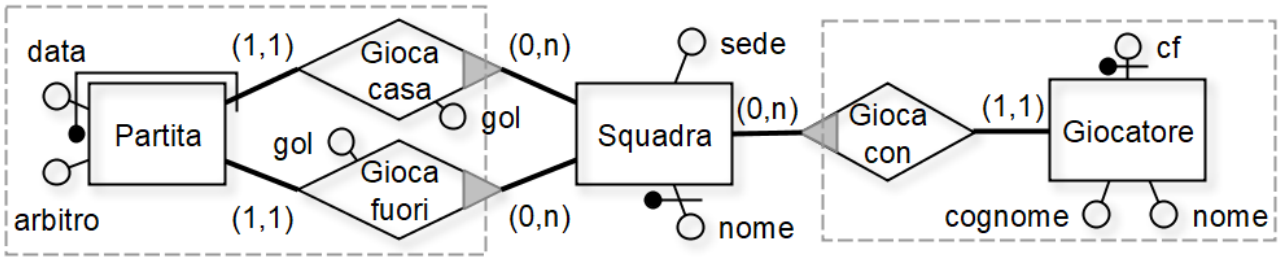
Si consideri il seguente schema, e si formulino in SQL le interrogazioni proposte.



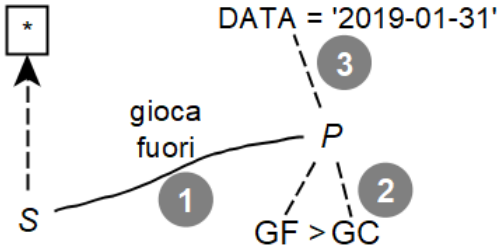

```
GIOCATORI (ID, CF, NOME, COGNOME, ID_SQ^)
SQUADRE   (ID, NOME, SEDE)
PARTITE   (ID, DATA, ARBITRO, ID_SQ_CASA^, GOL_CASA, ID_SQ_FUORI^, GOL_FUORI)
```

a. le squadre che hanno vinto in trasferta il 31 Gennaio 2019
b. le squadre che hanno sconfitto l'Inter sul suo campo (quando l'Inter giocava in casa)
c. l'elenco alfabetico delle squadre con il numero di partite giocate in casa e di partite giocate fuori casa
d. le squadre che non hanno mai segnato alcun gol fuori casa

a. le squadre che hanno vinto in trasferta il 31 Gennaio 2019

Considerata la generica squadra S (di cui si porranno in output tutti i campi), e la generica partita P, si ha:

1. S è la squadra che gioca fuori la partita P
2. P è stata vinta da chi giocava fuori casa
3. P è stata disputata il 31 Gennaio 2019

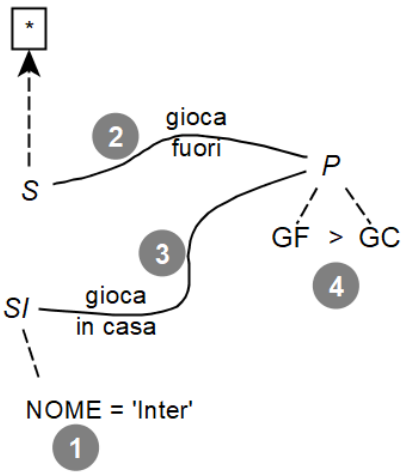



```
SELECT S.*
FROM   squadre S, partite P
WHERE  S.ID = P.ID_SQ_FUORI                -- 1
   AND P.GOL_FUORI > P.GOL_CASA            -- 2
   AND P.DATA='2019-01-31';                -- 3
```

b. le squadre che hanno sconfitto l'Inter sul suo campo (l'Inter giocava in casa)

Si considera la squadra generica S (che si avrà in output), un'altra squadra SI (l'Inter), ed una partita P, si ha:

