14 / 11 / 2024

CLAUSOLA HAVING / ATZE.4.3.4

Esempio di HAVING senza GROUP BY

Indicate la capienza media dei tipi di aerei, soltanto se questa capienza media è < 100.

```
/* HAVING SENZA GROUP BY. */
SELECT AVG(Capienza)
FROM TIPI_AEREI
HAVING AVG(Capienza) < 100;
```

SOLUZIONE DEGLI ESERCIZI PER CASA

SOLU.Aeroporti.H

Per ogni città italiana, indicate il numero di voli internazionali in partenza.

L'elenco deve contenere soltanto le città in cui questo numero è > 0.

SOLU.Aeroporti.I

Elencate le città francesi da cui partono almeno venti voli alla settimana diretti in ITA.

SOLU.Aeroporti.J

Elencate le città da cui partono voli diretti a Milano, ordinate alfabeticamente.

OPERAZIONI INSIEMISTICHE (CONTINUAZIONE) / ATZE.4.3.5

Alcune versioni di Sql non possiedono gli operatori `INTERSECT` e `EXCEPT`. Quindi, quando questi operatori non esistono, dovremo progettare le operazioni suddette mediante un select annidato (che vedremo tra poco). Poiché l'uso di questi operatori aumenta la leggibilità del software, io e voi li useremo opportunamente.

Naturalmente, useremo anche l'operatore `UNION`, che esiste in qualsiasi versione di Sql e che non è sostituibile da altre operazioni.

Operazioni insiemistici / Esercizi risolti

NB Inserite un commento per ogni sottoproblema nel quale avete scomposto il problema iniziale.

1. Considerate il db di riferimento.

Elencate le matricole dei capi i cui impiegati guadagnano tutti più di 40, senza usare operatori di aggregazione.

Es: L'input seguente:

MatriImpie	Stipendio	MatriImpieCapo
10	50	1
20	30	1
30	60	2
44	70	2

genera l'output seguente:

MatriImpieCapo
2

Soluzione

La consegna equivale a:

```
Matricole dei capi
EXCEPT
Matricole dei capi di (almeno) un impiegato che guadagna al massimo 40
```

Codifico ognuno dei due sottoproblemi precedenti:

```
SELECT MatriImpieCapo /* Matricole dei capi */
FROM IMPIEGATI
  EXCEPT
SELECT MatriImpieCapo
/* Matricole dei capi di (almeno) un impiegato che guadagna al massimo 40 */
FROM IMPIEGATI
WHERE Stipendio <= 40;
```

JOIN COMPLETO E INCOMPLETO

Il **risultato** di un join è detto **completo** se ogni riga di T1 e ogni riga di T2 compaiono nel risultato del join; altrimenti, è detto **incompleto**.

NB Il join completo o quello incompleto non sono operazioni differenti. Invece, sono esiti differenti della medesima operazione di join.

Es Considerate il db di riferimento.

Per ogni dipendente, elencate la sua matricola e la città in cui lavora.

L'output richiesto è qualcosa del tipo:

MatriImpie	Citta
1	PD
2	PD
3	MI
5	PD
7	PD

Notate che i dipendenti non assegnati a un dipartimento non compaiono nel risultato. Quindi, questo join è incompleto.

Come potete individuare facilmente, la soluzione è:

```
SELECT I.MatriImpie, D.Citta
FROM DIPARTIMENTI AS D JOIN IMPIEGATI AS I ON D.NomeDipa = I.NomeDipa;
```

NB: Quando il join è completo, si ha:

Cardinalità del join completo >= max {cardinalità di T1, cardinalità di T2}.

JOIN ESTERNO

Sono date:

~ Due tabelle `T1`, `T2`.

~ Una condizione `C` come nel join interno.

Join esterno sinistro

Il **join esterno sinistro** sugli attributi `X1` di `T1` e gli attributi `X2` di `T2` è l'unione tra

~ il join interno (eseguito mediante la condizione `C`)

e

~ la concatenazione tra

  ~ ogni riga di `T1` che non compare nel suddetto join interno

  e

  ~ i valori NULL negli attributi di `T2`.

Quindi, il join interno destro contiene sempre ogni riga di `T2`; infatti, contiene anche le righe di `T2` che non compaiono nel join interno).

Join esterno destro

Il join esterno destro è analogo a quello destro; l'unica differenza è lo scambio dei ruoli tra le tabelle `T1` e `T2`. La definizione formale è la seguente.

Il **join esterno destro** sugli attributi `X1` di `T1` e gli attributi `X2` di `T2` è l'unione tra

~ il join interno (eseguito mediante la condizione `C`)

e

~ la concatenazione tra

  ~ ogni riga di `T2` che non compare nel suddetto join interno

  e

  ~ i valori NULL negli attributi di `T1`.

Quindi, il join interno destro contiene sempre ogni riga di `T1`; infatti, contiene anche le righe di `T1` che non compaiono nel join interno.

Join esterno completo

Il **join esterno completo** sugli attributi `X1` di `T1` e gli attributi `X2` di `T2` è l'unione tra

~ il join esterno sinistro (eseguito mediante la condizione `C`)

e

~ il join esterno destro (eseguito mediante la condizione `C`).

