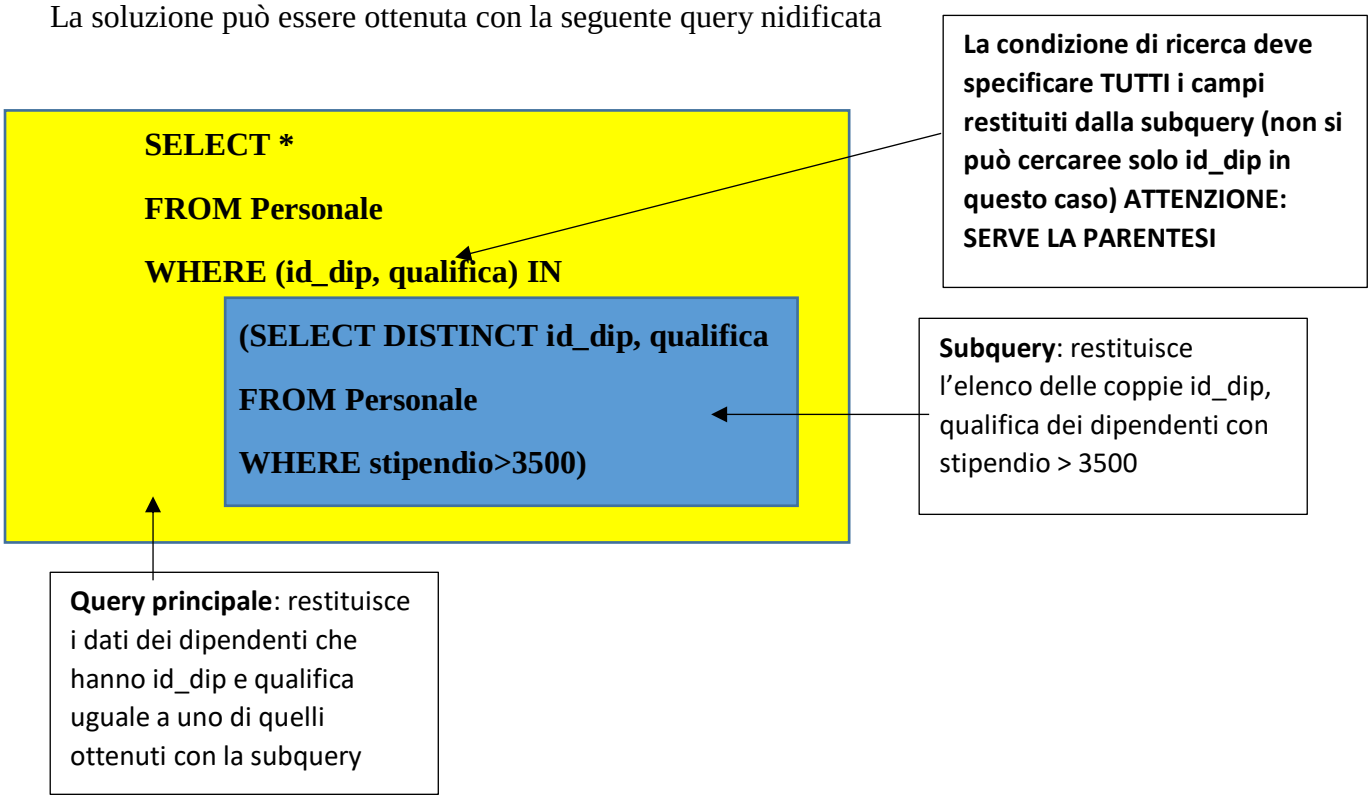
Esempio 2: si vuole conoscere il nome dei prodotti che usano 2 unità del componente C002. Si utilizza la seguente query:



L'operatore IN può operare non solo su un singolo campo, come nel caso precedente, ma anche su più campi come nel seguente esempio. In pratica consente di cercare delle righe all'interno della tabella restituita dalla subquery.

Esempio 3: si vuole selezionare i dati (tutti) degli impiegati che hanno lo stesso **id_dipartimento** e la stessa **qualifica** di tutti coloro che hanno uno stipendio maggiore di 3500€.

La soluzione può essere ottenuta con la seguente query nidificata



Attenzione: le **subquery** possono essere utilizzate solo all'interno della clausola **WHERE** o della clausola **FROM** (vedremo appositi casi).

Le **query nidificate** sono UNO dei due modi per estrarre informazioni da più tabelle. L'altro modo è quello di unire le tabelle da cui si vuole estrarre i dati attraverso operazioni di **join** (mostrate di seguito).

Quale delle due tecniche è meglio utilizzare? Le prestazioni dipendono sia dal tipo di query, sia dal DBMS Engine utilizzato, in quanto engine (motori) diversi possono essere ottimizzati per ricerche diverse.

Per ora, come criterio di scelta della query, cerchiamo di individuare query che forniscano dati corretti e che siano scritte nel modo più semplice possibile. Pertanto, ognuno scelga la modalità che preferisce. L'importante è che i risultati della query sia corretta.

JOIN IN SQL

Le operazioni sulle tabelle vengono identificate per mezzo della seguente sintassi:

π indica la **proiezione**, le **colonne** che si vogliono ottenere, è ciò che viene indicato immediatamente dopo SELECT, ad esempio:

$\pi_{\text{matricola,nominativo}}$ **T2** significa**:** "mostra solo le colonne matricola e nominativo della tabella T2

σ indica la **selezione**, le **righe** che si vogliono ottenere, è ciò che viene indicato immediatamente dopo WHERE, ad esempio:

$\sigma_{\text{provincia='MI'}}$ **T1** significa**:** "mostra solo le RIGHE IN CUI la colonna "provincia" ha valore "mi" di T1

$\bowtie$ indica la **congiunzione (o join)**. Il join è una sequenza di un prodotto cartesiano e di una selezione, ad esempio:

T1 $\bowtie_{\text{(Personale.id\_dip = Dipartimenti.id\_dip)}}$ **T2** significa: fai il prodotto cartesiano fra T1 e T2 mantenendo solamente le righe in cui id_dip di T1 è uguale a id_dip di T2

Primo Esempio: seleziona tutti i dipendenti (matricola e nominativo) che lavorano in provincia di Milano. Poichè l'informazione sulla provincia in cui hanno sede i dipartimenti è nella tabella Dipartimenti, mentre le informazioni su matricola e nominativo sono nella tabelle Personale, le due tabelle da "unire" con il JOIN sono proprio queste due. In seguito al join selezionerò solo le righe in cui la provincia è = 'MI', ulteriormente in seguito farò la proiezione per mostrare le sole colonne che voglio visualizzare (matricola e nominativo)

1. **T1** $\leftarrow$ **Personale** $\bowtie_{\text{(Personale.id\_dip = Dipartimenti.id\_dip)}}$ Dipartimenti
2. **T2** $\leftarrow \sigma_{\text{provincia='MI'}}$ **T1**
3. **T3** $\leftarrow \pi_{\text{matricola,nominativo}}$ **T2**

Secondo Esempio esplicativo

Vediamo come si svolge l'operazione di Join in SQL.

Selezionare il nome dei prodotti che sono costituiti da 2 componenti con id_comp=C002.

Questo esempio può essere risolto utilizzando le seguenti operazioni relazionali:

1. **T1** $\leftarrow$ **Prodotti** $\bowtie_{\text{(Prodotti.id\_prod = Composizione.id\_prod)}}$ Composizione
2. **T2** $\leftarrow \sigma_{\text{id\_comp=C002 AND unità\_comp=2}}$ **T1**
3. **T3** $\leftarrow \pi_{\text{nome\_prodotto}}$ **T2**

Ipotizziamo di avere le seguenti tabelle Prodotti e Composizione

Prodotti:

id_prod	id_dip	nome_prodotto	prezzo
P001	D1	Prodotto1	5
P002	D1	Prodotto2	7
P003	D2	Prodotto3	3
P004	D4	Prodotto4	5
P005	D5	Prodotto5	5

Composizione:

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

Vediamo i risultati di ciascuna delle 3 operazioni relazionali:

1. **T1 ← Prodotti ⋈ (Prodotti.id_prod = Composizione.id_prod) Composizione**

id_prod	id_dip	nome_prodotto	prezzo	id_prod	id_comp	unita_comp
P001	D1	Prodotto1	5	P001	C001	2
P001	D1	Prodotto1	5	P001	C002	2
P002	D1	Prodotto2	7	P002	C001	1
P002	D1	Prodotto2	7	P002	C003	3
P002	D1	Prodotto2	7	P002	C004	3
P003	D2	Prodotto3	3	P003	C002	2
P004	D4	Prodotto4	5	P004	C008	2
P004	D4	Prodotto4	5	P004	C009	2
P004	D4	Prodotto4	5	P004	C010	1
P004	D4	Prodotto4	5	P004	C015	1
P005	D5	Prodotto5	5	P005	C012	7
P005	D5	Prodotto5	5	P005	C013	3

2. **T2 ← σ id_comp=C002 AND unità_comp=2 T1**

id_prod	id_dip	nome_prodotto	prezzo	id_prod	id_comp	unita_comp
P001	D1	Prodotto1	5	P001	C001	2
P001	D1	Prodotto1	5	P001	C002	2
P002	D1	Prodotto2	7	P002	C001	1
P002	D1	Prodotto2	7	P002	C003	3
P002	D1	Prodotto2	7	P002	C004	3
P003	D2	Prodotto3	3	P003	C002	2
P004	D4	Prodotto4	5	P004	C008	2
P004	D4	Prodotto4	5	P004	C009	2
P004	D4	Prodotto4	5	P004	C010	1
P004	D4	Prodotto4	5	P004	C015	1
P005	D5	Prodotto5	5	P005	C012	7
P005	D5	Prodotto5	5	P005	C013	3

quindi:

id_prod	id_dip	nome_prodotto	prezzo	id_prod	id_comp	unita_comp
P001	D1	Prodotto1	5	P001	C002	2
P003	D2	Prodotto3	3	P003	C002	2