1. SI ha NOME= 'Inter'
2. S è la squadra che gioca fuori la partita P
3. SI è la squadra che gioca in casa la partita P
4. nella partita P, i gol fatti fuori sono maggiori dei gol in casa




```
SELECT DISTINCT S.*
FROM   squadre S, squadre SI, partite P
WHERE  SI.NOME = 'Inter'                   -- 1
   AND S.ID = P.ID_SQ_FUORI                -- 2
   AND SI.ID = P.ID_SQ_CASA                -- 3
   AND P.GOL_FUORI > P.GOL_CASA;           -- 4
```

c. l'elenco alfabetico delle squadre con il numero di partite giocate in casa e di partite giocate fuori casa

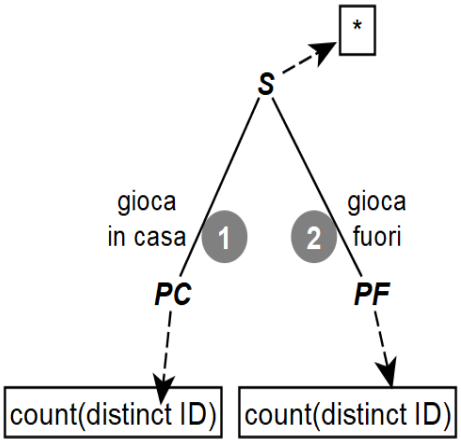
Vediamo due delle possibili formulazioni di questa interrogazione.

Considerata la generica squadra S, si costruiscono tutte le possibili TERNE ⟨squadra S, partita PC, partita PF⟩, dove:

1. S ha giocato in casa PC
2. S ha giocato fuori casa PF

Raggruppando per squadra (in base all'attributo chiave ID):

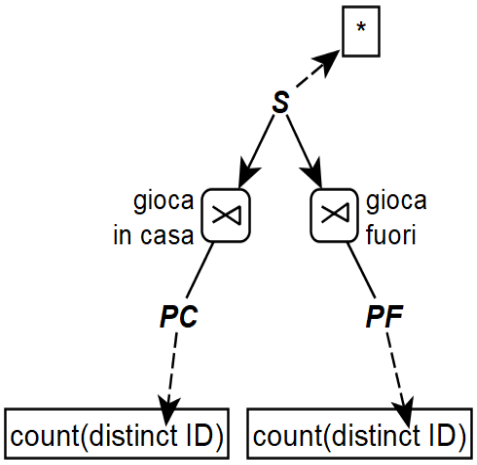
- si CONTANO le partite PC (eliminando i duplicati)
- si CONTANO le partite PF (eliminando i duplicati)




```
SELECT S.*,
       count(distinct PC.ID) as CASA,
       count(distinct PF.ID) as TRASF
FROM   squadre S, partite PC, partite PF
WHERE  S.ID = PC.ID_SQ_CASA                -- 1
   AND S.ID = PF.ID_SQ_FUORI               -- 2
GROUP BY S.ID
ORDER BY S.NOME;
```

L'eliminazione dei duplicati è assolutamente indispensabile.
Questa è forse la formulazione più semplice, ma non funziona SEMPRE bene.
Infatti, vengono escluse dall'output quelle squadre che hanno disputato ZERO partite in casa oppure fuori, non essendo possibile costruire una terna S, PC, PF con le condizioni suddette.
In altri termini, con questa interrogazione non verranno mai forniti risultati aventi ZERO come valore di conteggio.

