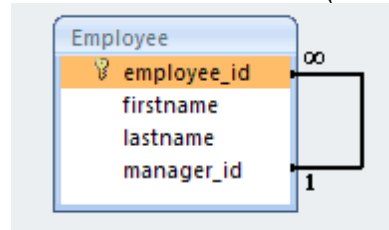


## SELF JOIN (2020)



### TABELLE IN ASSOCIAZIONE CON SE STESSA primo esempi: relazione coniugale (Tabella persone) in ASSOCIAZIONE 1:1

Si voglia rappresentare la situazione coniugale  
Si voglia poi rappresentare con una base di dati  
le coppie marito-moglie .. e si effettui poi alcune query ...(vedi sotto)

*/\* la tabella PERSONE (mariti+mogli) e' in associazione 1:1 con PERSONE (se stessa)*

```
create table persone ( codice integer primary key,  
                      cognome varchar(30) not null,  
                      nome varchar(30) not null,  
                      sesso char(1),  
                      coniuge integer references persone(codice),  
                      /* INTERESSANTE il references alla stessa tabella*/  
                      check (sesso IN ('M','F')),  
                      check (coniuge <> codice) ); /* Banale ma utile */
```

Perchè devo inserire prima i NULL ????? e poi i matrimoni ????

*/\* → perchè se no NON potevo inserire nulla in quanto si VIOLA il references già al 1° inserimento \*/*

```
insert into persone values (10,'Aldova','Nicola', 'M',NULL);  
insert into persone values (20,'Balza', 'Robert', 'M',NULL);  
insert into persone values (30,'Brozzo', 'Sebastiano', 'M',NULL);  
insert into persone values (40,'Bulla','Maria Carla','F',NULL);  
insert into persone values (50,'Catellano', 'David', 'M',NULL);  
insert into persone values (60,'Del Santo', 'Andrew', 'M',NULL);  
insert into persone values (70,'Mordaca', 'Elena', 'F',NULL);  
insert into persone values (80,'Foca', 'Sara', 'F',NULL);  
insert into persone values (90,'Tumi', 'Erica', 'F',NULL);  
insert into persone values (100,'Tumi', 'Elena', 'F',NULL);
```

```
update persone set coniuge=40 where codice=10;  
update persone set coniuge=10 where codice=40;  
/*Aldrova-Bulla */  
update persone set coniuge=70 where codice=20;  
update persone set coniuge=20 where codice=70;  
/*Balza - Mordaca */  
update persone set coniuge=90 where codice=30;  
update persone set coniuge=30 where codice=90;  
/*Brozzo - Tumi Erica */
```

### 1) Coppie, ma quale è il problema ?

```
SELECT T1.cognome, T1.nome, T2.cognome, T2.nome
FROM persone T1, persone T2
WHERE T1.coniuge = T2.codice;
```

Le coppie sono doppie... Si risolve con DISTINCT?

→ NO!

### 2) Coppie MARITO-MOGLIE (esercizio: fare coppie MOGLIE-MARITO)

```
SELECT T1.cognome, T1.nome, T2.cognome, T2.nome
from persone T1, persone T2
WHERE T1.coniuge = T2.codice AND T1.sesso='M';
```

### 3) CELIBI /\* NON FUNZIONA .. Perché ? \*/

```
SELECT cognome, nome
FROM persone
WHERE coniuge=NULL AND sesso='M';
```

### 4) CELIBI FUNZIONA (uso di IS NULL)

```
SELECT cognome, nome
FROM persone
WHERE coniuge IS NULL AND sesso='M';
```

### 5) SPOSATI /\* FUNZIONA!!! \*/

```
SELECT cognome, nome
FROM persone
WHERE coniuge IS NOT NULL AND sesso='M';
```

### 6) POSSIBILI COPPIE (CELIBI <----> NUBILI)

```
SELECT T1.cognome, T1.nome, T2.cognome, T2.nome
FROM persone T1, persone T2
WHERE T1.coniuge IS NULL AND T2.coniuge IS NULL
AND T1.sesso <> T2.sesso AND T1.sesso = 'M';
```

**Oppure ci arrivo con viste... (qui sotto vediamo alcuni esempi di possibili viste)**

```
CREATE view sposati as      /* SPOSATI vista*/  
SELECT codice, cognome, nome, sesso, coniuge  
FROM persone  
WHERE coniuge IS NOT NULL AND sesso='M';
```