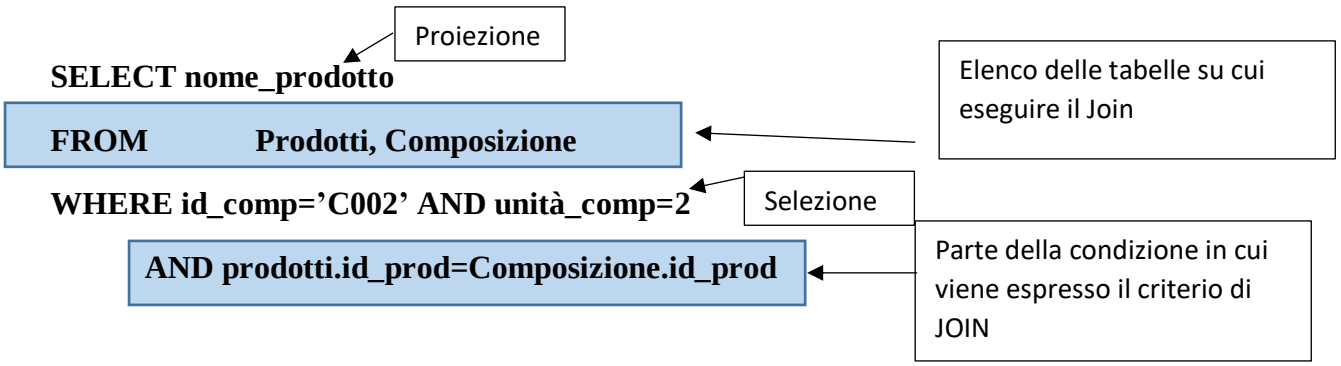
3. **T3 ← π $_{\text{nome\_prodotto}}$ T2**

id_prod	id_dip	nome_prodotto	prezzo	id_prod	id_comp	unita_comp
P001	D1	Prodotto1	5	P001	C002	2
P003	D2	Prodotto3	3	P003	C002	2

Quindi:

nome_prodotto
Prodotto1
Prodotto3

La sintassi per svolgere in SQL le operazioni è la seguente:



La clausola WHERE contiene quindi, oltre al criterio di selezione, anche il criterio di JOIN. In pratica **PRIMA la clausola FROM** esegue il prodotto cartesiano fra le due tabelle:

FROM Prodotti, Composizione

e **POI la clausola WHERE** contiene il criterio di selezione per realizzare il JOIN (con la *dot notation*)

Prodotti.id_prod=Composizioni.id_prod

si ricorda, infatti, **che il JOIN è dato da un prodotto cartesiano seguito da una selezione**.

La *dot notation* è sempre necessaria quando il criterio riguarda campi con lo stesso nome in tabelle diverse, è facoltativa altrimenti. Spesso è utile, per rendere più leggibile la query, ricorrere a degli alias.

Abbiamo visto che uno stesso problema può essere risolto in due modi diversi, con le query nidificate o con i JOIN. Ma quale dei due metodi è preferibile? Non esiste una regola generale, anche perché l'efficienza dell'operazione dipende da come il *motore* del DBMS (ossia quella parte di DBMS che si occupa di ricercare di dati) è ottimizzato per la ricerca dei dati, e ciò può variare da un DBMS ad un altro. Come criteri per la scelta di una soluzione piuttosto che un'altra utilizzeremo la semplicità di scrittura della query (visto che come prima richiesta è necessario che la query fornisca i risultati corretti e quindi se è più semplice vi sono meno possibilità di errore) e l'ottimizzazione in termini di risultati intermedi come già visto nell'algebra relazionale.

Esercizio (fare loro): risolvere l'esempio precedentemente svolto con query nidificate, attraverso join:

Seleziona i dati di tutti i dipendenti che lavorano nella provincia di Milano

Join fra più di due tabelle

L'operazione di Join può essere eseguita anche fra più di due tabelle, in tal caso considerando un JOIN fra N tabelle è necessario esprimere N-1 criteri di Join.

Primo esempio:

Si vuole ottenere il **nome** dei prodotti che utilizzano il componente che si chiama Componente03.

Dove si trovano le informazioni?

- Il nome dei prodotti si trova nella tabella Prodotti
- Il nome del componente si trova nella tabella Componenti
- L'informazione che lega Componenti e Prodotti, vale a dire l'indicazione di quali sono i componenti utilizzati da un prodotto si trovano nella tabella che unisce Prodotti e Componenti, vale a dire nella tabella Composizione.

Per estrarre l'informazione sarà dunque necessario effettuare un Join fra tutte e tre le tabelle

```
SELECT nome_prodotto FROM

prodotti, composizione, componenti

WHERE prodotti.id_prod=composizione.id_prod AND
composizione.id_comp=componenti.id_comp

AND nome_componente='Componente03';
```

Esempio:

si vuole ottenere il nome dei dipartimenti che hanno utilizzato il componente C003 nei loro prodotti.

Soluzione:

SELECT nome_dipartimento

FROM Dipartimenti, Prodotti, Composizione

WHERE id_comp=’C003’ AND

Dipartimenti.id_dip=Prodotti.id_dip AND Prodotti.id_prod=Composizione.id_prod;

Generalmente la condizione di Join è una condizione di **uguaglianza (in tal caso il join è chiamato Equi Join) fra la chiave primaia di una tabella e la chiave esterna** di un’altra tabella. Questa consuetudine non è però una regola, è possibile indicare altri criteri di join (maggiore di, minore di ecc.) oppure non indicare alcun criterio di Join fra due tabelle realizzando così il prodotto cartesiano.

Inoltre generalmente le chiavi primarie ed esterne utilizzate per il JOIN sono legate da integrità referenziale, ma anche questo non è un vincolo per eseguire un JOIN, l’unico vincolo per realizzare JOIN è quello che i due campi siano dello stesso tipo.

Esercizio fare loro. Scrivere la query per selezionare il costo unitario di ciascun componente dei prodotti Prodotto1 e Prodotto2. Il risultato deve fornire: il nome del prodotto, il nome del componente e il costo unitario del componente.

Altra sintassi per il JOIN

Un'altra sintassi per svolgere le operazioni di JOIN è quella che prevede l’utilizzo della parola chiave **INNER JOIN** da utilizzare sempre all’interno della clausola **FROM** specificando, sempre in questa clausola, la condizione di JOIN. Nella clausola WHERE andranno così indicate solamente le clausole per la selezione. Questa sintassi oltre ad essere più chiara perché separa la clausola di congiunzione dalla clausola di selezione, facilita il DB Engine nell’ottimizzazione per la ricerca di dati. Inoltre l’utilizzo di questa sintassi aiuta a non dimenticarsi di scrivere la clausola di JOIN, cosa che spesso avviene.

Ecco come possono essere risolti due esempi già visti utilizzando l’ INNER JOIN.

Esempio 1: conoscere il nome dei prodotti che usano 2 unità del componente C002 si utilizza la seguente query:

SELECT nome_prodotto

FROM (Prodotti INNER JOIN Composizione ON Prodotti.id_prod=Composizione.id_prod)

WHERE id_comp=’C002’ AND unità_comp=2;

Esempio2: nome dei dipartimenti che utilizzano il componente con id C002:

SELECT nome_dipartimento

FROM (

Dipartimenti INNER JOIN (Prodotti INNER JOIN Composizione ON Prodotti.id_prod=Composizione.id_prod) ON Dipartimenti.id_dip=Prodotti.id_dip

)

WHERE id_comp='C002';

Quando il campo su cui viene svolta la congiunzione ha lo stesso nome nelle due tabelle, come nell'esempio visto, SQL consente una sintassi più compatta per l'INNER JOIN utilizzano la parola chiave **USING (nome campo)** per specificare il campo utilizzato per il criterio di JOIN **(ricordarsi di mettere il nome del campo fra parentesi)**:

SELECT nome_dipartimento

FROM

(Dipartimenti INNER JOIN (Prodotti INNER JOIN Composizione USING (id_prod)) USING id_dip)

WHERE id_comp='C002';

Self Join

Il Self Join è una congiunzione di una tabella con se stessa**. Si usa per esprimere una condizione di ricerca fra righe diverse della stessa tabella**. Per eseguire il self join è utile avere a disposizione più viste della stessa tabella, tali diverse viste sono chiamate ***alias***, un alias è ottenuto assegnando un nome alla tabella tramite la parola chiave AS, come nel seguente esempio:

Esempio:

si vogliono ottenere i codici di tutti i prodotti che impiegano entrambi i componenti C001 e C002.

La tabella che contiene tali informazioni è Composizione:

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

Il risultato della query deve restituire solamente P001(è l'unico prodotto che impiega sia C001 che C002).

La prima idea (sbagliata) potrebbe essere quella di realizzare una query del tipo:

SELECT cod_prod FROM Composizione WHERE id_comp='C001' AND id_comp='C002'

non si ottiene alcuna tupla, questo perché l'operazione di selezione analizza UNA RIGA ALLA VOLTA e restituisce le sole righe che SODDISFANO la condizione, ma la condizione non è mai verificata perché per ogni tupla il valore di id_comp è uno solo, quindi non può essere contemporaneamente id_comp='C001' AND id_comp='C002

creando due tabelle uguali X e Y:

X

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

Y

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

Svolgendo un JOIN (Equi Join) sul campo id_prod si ottiene una tabella in cui, per ogni prodotto, ci sono tutte le possibili combinazioni dei componenti a 2 a 2. Da questa tabella possiamo selezionare le combinazioni in cui il primo componente è C001 e il secondo è C002

id_prod	id_comp	unita_comp	id_prod	id_comp	unita_comp
P001	C001	2	P001	C001	2
P001	C001	2	P001	C002	2
P001	C002	2	P001	C001	2
P001	C002	2	P001	C002	2
P002	C001	1	P002	C001	1
P002	C001	1	P002	C003	3
P002	C001	1	P002	C004	3
P002	C003	3	P002	C001	1
P002	C003	3	P002	C003	3
P002	C003	3	P002	C004	3
P002	C004	3	P002	C001	1
P002	C004	3	P002	C003	3
P002	C004	3	P002	C004	3
P003	C002	2	P003	C002	2
P004	C008	2	P004	C008	2
P004	C008	2	P004	C009	2
P004	C008	2	P004	C010	1
P004	C008	2	P004	C015	1
P004	C009	2	P004	C008	2
P004	C009	2	P004	C009	2
P004	C009	2	P004	C010	1
P004	C009	2	P004	C015	1
P004	C010	1	P004	C008	2
P004	C010	1	P004	C009	2
P004	C010	1	P004	C010	1

La query corretta è dunque:

SELECT X.id_prod

FROM Composizione AS X, Composizione AS Y

WHERE X.id_comp='C001' AND Y.id_comp='C002'

AND X.id_prod=Y.id_prod

Riassumendo si può dire che il self join fra DUE tabelle mostra tutte le combinazioni A DUE A DUE delle rigeh della tabella che soddisfano il criterio di join. Un self join fra TRE tabelle mostra tutte le combinazioni, a TRE a TRE, di tutte le righe della tabella, che rispettano il criterio di join E così via…. Pertanto viene utilizzato per verificare una condizione relativa a più righe della stessa tabella.

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

Vediamo altre due possibili soluzioni.

Soluzione con query nidificata

```
SELECT id_prod
FROM Composizione
WHERE id_comp='C001'
    AND id_prod IN (
        SELECT id_prod
        FROM Composizione AS T
        WHERE id_comp='C002');
```

id_prod
P001
P003

La query nidificata deve restituire un solo campo visto che cerco solamente id_prod

Soluzione con due subquery e un JOIN, **in questo caso le subquery inserite nella clausola FROM** anziché nella clausola WHERE.

va indicata una delle due tabelle altrimenti SQL segnala un errore perché non sa se prendere il campo id_prod di T1 o di T2

```
SELECT T1.id_prod
FROM (SELECT * FROM Composizione WHERE id_comp='C001') AS T1,
    (SELECT * FROM Composizione WHERE id_comp='C002') AS T2
WHERE T1.id_prod=T2.id_prod ;
```

attenzione: mettere l'alias fuori dalle parentesi quando usi una subquery nel FROM.