Quindi, il join esterno completo contiene sempre ogni riga di `T1` e ogni riga di `T2`; infatti, contiene anche le righe di `T1` e quelle di `T2` che non compaiono nel join interno.

Inoltre, non contiene due righe uguali (perché, per default, l'operazione di unione elimina le righe uguali).

Join esterno / Esercizi risolti

1. Considerate il db di riferimento.

Per ogni dipendente, elencate la sua matricola e l'eventuale città in cui lavora.

L'output richiesto è qualcosa del tipo:

MatriImpie	Citta
1	PD
2	PD
3	MI
4	NULL
5	PD
6	NULL
7	PD

Soluzione

Notate la parola "eventuale". Questa parola implica che dobbiamo visualizzare la matricola anche di un dipendente del quale non è nota la città in cui lavora.

Quindi, dobbiamo usare un join esterno:

```
SELECT I.MatriImpie, D.Citta
FROM DIPARTIMENTI AS D RIGHT JOIN IMPIEGATI AS I ON D.NomeDipa = I.NomeDipa
```

2. Considerate il db di riferimento.

Per ogni dipartimento, elencate il suo nome e le matricola degli eventuali impiegati che lavorano nel dipartimento.

L'output richiesto è qualcosa del tipo:

NomeDipa	MatriImpie
CE	NULL
HW	3
SW	1
SW	2
SW	5
SW	7

Soluzione

```
SELECT D.NomeDipa, I.MatriImpie
FROM DIPARTIMENTI AS D LEFT JOIN IMPIEGATI AS I ON D.NomeDipa = I.NomeDipa
ORDER BY D.NomeDipa;
```

2B. Risolvete l'esercizio precedente senza usare il join esterno.

Soluzione

La consegna equivale a elencare:

1: Per ogni dipartimento, il suo nome e le matricole degli impiegati che lavorano nel dipartimento

UNION

2: Ogni dipartimento senza impiegati

La consegna 2, a sua volta, equivale a elencare:

2.1: Ogni dipartimento

EXCEPT

2.2: Ogni dipartimento in cui lavora qualche impiegato

Risolvo i punti 1, 2.1, 2.2 in Sql, e, contemporaneamente, inserisco gli opportuni operatori insiemistici.

```
(SELECT D.NomeDipa, I.MatriImpie
/* 1: Per ogni dipartimento: nome e le matricole degli impiegati che vi lavorano */
FROM DIPARTIMENTI AS D LEFT JOIN IMPIEGATI AS I ON D.NomeDipa = I.NomeDipa
ORDER BY D.NomeDipa
) UNION /* 2: Dipartimenti senza impiegati */
(SELECT NomeDipa, NULL /* 2.1: Dipartimenti */
FROM DIPARTIMENTI
) EXCEPT
(SELECT NomeDipa, NULL /* 2.2: Dipartimenti in cui qualche impiegato lavora */
FROM IMPIEGATI
);
```

SELECT ANNIDATI

Un'interrogazione (o query) contiene:

~ 1 select;

oppure

~ 2 o più select.

Nel caso di due (o più) select, ci sono due situazioni:

~ I due select si "trovano allo stesso livello".

Allora, Sql risolve separatamente ogni select e poi combina i due risultati mediante un operatore insiemistico (`UNION`, `INTERSECT`, `EXCEPT`).

oppure

~ Un select è annidato (cioè, è interno) a un altro.

In questo sottocaso, ci sono due ulteriori situazioni:

~ Il select interno è indipendente da quello esterno.

oppure

~ Il select interno è collegato a quello esterno.

Un select è **annidato** quando è composto da un select esterno e da (almeno) un select interno.

Risultato

Un select annidato confronta (mediante un operatore specificato dal programmatore)
~ alcuni valori individuati dal select interno
con
~ ogni valore elencato del select esterno.

Poiché c'è un confronto, dobbiamo inserirlo in una clausola (del select esterno) che lo consente; ciò si fa, specificatamente, gestendo il confronto:
~ nel `WHERE`, quando dobbiamo confrontare ogni riga (del select esterno);
oppure
~ nell'`HAVING`, quando dobbiamo confrontare ogni gruppo (del select esterno).

Di conseguenza, il select annidato visualizza ogni riga (oppure ogni gruppo) del select esterno che soddisfa il confronto.

Es:

```
SELECT T1.X, COUNT(T1.K)
FROM T1
GROUP BY T1.X
HAVING COUNT(T1.K) < (
    SELECT COUNT(T1.K)
    FROM T1
    WHERE T1.X = 13
);
```

Input

Supponiamo che T1 contenga:

T1

K	X
A	11
C	12
D	13
E	13

Esecuzione della traccia

Il select interno individua:

COUNT(K)
2

Il select esterno "diventa":

```
SELECT T1.X, COUNT(T1.K)
FROM T1
GROUP BY T1.X
HAVING COUNT(T1.K) < 2;
```

Risultato finale:

X	COUNT(K)
11	1
12	1