**Approfondimento SQL → Utilizzo di IN in select interna**  
**VIEW CELIBI SENZA USO DI IS NULL / IS NOT NULL**  
**(FUNZIONA, INTERESSANTE!!! )**

```
create view celibi as  
SELECT cognome, nome  
FROM persone  
WHERE SESSO = 'M' AND codice NOT IN (select codice FROM SPOSATI);
```

```
CREATE view sposate as      /* SPOSATE vista*/  
SELECT codice, cognome, nome, sesso, coniuge  
FROM persone  
WHERE coniuge IS NOT NULL AND sesso='F';
```

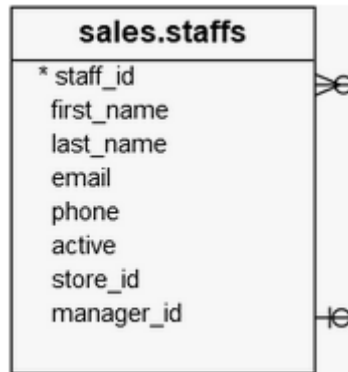
**VISTA NUBILI (COME NON SPOSATE!!!!)**

```
create view nubili as  
SELECT cognome, nome  
FROM persone  
WHERE SESSO = 'F' AND codice NOT in (select codice FROM SPOSATE);
```

**Possibili coppie WOW!!!: PRODOTTO CARTESIANO (CROSS JOIN) FRA VISTE!!!!**

```
SELECT celibi.cognome, celibi.nome, nubili.cognome,nubili.nome  
FROM celibi,nubili  
ORDER BY celibi.cognome,celibi.nome
```

## TABELLE IN ASSOCIAZIONE CON SE STESSA ASSOCIAZIONE 1:N



**Esempio: relazione gruppi/capogruppi → tabella classe (alunni)**

Si immagini di avere una classe divisa in gruppi, ciascuno con il loro capogruppo. Sono richieste, ad esempio, le seguenti query:

- 1) I capogruppo
- 2) I non capogruppo
- 3) Chi non ha capogruppo
- 4) Un gruppo, dato il suo capogruppo
- 5) Un gruppo, dato un componente del gruppo
- 6) Quanti componenti ha ciascun gruppo
- 7) Il gruppo più numeroso

**Procediamo con la creazione della tabella:**

```
CREATE TABLE classe (  
id          INTEGER PRIMARY KEY,  
cognome    VARCHAR(30) NOT NULL,  
nome       VARCHAR(30) NOT NULL,  
capo       integer REFERENCES classe(id) );
```

**popoliamo la tabella:**

```
-- I CAPOGRUPPO:  
  
INSERT INTO classe VALUES (1, 'ALE', 'NIC', NULL);  
INSERT INTO classe VALUES (2, 'ANI', 'LUKE', NULL);  
INSERT INTO classe VALUES (3, 'ARCU', 'GIULIO', NULL);  
INSERT INTO classe VALUES (4, 'CASCIA', 'ANDREA', NULL);  
INSERT INTO classe VALUES (5, 'CURA', 'SIMONE', NULL);
```

-- I componenti dei gruppi con il loro capogruppo:

```
INSERT INTO classe VALUES (6, 'GALVA', 'GIORGIO', 1);           --ALE
INSERT INTO classe VALUES (7, 'GORRE', 'OTTAVIA', 2);          --ANI
INSERT INTO classe VALUES (8, 'KING', 'VANESSA', 3);           --ARCU
INSERT INTO classe VALUES (9, 'LATTE', 'LEONARDO', 4);          --CASCIA
INSERT INTO classe VALUES (10, 'MANZO', 'ALESSANDRO', 5);       --CURA
INSERT INTO classe VALUES (11, 'MORE', 'REBECCA', 1);
INSERT INTO classe VALUES (12, 'NERO', 'GIACOMO', 2);
INSERT INTO classe VALUES (13, 'PETRINI', 'OLLIO', 3);
INSERT INTO classe VALUES (14, 'PULCA', 'MATTEO', 4);
INSERT INTO classe VALUES (15, 'QUIO', 'GIANLUCA', 5);
INSERT INTO classe VALUES (16, 'RATTO', 'ANDREA', 1);
INSERT INTO classe VALUES (17, 'SETTE', 'SIMONE', NULL); -- Non capo ma senza capogruppo
INSERT INTO classe VALUES (18, 'TANNO', 'GIORGIO', NULL); -- Non capo ma senza capogruppo
INSERT INTO classe VALUES (19, 'TEGGO', 'ALEANDRO', 4);
INSERT INTO classe VALUES (20, 'VALE', 'ALBERTO', 5);
INSERT INTO classe VALUES (21, 'ZANNI', 'ANDREA', 1);
```