T1

id_prod	id_comp	unita_comp
P001	C001	2
P002	C001	1

T2

id_prod	id_comp	unita_comp
P001	C002	2
P003	C002	2

JOIN:

id_prod	id_comp	unita_comp	id_prod	id_comp	unita_comp
P001	C001	2	P001	C002	2

Selezione

Come già detto le subquery possono essere inserite nella clausola FROM o nella clausola WHERE (con IN).

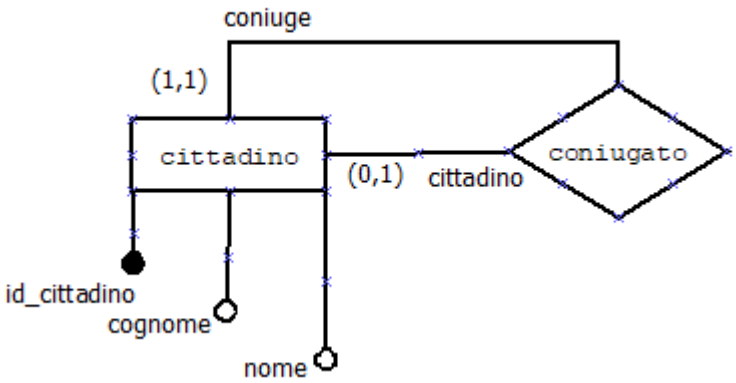
Non esiste un limite **teorico** alla nidificazione delle subquery, ogni nidificazione però rallenta le operazioni di ricerca dei dati.

Le **subquery** quindi sono delle query che danno come risultato delle tabelle temporanee che possono essere utilizzate sia all'interno della clausola **WHERE** sia della clausola **FROM**. Quando sono utilizzate nella clausola **FROM** è necessario utilizzare un **ALIAS** da assegnare al risultato della subquery.

Generalmente, quando possibile, è più intuitivo l'utilizzo dei una subquery anziché di un self Join, però non sempre le due operazioni sono equivalenti. Il self Join sostanzialmente viene utilizzato quando si vogliono **CONFRONTARE FRA LORO** righe della stessa tabella. Per poter fare questo è necessario duplicare le righe, creare un prodotto cartesiano, seguito poi da una selezione (confronto fra righe). Vediamo due esempi:

Esempio1 (fare insieme): database cittadini

Supponiamo di avere un database che rappresenta la seguente situazione:



Rappresentiamo la situazione con un database con una sola tabella cittadini che presenta una chiave esterna "id_coniuge" associata alla propria chiave primaria:

cittadini: (**id_cittadino**, cognome, nome, id_coniuge) FK: id_coniuge → cittadini.id_cittadino

Costruiamo insieme la tabella e aggiungiamo due cittadini sposati e uno no.

id_cittadino	cognome	nome	id_coniuge
1	Pinna	Luciano	2
2	Cassia	Laura	1
3	Scapolone	Antonio	*NULL*

Si vuole visualizzare, per ogni cittadino coniugato, il proprio cognome, nome e il cognome, nome del coniuge.

SOLUZIONE:

Con un SELF JOIN fra PK e FK si ottiene una tabella che contiene tutte le coppie di cittadini sposati fra loro:

id_cittadino	cognome	nome	id_coniuge	id_cittadino	cognome	nome	id_coniuge
2	Cassia	Laura	1	1	Pinna	Luciano	2
1	Pinna	Luciano	2	2	Cassia	Laura	1

Facendo una proiezione mostrando solo i nomi si ottiene il risultato

cognome	nome	cognome	nome
Cassia	Laura	Pinna	Luciano
Pinna	Luciano	Cassia	Laura

SELECT cittadini.cognome, cittadini.nome, coniuge.cognome, cconiuge.nome

FROM cittadini, cittadini as coniugi

WHERE cittadini.id_coniuge=coniuge.id_cittadino

La coppia viene mostrata due volte perché per OGNI cittadino viene mostrato il coniuge.

Esempio 2: (fare insieme) chi in D1 guadagna più di chi in D2?

Si vuole sapere quali sono le persone del dipartimento D1 che guadagno più di alcune persone (e si vuole sapere quali) del dipartimento to D2.

La soluzione è quella di:

1. duplicare la tabella personale chiamandola x e y
2. Svolgere il prodotto cartesiano,
3. Selezionare solo le righe in cui x.stipendio>y.stipendio
4. Selezionare solo le righe di x in cui id_dip=’D1’ e solo le righe di y in cui id_dip=’D2’

Le operazioni 2 e 3 svolgono un’operazione di join chiamato THETA JOIN di cui daremo in seguito la definizione

SOLUZIONE:

SELECT x.nominativo,x.stipendio,y.nominativo,y.stipendio

FROM personale as x, personale as y

WHERE x.id_dip='D1' AND y.id_dip='D2'

AND x.stipendio>y.stipendio;

Esercizio (fare loro): selezionare nominativo e data di nascita di tutti i dipendenti più giovani di VERDI MARCO

Una prima soluzione è quella di utilizzare la query nidificata:

Mostrare tutti i dipendenti più giovani di VERDI MARCO.

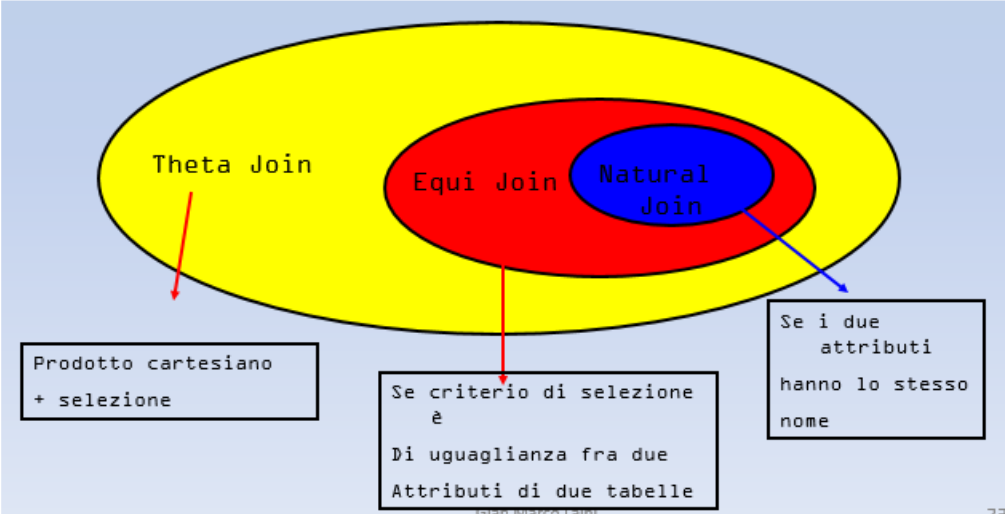
```
SELECT nominativo, data_nascita
FROM personale
WHERE data_nascita>(select data_nascita
FROM personale
WHERE nominativo="verdi marco")
```

Altra soluzione: si crea un alias della tabella personale (personale2) e si svolge un theta join, ossia un prodotto cartesiano fra personale e personale2, seguito da una selezione che seleziona solamente le righe in cui data_nascita della tabella 1 > data_nascita della tabella 2 in cui è stato selezionato solo "verdi marco"

```
SELECT personale.nominativo, personale.data_nascita
FROM personale, personale as personale2
WHERE personale2.nominativo="verdi marco" AND
personale.data_nascita>personale2.data_nascita
```

I meravigliosi nomi dei Join

Con il termine Join si indica l'operazione che porta alla congiunzione fra tabelle. In base al criterio utilizzato per effettuare tale operazione, i JOIN vengono indicati con termini diversi come spiegato nella seguente immagine:



Poiché nell'ultimo esercizio visto la congiunzione è stata svolta fra due tabelle utilizzando il criterio che **personale.data_nascita>personale2.data_nascita,** ciò che è stato realizzato è un THETA JOIN.

Join Esterni (Outer Join)

Le operazioni di JOIN fra due tabelle viste finora forniscono come risultato una tabella in cui vengono scartate le righe nelle quali non viene verificato il criterio di join. Questo comportamento può generare delle situazioni in cui il JOIN "perde" alcuni dati. Per evitare questo è possibile ricorrere ai JOIN ESTERNI con le parola chiave LEFT OUTER JOIN o LEFT JOIN e RIGHT OUTER JOIN o RIGHT JOIN. I JOIN ESTERNI, agiscono mantenendo **tutte le righe** della prima tabella (LEFT JOIN) o della seconda tabella (RIGHT JOIN) nel risultato della query, anche se tali righe non soddisfano il criterio di JOIN. Il JOIN visto finora (senza le parole chiave LEFT o RIGHT) è detto JOIN INTERNO.

Esempio: supponiamo sia presente un dipartimento D5 senza alcun dipendente. Si vogliono visualizzare i nomi di tutti i Dipartimenti e di ciascun dipendente che vi lavora. La seguente query non è corretta in quanto non mostra il dipartimento D5, a causa del fatto che il JOIN non trova alcuna corrispondenza fra dell'id_dip nella tabella dei Dipendenti

SELECT dipartimenti.nome, nominativo

FROM (Dipartimenti, Personale)

WHERE Dipartimenti.id_dip=Personale.id_dip

ORDER BY dipartimenti.id_dip;

nome	nominativo
primo dipartimento	ROSSI PIERO
primo dipartimento	BIANCHI MAURO
secondo dipartimento	NERI GIOVANNI
secondo dipartimento	VERDI MARCO
secondo dipartimento	PIERINI MARIO
secondo dipartimento	CARLETTI PAOLO
terzo dipartimento	BELLI DANIELA
terzo dipartimento	SANDRI DONATA LUIGINA
terzo dipartimento	GIANNINI PIETRO
terzo dipartimento	TESINI MARIO
terzo dipartimento	LAPINI PAOLO
quarto dipartimento	ARNETTI ENNIO
quarto dipartimento	SOLDANI GIULIO

Il problema è che non viene visualizzato il nome del dipartimento D5

Per consentire di visualizzare i dati del dipartimento D5 anche se non ha nessun dipendente, si utilizza il join sinistro

necessario USING o ON

SELECT dipartimenti.nome, nominativo

FROM Dipartimenti LEFT JOIN Personale USING (id_dip)

ORDER BY id_dip;

nome	nominativo
primo dipartimento	ROSSI PIERO
primo dipartimento	BIANCHI MAURO
secondo dipartimento	NERI GIOVANNI
secondo dipartimento	VERDI MARCO
secondo dipartimento	PIERINI MARIO
secondo dipartimento	CARLETTI PAOLO
terzo dipartimento	BELLI DANIELA
terzo dipartimento	SANDRI DONATA LUIGINA
terzo dipartimento	GIANNINI PIETRO
terzo dipartimento	TESINI MARIO
terzo dipartimento	LAPINI PAOLO
quarto dipartimento	ARNETTI ENNIO
quarto dipartimento	SOLDANI GIULIO
quinto dipartimento	*NULL*

Se invece si aggiungesse alla tabella personale un dipendente in cui il campo id_dip non fosse valorizzato (se i vincoli di integrità lo permettono), per visualizzare i dati di tutti i dipendenti e dei loro dipartimenti, visualizzando anche il dipendente senza dipartimento, è necessario ricorrere ad un join destro.