Per avere un risultato valido nel caso generale, possiamo ricorrere al semi-join, che ammette la generazione di valori nulli.
Nel costruire le terne ⟨squadra S, partita PC, partita PF⟩, partiamo dalle squadre, ed usiamo il LEFT JOIN.
Si confronti questa interrogazione con la precedente, trasformando le condizioni nella `'where'` in condizioni di join.




```
SELECT S.*,
       count(distinct PC.ID) as CASA,
       count(distinct PF.ID) as TRASF
FROM   squadre S
  LEFT JOIN partite PC ON S.ID = PC.ID_SQ_CASA
  LEFT JOIN partite PF ON S.ID = PF.ID_SQ_FUORI
GROUP BY S.ID
ORDER BY S.NOME;
```

Prima di applicare l'operatore di aggregazione, le terne generate dal `LEFT JOIN` includono anche tuple in cui ogni partita di una squadra che ha giocato solo in casa è giustapposta a valori nulli (in corrispondenza degli attributi della partita fuori casa).
A tale proposito, è interessante vedere il risultato intermedio PRIMA di applicare l'operatore aggregativo, nei due casi (prima soluzione e seconda soluzione, con due tuple in più):

```
SELECT *
FROM   squadre S, partite PC, partite PF
```

```
WHERE  S.ID = PC.ID_SQ_CASA
   AND S.ID = PF.ID_SQ_FUORI;
```

```
SELECT *
FROM   squadre S
  LEFT JOIN partite PC ON S.ID = PC.ID_SQ_CASA
  LEFT JOIN partite PF ON S.ID = PF.ID_SQ_FUORI;
```

d. le squadre che non hanno mai segnato alcun gol fuori casa

Vediamo due delle molte possibili formulazioni di questa interrogazione (con negazione).

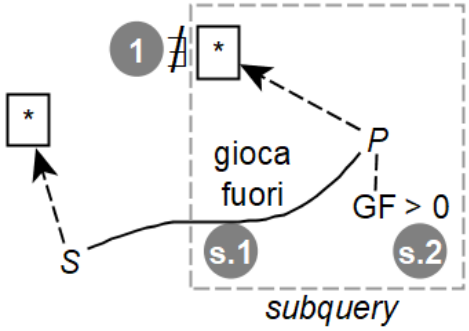
Considerata la generica squadra S:

subquery SQ: considerato l'insieme delle partite P tali che:

s1. S gioca fuori la partita P

s2. la squadra che gioca fuori ha segnato

unica condizione: la subquery SQ è vuota.



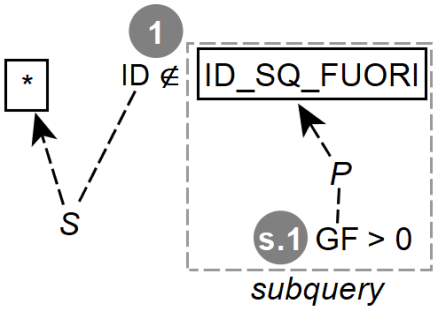

```
SELECT S.*
FROM   squadre S
WHERE  NOT EXISTS                -- 1 (unica condiz)
   (SELECT *
    FROM   partite P
    WHERE  S.ID = P.ID_SQ_FUORI -- s.1
    AND  P.GOL_FUORI > 0);      -- s.2
```

Consideriamo la seguente sottointerrogazione:

subquery SQ: prendiamo l'identificativo di tutte le squadre che hanno giocato fuori casa una partita P tale che:

s1. la squadra che gioca fuori ha segnato.

La query vera e propria consiste nel prendere tutte le squadre S il cui identificativo NON è nell'insieme SQ (unica condizione).




```
SELECT S.*
FROM   squadre S
WHERE  s.ID NOT IN               -- 1 (unica condiz)
   (SELECT P.ID_SQ_FUORI
    FROM   partite P
    WHERE  P.GOL_FUORI > 0);     -- s.1
```

Si osserva che la subquery SQ (che NON È parametrica, ossia può essere eseguita indipendentemente) contiene un elemento per ogni partita in cui la squadra fuori ha segnato, quindi la stessa squadra può comparire più volte. Questi duplicati non alterano in alcun modo la condizione di appartenenza. In altri termini, l'utilizzo della parola chiave "DISTINCT" nella sottointerrogazione è ininfluente sul risultato.