Ed ora le query:

--1) I capogruppo

```
--drop view capi;
CREATE view capi(codice,nominativo) AS
SELECT id, cognome || ' ' || nome
FROM classe
WHERE id IN (SELECT DISTINCT capo FROM classe);

SELECT * FROM capi
ORDER BY nominativo;
```

--2) I non capogruppo

```
SELECT id,cognome,nome FROM
classe WHERE id NOT IN (SELECT codice FROM capi)
ORDER BY id;
```

--3) Chi non ha capogruppo

```
SELECT id,cognome,nome FROM
classe WHERE capo ISNULL AND id NOT IN (SELECT codice FROM capi)
ORDER BY id;
```

--4) Un gruppo, dato il suo capogruppo

```
SELECT id, cognome,nome
FROM classe
WHERE capo = 1 -- per codice
ORDER BY cognome,nome;
```

--oppure, se viene ricercato per il cognome del capogruppo:

```
SELECT T2.cognome || ' ' || T2.nome AS Capogruppo,  
T1.id, T1.cognome, T1.nome  
FROM classe T1, classe T2  
WHERE T1.capo = T2.id AND T2.cognome = 'ALE' -- per cognome  
ORDER BY Capogruppo, T1.cognome, T1.nome
```

--5) Un gruppo, dato un componente del gruppo

```
SELECT T1.capo, T1.cognome, T1.nome  
FROM classe T1, classe T2  
WHERE T1.capo = T2.capo AND T2.id = 14 -- Pulcini, gruppo Casciana  
ORDER BY T1.cognome, T1.nome -- manca il capogruppo!!!
```

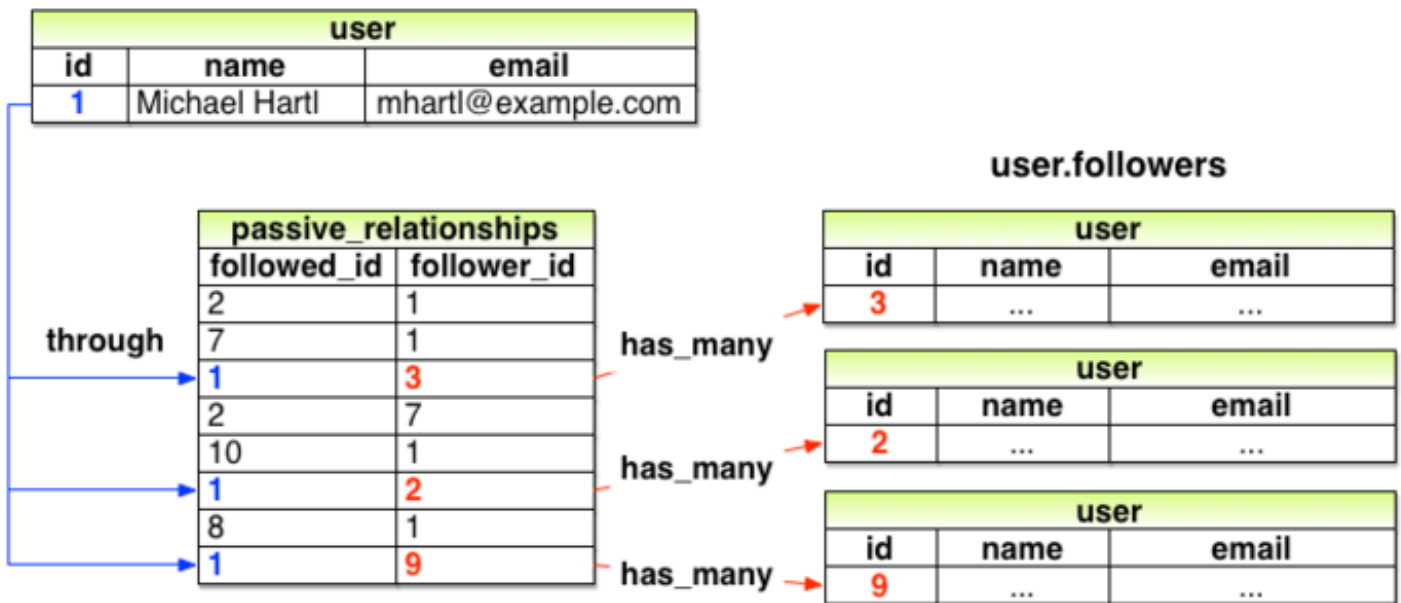
--6) Quanti componenti ha ciascun gruppo

```
--drop view numerocomponenti;  
CREATE view numerocomponenti (numero, capo) AS  
SELECT count(*)+1, capo -- +1, c'e' anche il capo!!!  
FROM classe  
WHERE capo IS NOT NULL  
GROUP BY capo;  
  
SELECT * from numerocomponenti;
```