SELECT nominativo, dipartimenti.nome as dipartimento

FROM Dipartimenti RIGHT JOIN Personale ON Dipartimenti.id_dip=Personale.id_dip

ORDER BY dipartimenti.id_dip;

nominativo	dipartimento
BRESCHI CARLA	*NULL*
ROSSI PIERO	primo dipartimento
BIANCHI MAURO	primo dipartimento
NERI GIOVANNI	secondo dipartimento
PIERINI MARIO	secondo dipartimento
CARLETTI PAOLO	secondo dipartimento
VERDI MARCO	secondo dipartimento
LAPINI PAOLO	terzo dipartimento
SANDRI DONATA LUIGINA	terzo dipartimento
GIANNINI PIETRO	terzo dipartimento
BELLI DANIELA	terzo dipartimento
TESINI MARIO	terzo dipartimento
ARNETTI ENNIO	quarto dipartimento
SOLDANI GIULIO	quarto dipartimento

Per ottenere il **FULL OUTER JOIN**, ossia una tabella che contenga sia i risultati del LEFT JOIN che del RIGHT JOIN, è necessario utilizzare la parola chiave UNION, che permette di implementare in SQL l'operazione insiemistica dell'unione, poiché il DBMS MariaDB (come molti altri DBMS) non prevede un operatore SQL che realizzi il FULL JOIN.

SELECT nominativo, nome as dipartimento FROM dipartimenti RIGHT OUTER JOIN personale USING (id_dip)

UNION

SELECT nominativo, nome as dipartimento FROM dipartimenti LEFT OUTER JOIN personale USING (id_dip)

nominativo	dipartimento
ROSSI PIERO	primo dipartimento
NERI GIOVANNI	secondo dipartimento
ARNETTI ENNIO	quarto dipartimento
BELLI DANIELA	terzo dipartimento
VERDI MARCO	secondo dipartimento
SANDRI DONATA LUIGINA	terzo dipartimento
GIANNINI PIETRO	terzo dipartimento
TESINI MARIO	terzo dipartimento
BIANCHI MAURO	primo dipartimento
PIERINI MARIO	secondo dipartimento
CARLETTI PAOLO	secondo dipartimento
SOLDANI GIULIO	quarto dipartimento
LAPINI PAOLO	terzo dipartimento
BRESCHI CARLA	*NULL*
NULL	quinto dipartimento

I risultati in comune alle due query unita da UNION vengono scritti una sola volta

Esempio interessante che evidenzia l'utilità del join esterno (fare insieme). Mostrare il nome dei dipartimenti che non hanno alcun dipendente.

Soluzione 1 NON CORRETTA:

Con una subquery seleziono tutti gli "id_dip" presenti nella tabella Personale. Con la query esterna seleziono dalla tabella dipartimenti, i nomi di tutti i dipartimenti che non sono presenti nel risultato della subquery

SELECT nome_dipartimento FROM dipartimenti

WHERE id_dip NOT IN (SELECT DISTINCT id_dip FROM personale2)

Attenzione: non funzione se ho un dipendente che non è assegnato ad alcun dipartimento e quindi un risultato null nella query interna.

In generale, la presenza di valori NULL in un elenco di valori in cui viene ricercato un valore con IN o NOT IN è da evitare poiché NULL in SQL viene sempre considerato come un valore sconosciuto, quindi ogni confronto con NULL restituisce un risultato indeterminato.

Soluzione 2 con left join

SELECT DISTINCT nome_dipartimento FROM

dipartimenti LEFT JOIN personale2 USING (id_dip)

WHERE ISNULL (matricola)

Esercizio (fare loro): Determinare i nomi dei dipartimenti in cui non è stato realizzato alcun prodotto.

Operatori ALL, ANY

Si utilizzano nelle subquery della clausola WHERE, quando la subquery restituisce più di un valore. **ALL** permette di ottenere da una tabella le tuple che soddisfano una certa relazione di confronto (maggiore di, uguale a, minore di) rispetto a di TUTTI i valori restituiti dalla subquery, **ANY** permette di ottenere da una tabelle le tuple che soddisfano una relazione di confronto con ALMENO UNO dei valori restituiti dalla subquery.

Esempio per ALL:

si vuole conoscere matricola e nominativo del dipendente con lo stipendio più basso.

Soluzione:

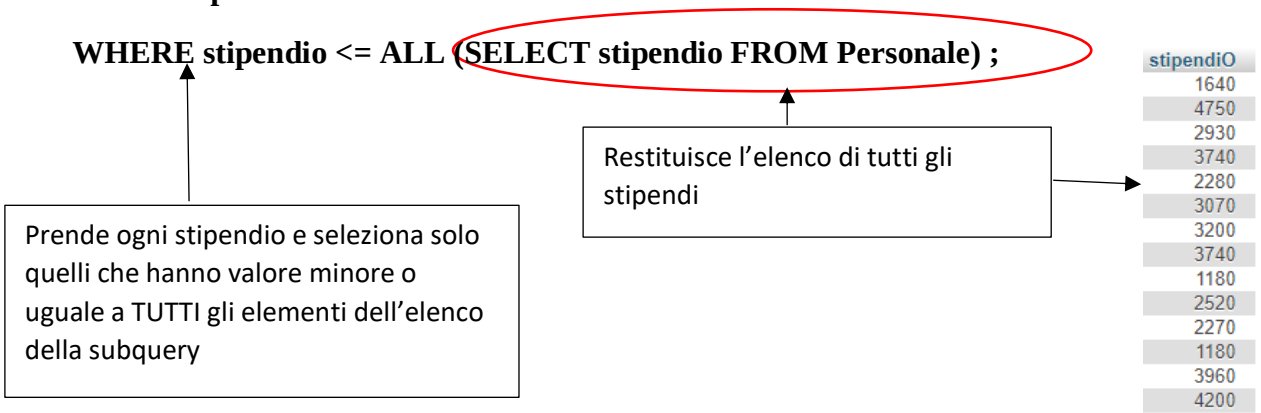
- Con una query interna si selezionano tutti i valori di stipendio da Personale
- Con la query esterna si cerca il dipendente il cui valore è minore o uguale a TUTTI quegli stipendi.

Attenzione! E' necessario che non sia presente un dipendente senza stipendio (stipendio NULL) perché in tal caso l'espressione stipendio <= Null risulta falsa, quindi non viene rilevato nessun dipendente che abbia stipendio <= a tutti gli stipendi.

SELECT matricola, nominativo

FROM Dipendenti

WHERE stipendio <= ALL (SELECT stipendio FROM Personale) ;



Si può fare in altro modo ? SI (vedremo la funzione MIN)

SELECT matricola, nominativo

FROM personale

WHERE stipendio=(SELECT min(stipendio) from personale);

Esempio per ANY:

Selezionare nominativo di tutti i dipendenti che hanno un'età maggiore di uno qualsiasi dei dipendenti del dipartimento D1

```
SELECT nominativo, data_nascita, id_dip from personale WHERE
data_nascita <any (SELECT data_nascita FROM personale WHERE id_dip="D1")
AND id_dip!= "D1";
```

la clausola WHERE AND id_dip!="D1" serve per escludere dai risultati i dipendenti del dipartimento 'D1'

Si può fare in altro modo? SI (vedremo la funzione MAX)

Si seleziona nella subquery la data di nascita del più giovane del dipartimento D1 e nella query esterna tutti coloro che hanno data_nascita minore di quella

```
SELECT nominativo, id_dip
FROM personale WHERE data_nascita<(SELECT max(data_nascita) FROM
personale WHERE id_dip='D1'
AND id_dip!= "D1";
```

ESERCIZI con database azienda_dati_originale.sql

1. **Selezionare matricola, nominativo dei dipendenti che hanno lavorato alla produzione di almeno uno fra i seguenti prodotti: Prodotto1 e Prodotto2.**
2. **Selezionare i prodotti (nome_prodotto) che costano più del prodotto 3**
3. **Selezionare i nomi dei componenti che non sono utilizzati in alcun prodotto**
4. **Selezionare il nome dei dipartimenti che non hanno mai utilizzato il componente C002**
5. **Selezionare il nome del dipartimento che ha realizzato il prodotto più costoso**
6. **Selezionare i nomi dei prodotti che costano meno di uno qualsiasi dei prodotti realizzati dal dipartimento D1**
7. **Selezionare i nomi dei prodotti che costano più di tutti i prodotti realizzati dal dipartimento D1**
8. **Visualizzare i dipendenti che lavorano nella stessa provincia di Verdi Marco**
9. **Selezionare il nome dei dipartimenti in cui NON lavora Verdi Marco**
10. **Selezionare il nome dei dipartimenti in cui non lavora Verdi Marco ma lavora almeno un dipendente**
11. **Mostrare tutti i dipendenti (matricola, nominativo, stipendio, nome del dipartimento) dei dipartimenti in cui esiste almeno un dipendente con stipendio >=3000 €**
12. **Esercizio: mostrare il nome del prodotto, o dei prodotti, più costosi (in seguito vedremo come farlo con la funzione max)**
13. **Si vuole dare un premio ai dipendenti (visualizzare matricola e nominativo) che lavorano in un dipartimento che ha utilizzato il componente C002. Visualizzare tali dipendenti.**
14. **Mostrare i nomi dei dipartimenti in cui non lavora nessuno**

Esercizi analoghi DB Film

1. **Mostrare i dati degli attori che hanno recitato in almeno un film premiato (con qualsiasi premio)**
2. **Mostrare i dati degli attori che hanno recitato in almeno un film che ha vinto un premio oscar meno di 10 anni fa (la query deve valere in qualsiasi data essa viene eseguita). Si**
3. **Mostrare gli attori (nominativo e data di nascita) più vecchi di Diego Abatantuono**
4. **Mostrare i titoli dei film che non hanno colonna sonora**
5. **Mostrare i titoli dei film che non hanno mai vinto al festival di Cannes**
6. **Mostrare il titolo e l'anno di realizzazione del film (o dei film) più vecchio (più vecchi) (poi vedremo come farlo con la funzione di aggregazione min)**
7. **Mostrare gli attori (nominativo) che hanno partecipato al film (ai film) più recente (più recenti). Mostrare anche il nome del film**
8. **Selezionare le commedie che hanno avuto una valutazione maggiore di un qualsiasi film drammatico**
9. **Selezionare il titolo, l'anno e il genere di tutti i film realizzati prima di tutti i film horror**
10. **Selezionare gli attori che hanno lavorato in film con Elijah Wood (mostrare anche i titoli dei film)**
11. **Selezionare i film in cui NON recita Elijah Wood**

12. **Mostrare tutti i premi vinti dai film in cui recitano attori nati dopo il 1970 (mostrare il premio, l'anno in cui è stato vinto il premio, il titolo del film e l'anno di realizzazione del film)**
13. **Mostrare i film in cui recita l'attore più vecchio**
14. **Mostrare i film che non hanno alcun brano come colonna sonora**

2. LE FUNZIONI DI AGGREGAZIONE E LA CLAUSOLA DI RAGGRUPPAMENTO

FUNZIONI DI AGGREGAZIONE

Funzioni che agiscono su **insiemi di righe** di una tabella. Sono utilizzate come argomento **immediatamente accanto alla clausola SELECT**.

Le funzioni (COUNT, SUM, MAX, MIN, AVG) sono eseguite **solo sulle righe che soddisfano le condizioni indicate con la clausola WHERE**. Il significato delle funzioni è abbastanza intuitivo

- COUNT: conta le righe. Restituisce il numero di righe che soddisfano la clausola WHERE (righe selezionate)
- SUM: somma i valori. Restituisce la somma dei valori contenuti nelle righe selezionate, i valori da sommare possono riferirsi anche a diversi campi e ad operazioni aritmetiche sui valori dei campi.
- AVG: calcola la media. Restituisce la media dei valori contenuti nelle righe selezionate.
- MAX: restituisce il valore massimo fra quelli delle righe selezionate
- MIN: restituisce il valore minimo fra quelli delle righe selezionate

Si sottolinea che ciascuna di queste funzioni di aggregazione restituisce **UN SOLO VALORE** ottenuto dalle righe che soddisfano la clausola WHERE.

SQL consente di utilizzare contemporaneamente più funzioni di aggregazione in un singolo SELECT

I campi che contengono valori NULL non vengono conteggiati (in COUNT e AVG)

Esempio:

conoscere il prezzo minimo, massimo e medio di vendita dei prodotti del dipartimento D1

id_prod	id_dip	nome_prodotto	prezzo
P001	D1	Prodotto1	5
P002	D1	Prodotto2	7
P003	D2	Prodotto3	3
P004	D4	Prodotto4	5
P005	D5	Prodotto5	5

SELECT MIN(prezzo), MAX(prezzo), AVG(prezzo)

FROM prodotti

WHERE id_prod='D1';

La sintassi prevede che l'argomento della funzione di aggregazione sia un campo (o in alcuni casi più campi) e venga scritto fra parentesi tonde. Non lasciare spazio fra la funzione e la parentesi.

La sintassi prevede che le funzioni di aggregazione, quando sono più di una vengano separate dalla virgola

Risultato;

MIN(prezzo)	MAX(prezzo)	AVG(prezzo)
5	7	6

Per arrotondare il risultato di una funzione di aggregazione ad un certo numero di cifre decimali si utilizza la funzione ROUND:

SELECT ROUND(AVG(prezzo),3) FROM prodotti

Esempio:

conoscere l'importo complessivo degli stipendi dei dipendenti con qualifica Q5

SELECT SUM (stipendio)

FROM Personale

WHERE qualifica='Q5';

Descriviamo passo passo che accade:

1. La clausola WHERE seleziona tutti e soli i dipendenti con qualifica Q5 dalla tabella Personale

matricola	id_dip	nominativo	data_nascita	qualifica	stipendio
00013	D1	ROSSI PIERO	1965-01-15	Q2	1640
parte dei campi su essi.		ARNETTI ENNIO	1967-01-15	Q5	4750
0034	D2	NERI GIOVANNI	1974-08-05	Q4	2930
04346	D3	BELLI DANIELA	1977-02-25	Q5	3740
04434	D2	VERDI MARCO	1977-05-09	Q3	2280
04450	D3	SANDRI DONATA	1979-05-01	Q3	3070
04532	D3	GIANNINI PIETRO	1978-04-06	Q3	3200
04541	D3	TESINI MARIO	1978-12-28	Q4	3740
04551	D1	BIANCHI MAURO	1968-07-09	Q1	1180
04717	D2	PIERINI MARIO	1969-06-20	Q4	2520
04794	D2	CARLETTI PAOLO	1981-07-02	Q3	2270
05019	D4	SOLDANI GIULIO	1983-03-27	Q1	1180
05462	D3	LAPINI PAOLO	1977-01-11	Q5	3960
05477	D4	BRESCHI CARLA	1975-05-02	Q4	4200

matricola	id_dip	nominativo	data_nascita	qualifica	stipendio
00075	D4	ARNETTI ENNIO	1967-01-15	Q5	4750
04346	D3	BELLI DANIELA	1977-02-25	Q5	3740
05462	D3	LAPINI PAOLO	1977-01-11	Q5	3960

2. La funzione SUM somma tutti gli stipendi delle righe selezionate.

Esempio:

conoscere il numero di dipendenti del dipartimento D3

SELECT COUNT (*)

FROM Personale

WHERE id_dip='D3';

ATTENZIONE: se nella funzione di aggregazione COUNT si utilizza come argomento * (come nell'esempio) vengono contate TUTTE le righe che soddisfano la clausola WHERE. Si potrebbe

anche specificare come argomento di COUNT un campo specifico (ad esempio COUNT (nominativo)), in tal caso **NON verrebbero contate** le righe contenenti, per quel campo, il valore **NULL**. Inoltre, sempre riferendosi al caso di COUNT in cui si specifica un particolare campo, utilizzando la parola chiave DISTINCT all'interno della parentesi che specifica il campo da "contare", le righe in cui il campo specificato si ripete, sarebbero contate una sola volta.

Esempio:

il caso precedente potrebbe essere scritto indicando un campo qualsiasi della tabella Personale come argomento di COUNT

SELECT COUNT(qualifica)

FROM Personale

WHERE id_dip='D3';

se ci fosse un dipendente in cui il campo qualifica non è valorizzato (NULL) in questo caso non sarebbe contato.

Esempio:

conoscere il numero di dipartimenti che **hanno** dei dipendenti con qualifica Q5.

SELECT COUNT (DISTINCT id_dip)

FROM Personale

WHERE qualifica='q5';

ATTENZIONE: mettendo DISTINCT prima di COUNT il risultato non è corretto

ESERCIZI fare loro:

- indicare il prezzo del prodotto meno costoso
- Indicare il prezzo medio dei prodotti

Esempio di un possibile errore: selezionare il codice e il nome del prodotto che costa di meno.

Possibile errore, potrebbe sembrare intuitiva la seguente query:

SELECT id_prod,nome_prodotto,min(prezzo)
FROM prodotti

Ma in realtà questa query è sbagliata, infatti la funzione di aggregazione restituisce un solo valore e il valore restituito è valore minimo dell'attributo prezzo nella tabella (ossia 0). Il valore restituito dalla funzione di aggregazione non ha alcun legame con i primi due attributi selezionati, infatti il risultato mostrato sarà

id_prod	nome_prodotto	min(prezzo)
P001	Prodotto1	0

Anche se in realtà il prodotto1 ha prezzo=5!!!

Il DBMS ragiona nel seguente modo:

1. Svolge la funzione di aggregazione restituendo UNA SOLA riga (valore=0)
2. Va nella tabella prodotti e seleziona UNA SOLA RIGA, la prima che trova (Prodotto1) anche se non c'è alcun legame con il prezzo individuato dalla funzione di aggregazione!

Soluzione corretta: la funzione di aggregazione va indicata come subquery nella clausola WHERE

```
SELECT id_prod, nome_prodotto
FROM prodotti
WHERE prezzo=(SELECT MIN(prezzo) FROM prodotti)
```

ESERCIZI

- Selezionare il nome e il prezzo dei prodotti il cui prezzo è superiore al prezzo medio
- Indicare il numero di prodotti il cui prezzo è superiore al prezzo medio
- Indicare l'età media dei dipendenti che lavorano nei dipartimenti D2 e D3
- Indicare il nome del dipartimento che realizza il prodotto più costoso

CLAUSOLA DI RAGGRUPPAMENTO

E' la clausola GROUP BY, viene scritta **dopo** la clausola **WHERE**, e consente di partizionare in "gruppi" le righe selezionate in base all'uguaglianza di valori di alcuni attributi (stabiliti, appunto nella clausola di raggruppamento).

L'utilità del raggruppamento è quella di applicare a ciascun gruppo creato le funzioni di aggregazione. In questo modo si ottiene **UN** risultato della funzione di aggregazione per **OGNI** gruppo ottenuto.

Esempio:

si vuole conoscere lo stipendio medio per ogni dipartimento.

Per risolvere il problema:

1. si parte dalla tabella Personale e si raggruppano le righe per dipartimento (GROUP BY (id_dip)) ottenendo questo risultato:

   Gruppo 1:

matricola	id_dip ▲ 1	nominativo	data_nascita	qualifica	stipendio
00013	D1	ROSSI PIERO	1965-01-15	Q2	1640
04551	D1	BIANCHI MAURO	1968-07-09	Q1	1180

   Gruppo 2:

0034	D2	NERI GIOVANNI	1974-08-05	Q4	2930
04794	D2	CARLETTI PAOLO	1981-07-02	Q3	2270
04434	D2	VERDI MARCO	1977-05-09	Q3	2280
04717	D2	PIERINI MARIO	1969-06-20	Q4	2520

   Gruppo 3:

05462	D3	LAPINI PAOLO	1977-01-11	Q5	3960
04541	D3	TESINI MARIO	1978-12-28	Q4	3740
04532	D3	GIANNINI PIETRO	1978-04-06	Q3	3200
04450	D3	SANDRI DONATA	1979-05-01	Q3	3070
04346	D3	BELLI DANIELA	1977-02-25	Q5	3740

   Gruppo 4:

05019	D4	SOLDANI GIULIO	1983-03-27	Q1	1180
00075	D4	ARNETTI ENNIO	1967-01-15	Q5	4750
05477	D4	BRESCHI CARLA	1975-05-02	Q4	4200

2. Per ogni gruppo si calcola lo stipendio medio con la funzione di aggregazione AVG (AVG(stipendio)).

La query risulta quindi essere la seguente:

SELECT id_dip, AVG(stipendio) AS stipendio_medio

FROM Personale

GROUP BY (id_dip)

Risultato:

id_dip	stipendio_medio
D1	1410
D2	2500
D3	3542
D4	3376.6666666666665

Esempio:

ottenere il codice e il nome di ogni dipartimento e il **numero di componenti utilizzati (ogni componente deve essere contato una sola volta) da quel dipartimento per produrre i propri prodotti**:

SELECT id_dip,nome_dipartimento, COUNT(DISTINCT id_comp) AS numero_componenti

FROM ((dipartimenti INNER JOIN prodotti USING(id_dip)) INNER JOIN composizione ON composizione.id_prod=prodotti.id_prod)

GROUP BY id_dip,nome_dipartimento;

Risultato:

id_dip	nome_dipartimento	numero_componenti
D1	Primo dipartimento	4
D2	Secondo dipartimento	1
D4	Quarto diprtimento	4
D5	Quinto dipartimento	2

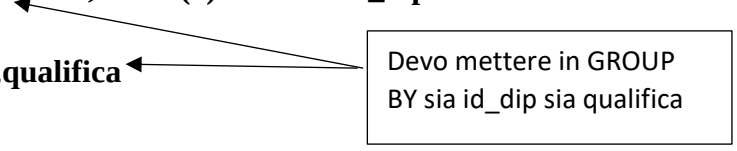
Osservazione importante:

Quando la clausola SELECT è seguita sia da una lista di attributi semplici sia da una funzione di aggregazione, è **necessario,** per ottenere risultati corretti, che la clausola GROUP BY effettui un raggruppamento in base (almeno) a tutti i campi presenti nel SELECT. Quindi 'almeno' tutti campi semplici di SELECT che generano un raggruppamento devono ritrovarsi nel GROUP BY.