--7) Il gruppo più numeroso

```
SELECT capo, numero  
FROM numerocomponenti  
WHERE numero = (SELECT MAX(numero) FROM numerocomponenti);
```

## ASSOCIAZIONE CON SE STESSO: di tipo N:N



Si voglia rappresentare la **situazione AMICIZIE** (ad esempio all'interno di una **classe**)

Ogni persona e' in ASSOCIAZIONE di amicizia 1:N con altre persone.

L'ASSOCIAZIONE e' fra le tabelle PERSONA e PERSONA, ed e' quindi N:N

Si voglia poi rappresentare con una base di dati le amicizie in una classe.. e poi alcune query ...

```
/* Creo tabella persone */
```

```
create table persone ( codice integer primary key,
                       cognome varchar(30) not null,
                       nome varchar(30) not null,
                       sesso char(1),
                       ntel varchar(20),
                       check (sesso IN ('M','F')) );
```

```
/* creo poi la tabella amicizie */
```

```
create table amici (
    cod integer references persone(codice),
    amico integer references persone(codice),
    /* Interessante: 2 references simili */
    check (cod <> amico),
    /* uno non puo' essere amico con se se stesso */
    primary key (cod, amico) );
```

## --Seguono Insert:

```
insert into persone values (10,'Aldrova','Nicola','M','0521 243456');
insert into persone values (20,'Balza','Roberto','M','0521 243455');
insert into persone values (30,'Brozza','Sebastiano','M','0521 233453');
insert into persone values (40,'Bulla','Maria Cristina','F','0521 243456');
insert into persone values (50,'Catella','Davide','M','0521 223452');
insert into persone values (60,'Del Santo','Andrew','M','0521 213457');
insert into persone values (70,'Mordaca','Elena','F','0521 263451');
insert into persone values (80,'Foca','Sara','F','0521 223450');
insert into persone values (90,'Tumi','Erica','F','0521 243456');

insert into amici values (10,20);
insert into amici values (10,40);
insert into amici values (10,80); /* amici di Aldrova */

insert into amici values (40,80);
insert into amici values (40,90);
insert into amici values (40,70); /* amici di Bulla */

insert into amici values (50,10);
insert into amici values (50,30);
insert into amici values (50,70); /* amici di Catella */

insert into amici values (30,90); /* amico di Brozza */
```

## --1) Tutti gli amici con relativo numero di telefono (JOIN SU SE STESSA)

```
SELECT T1.cognome, T1.ntel, T2.cognome, T2.ntel
FROM persone T1, persone T2, amici
WHERE T1.codice = amici.cod AND amici.amico = t2.codice;
```

## --2) Gli amici di “Catella”

```
SELECT codice, cognome, nome, ntel
FROM persone, amici
WHERE amici.cod=50 and persone.codice=amici.amico
```

## --3) Tutti le persone che hanno scelto “MORDACA” come loro amica --(Mordaca ha codice 70)

```
SELECT codice, cognome, nome
FROM amici, persone
WHERE codice = amici.cod AND amici.amico = 70
```



--4) Amici degli amici di “Catella” → DIFFICILE senza vista

--(meglio passare per la vista "amici di catella", codice =50)

```
create view amici_catella(codice, cognome, nome, ntel) AS
SELECT codice, cognome, nome, ntel
FROM persone, amici
WHERE amici.cod=50 and persone.codice=amici.amico;

/* .. e poi JOIN amici di catella <---> e loro amici: WOW! */

SELECT T1.cognome, T1.ntel, T2.cognome, T2.ntel
from amici_catellani T1, persone T2, amici
WHERE T1.codice = amici.cod AND amici.amico = t2.codice;

--Oppure (tabulato semplificato):

SELECT persone.cognome, persone.nome, persone.ntel
FROM amici_catella, amici, persone
WHERE amici.cod=amici_catella.codice AND persone.codice=amico

--Infine una soluzione senza vista: (complicata!)

SELECT T2.nome, T2.cognome
FROM (SELECT T2.codice as cod
FROM persone T1, persone T2, amici
WHERE T1.codice = cod AND T2.codice=amico AND T1.codice=50) as
C, amici, persone T2
WHERE C.cod=amici.cod AND T2.codice=amico;
```