Il motivo è chiarito dal seguente esempio:

Voglio mostrare quanti dipendenti ci sono per ogni dipartimento raggruppati per qualifica.

SELECT id_dip, qualifica, count(*) as numero_dipendenti
FROM personale
GROUP BY id_dip,qualifica



id_dip	qualifica	numero_dipendenti
D1	Q1	2
D2	Q3	1
D2	Q4	1
D2	Q5	2
D3	*NULL*	1
D3	Q1	2
D3	Q2	1
D3	Q3	1
D4	Q3	1
D4	Q4	2

Altrimenti, se raggruppassi solamente per id_dip otterrei un risultato sbagliato

SELECT id_dip, qualifica, count(*) as numero_dipendenti
FROM personale
GROUP BY id_dip

id_dip	qualifica	numero_dipendenti
D1	Q1	2
D2	Q5	4
D3	Q1	5
D4	Q4	3

Il risultato è **sbagliato**, sembra che nel dipartimento D2 ci siano 4 dipendenti tutti con qualifica Q5!

Altro esempio:

calcolare il costo unitario di produzione di ciascun prodotto come somma del costo unitario di ogni componente moltiplicato per la quantità utilizzata nel singolo prodotto:

SELECT prodotti.id_prod, prodotti.nome_prodotto,
SUM(componenti.costo_unitario*composizione.unita_comp) AS costo

FROM prodotti,composizione,componenti

WHERE prodotti.id_prod=composizione.id_prod AND
composizione.id_comp=componenti.id_comp

GROUP BY id_prod,nome_prodotto;

Risultato:

id_prod	nome_prodotto	costo
P001	Prodotto1	1.6
P002	Prodotto2	1.95
P003	Prodotto3	1
P004	Prodotto4	1.75
P005	Prodotto5	0.5

Attenzione: non mettere la parentesi all'elenco GROUP BY altrimenti non funziona).

Altro esempio:

Aggiungere alla query precedente un campo calcolato "guadagno" come differenza fra prezzo di ciascun prodotto e il costo

```
SELECT  prodotti.id_prod, prodotti.nome_prodotto,
SUM(componenti.costo_unitario*composizione.unita_comp) AS costo, prezzo -
SUM(componenti.costo_unitario*composizione.unita_comp) as guadagno

FROM prodotti,composizione,componenti

WHERE prodotti.id_prod=composizione.id_prod AND
composizione.id_comp=componenti.id_comp

GROUP BY id_prod,nome_prodotto;
```

id_prod	nome_prodotto	costo	guadagno
P001	Prodotto1	1.6	3.4
P002	Prodotto2	1.95	5.05
P003	Prodotto3	1	2
P004	Prodotto4	1.75	3.25
P005	Prodotto5	0.5	4.5

Approfondimento con uso di IF()

Nel caso ci fossero prodotti che non sono composti da alcun componente, essi non sarebbero riportati nella query precedente perché il JOIN fra prodotti e composizione li eliminerebbe.

Per visualizzare anche i prodotti realizzati senza alcun componente sono necessari due LEFT JOIN

```
SELECT nome_prodotto, sum(composizione.unita_comp*componenti.costo_unitario) as costo_prodotto, prezzo-sum(composizione.unita_comp*componenti.costo_unitario) as guadagno_prodotto

FROM (prodotti LEFT JOIN composizione using(id_prod)) LEFT JOIN componenti using(id_comp)

GROUP BY nome_prodotto
```

Per i prodotti senza componenti però, in questo caso, non viene calcolato il guadagno perché la sottrazione non viene svolta essendo il costo=NULL.

Visto che comunque anche questi prodotti danno un guadagno, se si volesse mostrare il loro guadagno (coincidente con il prezzo), si dovrebbe ricorrere ad una struttura IF. **La struttura IF può essere inserita nella clausola SELECT**. La procedura è questa:

Si determina il numero di componenti di cui è composto ciascun prodotto, se il numero di componenti è =0, viene mostrato come guadagno il valore di "prezzo", altrimenti il valore di "prezzo"-"calcolo del costo" (per leggere meglio la query è stata eliminata la parte che visualizza il costo):

```
SELECT nome_prodotto,count(id_comp) as numero_componenti,

if(count(id_comp)=0,prezzo,prezzo-sum(composizione.unita_comp*componenti.costo_unitario)) as guadagno_prodotto

FROM (prodotti LEFT JOIN composizione using(id_prod)) LEFT JOIN componenti using(id_comp)

GROUP BY nome_prodotto
```

Come abbiamo visto la clausola GROUP BY consente di raggruppare le tuple selezionate in gruppi in base al valore assunto da uno o più campi. Utilizzata insieme alle funzioni di aggregazione, GROUP BY, consente di ottenere un risultato calcolato sulle righe di ciascun gruppo creato (la somma dei valori, la media, il massimo, il minimo, il conteggio di quanti sono).

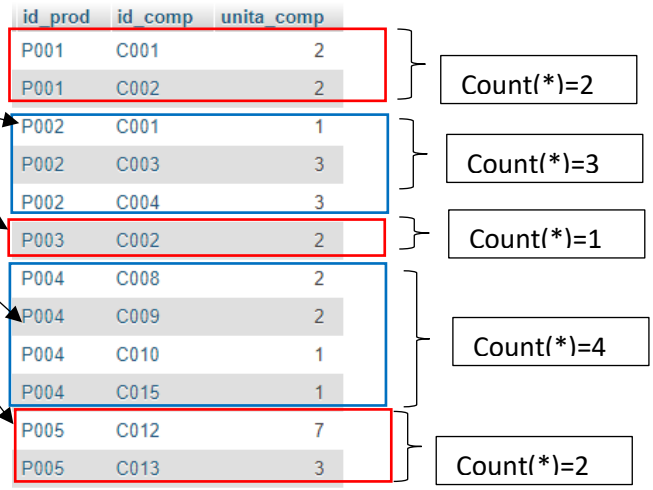
CLAUSOLA HAVING

In SQL esiste la possibilità di **selezionare solo alcuni dei gruppi realizzati con GROUP BY esprimendo delle condizioni sulle funzioni di aggregazione associate ai gruppi. Questo viene svolto attraverso la clausola HAVING**. Tale clausola consente quindi si esprimere una CONDIZIONE SU UN INTERO GRUPPO. La clausola HAVING può essere utilizzata solamente insieme ad un GROUP BY e va scritta nel comando SQL **successivamente** al **GROUP BY**.

Esempio:

elencare i codici dei prodotti che utilizzano almeno 3 componenti distinti.

```
SELECT id_prod, COUNT(*) as numero_componenti
FROM composizione
GROUP BY id_prod
HAVING numero_componenti>=3
```



Oppure:

```
SELECT id_prod, count (*)
FROM Composizione
GROUP BY id_prod
HAVING COUNT(*)>=3;
```

Risultato:

id_prod	numero_componenti
P002	3
P004	4

Si sottolinea la differenza fra le clausole WHERE e HAVING. La prima effettua una selezione sulle singole righe di una tabella. La seconda effettua una selezione fra i GRUPPI in cui è stata suddivisa una tabella, in base al risultato di una o più funzioni di aggregazione calcolate sui gruppi stessi.

Quando WHERE e HAVING sono presenti contemporaneamente in una query, prima viene effettuata la selezione con WHERE, poi le righe selezionate vengono raggruppate con GROUP BY, poi vengono calcolati i risultati delle funzioni di aggregazione sui vari gruppi, poi vengono selezionati i gruppi con la clausola HAVING.

Vediamo alcuni esempi.

Esempio1 (non su Having):

determinare il codice dei prodotti che utilizzano la maggior quantità del componente C002

Soluzione:

1. Dove trovo la quantità di componente 02 utilizzata da ogni prodotto?

   Nella tabella Composizione:

id_prod	id_comp	unita_comp
P001	C001	2
P001	C002	2
P002	C001	1
P002	C003	3
P002	C004	3
P003	C002	2
P004	C008	2
P004	C009	2
P004	C010	1
P004	C015	1
P005	C012	7
P005	C013	3

2. Individuo il valore massimo di C002 presente in un prodotto con la funzione di aggregazione MAX

```
SELECT MAX(unita_comp)
FROM Composizione
WHERE id_comp='C002';
```

   → **2**

   Questa query restituisce un valore (scalare) pari a 2.

3. Utilizzo la query precedente come subquery per determinare l'id_prod dei prodotti che utilizzano tale quantità di componente C002

```
SELECT id_prod
FROM Composizione
WHERE id_comp='C002' AND unita_comp=
(SELECT MAX(unita_comp)
FROM Composizione
WHERE id_comp='C002');
```

   →

id_prod
P001
P003

Esempio 2:

determinare il codice del dipartimento con lo stipendio medio maggiore e il valore di tale stipendio medio.

1. Determino gli stipendi medi dalla tabella Personale raggruppando per dipartimento

matricola	id_dip	nominativo	data_nascita	qualifica	stipendio
00013	D1	ROSSI PIERO	1965-01-15	Q2	1640
00075	D4	ARNETTI ENNIO	1967-01-15	Q5	4750
0034	D2	NERI GIOVANNI	1974-08-05	Q4	2930
04346	D3	BELLI DANIELA	1977-02-25	Q5	3740
04434	D2	VERDI MARCO	1977-05-09	Q3	2280
04450	D3	SANDRI DONATA	1979-05-01	Q3	3070
04532	D3	GIANNINI PIETRO	1978-04-06	Q3	3200
04541	D3	TESINI MARIO	1978-12-28	Q4	3740
04551	D1	BIANCHI MAURO	1968-07-09	Q1	1180
04717	D2	PIERINI MARIO	1969-06-20	Q4	2520
04794	D2	CARLETTI PAOLO	1981-07-02	Q3	2270
05019	D4	SOLDANI GIULIO	1983-03-27	Q1	1180
05462	D3	LAPINI PAOLO	1977-01-11	Q5	3960
05477	D4	BRESCHI CARLA	1975-05-02	Q4	4200

SELECT id_dip, AVG(stipendio) AS stipendio_medio

FROM Personale

GROUP BY id_dip

id_dip	AVG(stipendio)
D1	1410
D2	2500
D3	3542
D4	3376.6666666666665

2. Trovo il massimo fra i risultati della query precedente utilizzando tale query come subquery. **Chiamando T il risultato della query precedente**:

SELECT MAX(stipendio_medio)

FROM

(SELECT AVG(stipendio) AS stipendio_medio

FROM personale

GROUP BY id_dip) as T

3542

ATTENZIONE, serve l'ALIAS T altrimenti non funziona

(**occhio**, la seguente query funziona ma restituisce risultato sbagliato

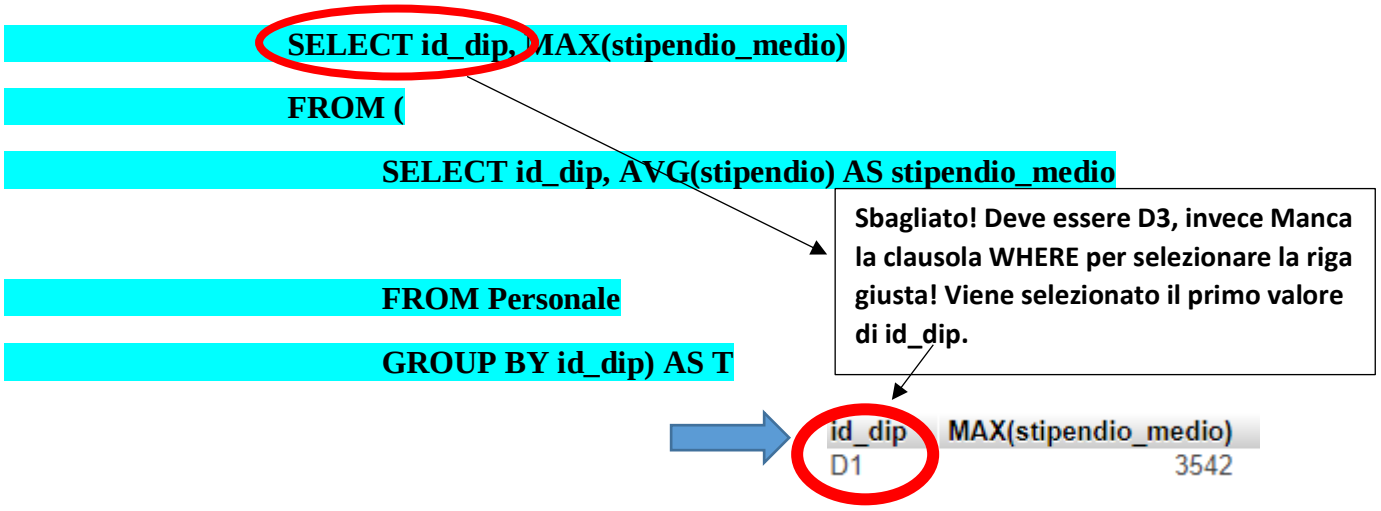
SELECT id_dip, MAX(stipendio_medio)

FROM (

SELECT id_dip, AVG(stipendio) AS stipendio_medio

FROM Personale

GROUP BY id_dip) AS T



3. Seleziono, dalla query del punto precedente, solamente i gruppi che hanno come stipendio medio il valore calcolato al punto 2

   SELECT id_dip, AVG(stipendio) as stipendio_medio

   FROM personale

   GROUP BY id_dip

   HAVING stipendio_medio=(

   SELECT max(stipendio_medio)

   FROM (SELECT id_dip, AVG(stipendio) as stipendio_medio

   FROM personale

   GROUP BY id_dip) as T)

Esempio 3:

determinare il codice dipartimento con il maggior numero di dipendenti.

1. Determino il numero di dipendenti di ciascun dipartimento dalla tabella Personale raggruppando per dipartimento e on la funzione di aggregazione COUNT

   (SELECT COUNT(matricola) AS n_dipendenti
   FROM Personale
   GROUP BY id_dip) AS T

id_dip	COUNT(*)
D1	2
D2	4
D3	5
D4	3

2. Per determinare il valore massimo del risultato della query precedente (chiamata T)

   SELECT MAX(n_dipendenti)

   FROM T  **5**

3. Riscrivo la query del punto 1 aggiungendo nel SELECT l'id_dip e aggiungendo la clausola HAVING in cui selezioni solo il gruppo che come valore di n_dipendenti il valore calcolato al punto 2

   SELECT id_dip, COUNT(matricola) AS n_dipendenti
   FROM personale
   GROUP BY (id_dip)
   HAVING n_dipendenti = SELECT MAX(n_dipendenti) FROM T

4. La query finale si ottiene sostituendo a T, nella query precedente, la subquery del punto 1

   SELECT id_dip, count(matricola) as n_dipendenti
   FROM personale
   GROUP BY id_dip
   HAVING n_dipendenti=(
   SELECT MAX(n_dipendenti)
   FROM (SELECT COUNT(matricola) AS n_dipendenti
   FROM Personale
   GROUP BY id_dip) AS T)

id_dip	n_dipendenti
D3	5

Esercizi Riassuntivi. Database Azienda

1. **Selezionare lo stipendio minimo**
2. **Selezionare il dipendente (matricola e nominativo) con lo stipendio minimo**
3. **Mostrare, per ogni dipartimento, il dipendente con lo stipendio massimo (si mostri l' id_dip, il nominativo e lo stipendio)**
   Ragionamento: determino con una query la tabella T1 con gli stipendi massimi e chiamo tale risultato tabella T, poi faccio il Join fra T e personale sulla base dello stipendio e del' id_dip
4. **Per ogni dipartimento, mostrare la quantità complessivamente utilizzata di ciascun componente per la realizzazione dei propri prodotti. Si mostrino il nome del dipartimento, il nome del componente e la quantita utilizzata.**
5. **Mostrare, per ogni dipartimento, qual è il prezzo medio dei prodotti realizzati, (mostrare id_dip e prezzo medio)**
6. **Mostrare, per ogni dipartimento, quale è il componente più utilizzato (e quanto ne viene utilizzato) per la realizzazione di ciascuno dei propri prodotti (si mostrino, il nome del dipartimento, il nome del componente e la quantità utilizzata)**

Strategia:

- **Faccio il primo raggruppamento (tabella T3)**

T3: Per ogni dipartimento la quantità di ogni componente utilizzata

+ Opzioni

id_dip	id_comp	quantita
D1	C001	3
D1	C002	2
D1	C003	3
D1	C004	3
D2	C002	2
D4	C008	2
D4	C009	2
D4	C010	1
D4	C015	1
D5	C012	7
D5	C013	3

- **Su T3 faccio il secondo raggruppamento (chiamo la tabella T2) raggruppando solo per id_dip e ottenendo il massimo valore di quantità per ciascun dipartimento.**

T2: per ogni dipartimento la quantità **massima** di componente utilizzata

id_dip	quantita_massima
D1	3
D2	2
D4	2
D5	7

- **Dalla query di partenza (T3) faccio il JOIN con (T2) (oppure uso IN) ottenendo per ogni Id_dip solo il componente con quantità massima.**

Join (verde)
T2.quantita_massima=T3.quantita AND
T2.id_dip=T3.id_dip

id_dip	id_comp	quantita_massima
D1	C004	3
D1	C003	3
D1	C001	3
D2	C002	2
D4	C009	2
D4	C008	2
D5	C012	7

- **Per avere il nome del dipartimento e il nome del componente, aggiungo al JOIN precedente le tabelle dipartimenti e componenti**

Join (nero) con le tabelle dipartimenti e componenti per ottenere i nomi dei dipartimenti e i nomi dei componenti

nome	nome_componente	quantita_massima
primo dipartimento	Componente04	3
primo dipartimento	Componente03	3
primo dipartimento	Componente01	3
secondo dipartimento	Componente02	2
quarto dipartimento	Componente09	2
quarto dipartimento	Componente08	2
quinto dipartimento	Componente12	7

Soluzione:

```
SELECT dipartimenti.nome,componenti.nome_componente,
T4.quantita_massima
FROM
(SELECT T2.id_dip, T3.id_comp, T2.quantita_massima
FROM
(SELECT T.id_dip, max(T.quantita) as quantita_massima
FROM
(SELECT prodotti.id_dip, composizione.id_comp,
SUM(composizione.unita_comp) as quantita
FROM prodotti,composizione
WHERE prodotti.id_prod=composizione.id_prod
GROUP BY prodotti.id_dip,composizione.id_comp) as T
GROUP BY T.id_dip) as T2, (SELECT prodotti.id_dip,
composizione.id_comp, SUM(composizione.unita_comp) as quantita
FROM prodotti,composizione
WHERE prodotti.id_prod=composizione.id_prod
GROUP BY prodotti.id_dip,composizione.id_comp) as T3
WHERE T2.quantita_massima=T3.quantita and
T2.id_dip=T3.id_dip) as T4, dipartimenti,componenti
WHERE T4.id_dip=dipartimenti.id_dip AND
T4.id_comp=componenti.id_comp;
```

7. **Mostrare quale è il dipartimento con l'età media dei dipendenti più bassa, e mostrare il dipartimento (id_dip) e il valore di tale età media.**

Passo 1 calcolo età media raggruppata per dipartimento

```
SELECT id_dip, avg(Timestampdiff(Year,data_nascita,CURRENT_DATE)) as eta_media
FROM personale
GROUP BY id_dip
```

id_dip	eta_media
NULL	45.0000
D1	46.0000
D2	46.0000
D3	43.6000
D4	47.0000

Passo 2 determino il valore minimo per questa tabella

```
SELECT min(eta_media) as eta_media_minima
FROM
(SELECT id_dip, avg(Timestampdiff(Year,data_nascita,CURRENT_DATE)) as eta_media
FROM personale
GROUP BY id_dip) as T
```

eta_media_minima
43.6000

Passo 3 "Replicando" la query di partenza con HAVING, individuo il dipartimento che ha eta_media corrispondente ad eta_media_minima

```
SELECT id_dip, avg(Timestampdiff(Year,data_nascita,CURRENT_DATE)) as eta_media
FROM personale
GROUP BY id_dip
HAVING eta_media=(SELECT min(eta_media) as eta_media_minima
FROM
(SELECT id_dip, avg(Timestampdiff(Year,data_nascita,CURRENT_DATE)) as eta_media
FROM personale
GROUP BY id_dip) as T)
```

id_dip	eta_media
D3	43.6000

Esercizi Riassuntivi. Database Cinema

1. **Selezionare l'anno del (dei) film più vecchio(i)**
2. **Selezionare il titolo e l'anno del film più vecchio**
3. **Mostrare, per ogni film, il numero di attori partecipanti**
4. **Mostrare quanti film si sono aggiudicati ciascun premio (quanti film hanno vinto la palma d'oro, quanti film, l'orso d'oro ecc..)**
   Strategia: faccio un join fra le tabelle film, ha_vinto_premi e raggruppo per descrizione_premi contando i film
5. **Mostrare, per ogni genere cinematografico, quanti premi sono stati vinti**
   Strategia: faccio un JOIN fra genere, film, ha_vinto, premi e raggruppo per genere contando le righe (count(*)).
6. **Mostrare quale è il film (o i film) che ha (hanno) vinto più premi. Mostrare il titolo e il numero di premi vinti**
7. **Mostrare quali sono gli attori che recitano nel film che ha vinto più premi**

   Strategia: partendo dalla query precedente faccio un join con le tabelle attore, recita_in e faccio una proiezione sul titolo e il nominativo.
8. **Mostrare i film che non hanno vinto alcun premio**

CLAUSOLA LIMIT

La clausola LIMIT limita il numero di righe che saranno restituite.

Per mostrare il dipendente più vecchio si può scrivere:

```
SELECT *
FROM Personale
ORDER BY data_nascita DESC
LIMIT 1
```

Attenzione però: se ci sono due dipendenti nati lo stesso giorno ne mostra solo uno

La seguente strategia con LIMIT può essere utilizzata per la seguente "query del massimo " e simili tenendo sempre conto però che in caso di più risultati viene mostrata una sola tupla, quindi il risultato in realtà non è corretto

Esempio:

Mostrare il codice prodotto con il maggior numero di componenti

Soluzione corretta:

```
SELECT id_prod, count(*) as numero_componenti
FROM composizione
group by id_prod
HAVING numero_componenti=
    (SELECT max(numero_componenti)
    FROM
        (SELECT id_prod, count(*) as numero_componenti
        FROM composizione
        group by id_prod) as T)
```

Riscrivi la stessa query

Soluzione con LIMIT, funziona solamente se c'è UN SOLO prodotto con numero massimo di componenti:

```
SELECT id_prod, count(*) as numero_componenti
FROM composizione
group by id_prod
order by numero_componenti DESC
LIMIT 1
```

3. OPERATORI DI UNIONE, INTERSEZIONE E DIFFERENZA

UNIONE

Si ricorda che l'unione fra due tabelle restituisce una tabella contenente sia le righe della prima che della seconda tabella. Le righe uguali vengono eliminate automaticamente. Le tabelle devono avere gli stessi campi.

In SQL l'unione è ottenuta attraverso l'operatore **UNION** da utilizzarsi con la seguente sintassi:

SELECT <lista campi1>
FROM <lista tabelle1>
[WHERE <condizione1>]

UNION

SELECT <lista campi2>
FROM <lista tabelle2>
[WHERE <condizione2>]

dove <lista campi1> e <lista campi2> deve essere uguale

Se si vuole che le righe uguali non vengano eliminate si usa il comando **UNION ALL**.

Esercizio: nel database Cinema mostrare l'elenco con i nominativi degli attori e dei musicisti che hanno lavorato in ogni film (mostrare anche il titolo del film)

```
SELECT nominativo, titolo
FROM attori,recita_in,film
WHERE attori.id_attore=recita_in.id_attore AND recita_in.id_film=film.id_film
UNION
SELECT nominativo, titolo
FROM musicisti,colonne_sonore,film
WHERE musicisti.id_musicista=colonne_sonore.id_musicista AND
colonne_sonore.id_film=film.id_film
ORDER BY titolo;
```

INTERSEZIONE

L'intersezione fra due tabelle restituisce una tabella contenente le righe che sono presenti sia nella prima che nella seconda tabella. I campi delle due tabelle devono essere uguali. In SQL l'operatore che svolge questa azione è l'operatore **INTERSECT, CHE PERO' NON E' SUPPORTATO DA DBMS MySQL/MariaDB.**

L'operazione insiemistica di intersezione si può generalmente ottenere mediante selezione su un SELF JOIN

Esempio:

si vuole ottenere il codice dei prodotti che utilizzano sia i componenti C001 che C003.

La soluzione si potrebbe ottenere come intersezione della tabella dei prodotti che usano C001con la tabella dei prodotti che usano C003.

SELECT id_prod

FROM Composizione

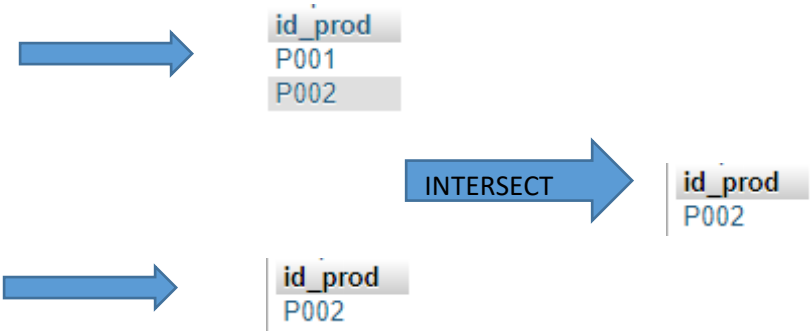
WHERE id_com='C001'

INTERSECT

SELECT id_prod

FROM Composizione

WHERE id_com='C003'



Prima soluzione con MariaDB: con una query interna seleziono gli id_prod che contengono il componente C002, con una query esterna seleziono gli id_prod che contengono il componente C001 E fanno parte dei risultati della query 1:

```
SELECT id_prod
FROM composizione
WHERE id_comp="C001" and id_prod IN (SELECT id_prod
FROM composizione
WHERE id_comp="C003")
```

Un’altra soluzione con MariaDB è quella di realizzare un SELF JOIN sul campo id_prod con la tabella Composizione in modo da avere tutte le combinazioni di componenti di ciascun prodotto a 2 a 2. Sulla tabella ottenuta si selezionano le righe in cui il primo componente è C001 e il secondo è C003.

SELECT id_prod

FROM

(composizione AS x INNER JOIN composizione AS y

USING (id_prod))

WHERE x.id_comp='C001' AND y.id_comp='C003';

id_prod	id_comp	unita_comp	id_comp	unita_comp
P001	C001	2	C001	2
P001	C001	2	C002	2
P001	C002	2	C001	2
P001	C002	2	C002	2
P002	C001	1	C001	1
P002	C001	1	C003	3
P002	C001	1	C004	3
P002	C003	3	C001	1
P002	C003	3	C003	3
P002	C003	3	C004	3
P002	C004	3	C001	1
P002	C004	3	C003	3
P002	C004	3	C004	3
P003	C002	2	C002	2
P004	C008	2	C008	2
P004	C008	2	C009	2
P004	C008	2	C010	1
P004	C008	2	C015	1
P004	C009	2	C008	2
P004	C009	2	C009	2
P004	C009	2	C010	1
P004	C009	2	C015	1
P004	C010	1	C008	2
P004	C010	1	C009	2
P004	C010	1	C010	1

id_prod
P002

Esercizio. Nel database cinema selezionare gli attori che recitano in tutti i seguenti film:“Frida”,“la casa” , “The heateful eight”

DIFFERENZA

La DIFFERENZA fra due tabelle restituisce una tabella contenente le righe che sono presenti nella prima ma non nella seconda tabella. I campi delle due tabelle devono essere uguali. In SQL l'operatore che svolge questa azione è l'operatore **EXCEPT, CHE PERO' NON E' SUPPORTATO DA DBMS MySQL/MariaDB.**

Per ottenere i codici dei prodotti che usano il componente C001 ma non il componente C003 con EXCEPT si potrebbe fare la differenza fra l'elenco dei prodotti che usano C001 e l'elenco dei prodotti che usano C003:

SELECT id_prod

FROM Composizione

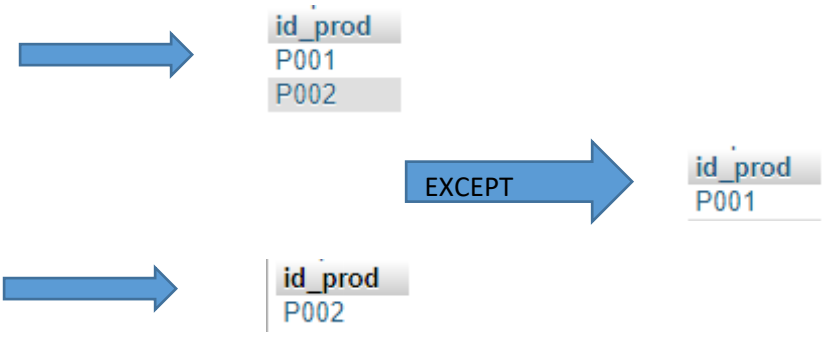
WHERE id_com='C001'

EXCEPT

SELECT id_prod

FROM Composizione

WHERE id_com='C003'



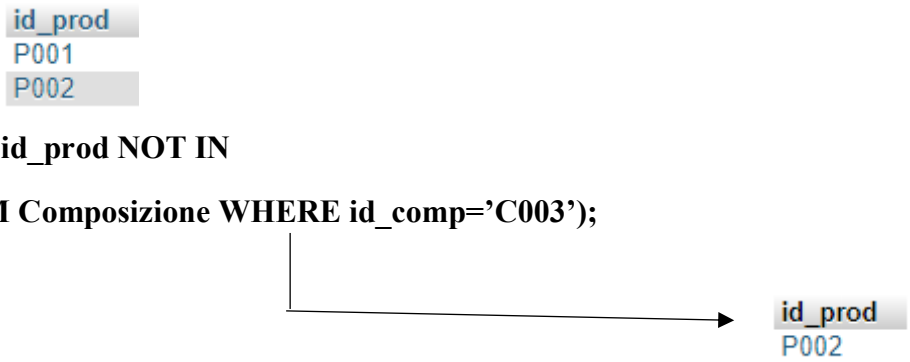
EXCEPT può essere facilmente sostituito con un NOT IN. Quindi l'elenco dei prodotti che usano C001 ma non usano C003 si può ottenere con la seguente clausola WHERE in cui vengono selezionati i prodotti che usano C001 e che NON SONO nella tabella dei prodotti che usano C003

SELECT id_prod

FROM Composizione

WHERE id_comp='C001' AND id_prod NOT IN

(SELECT id_prod FROM Composizione WHERE id_comp='C003');



Risultato

id_prod
P001

Esercizio: Nel database cinema selezionare gli attori che recitano nel film "la casa" ma non recitano nel film "frida"