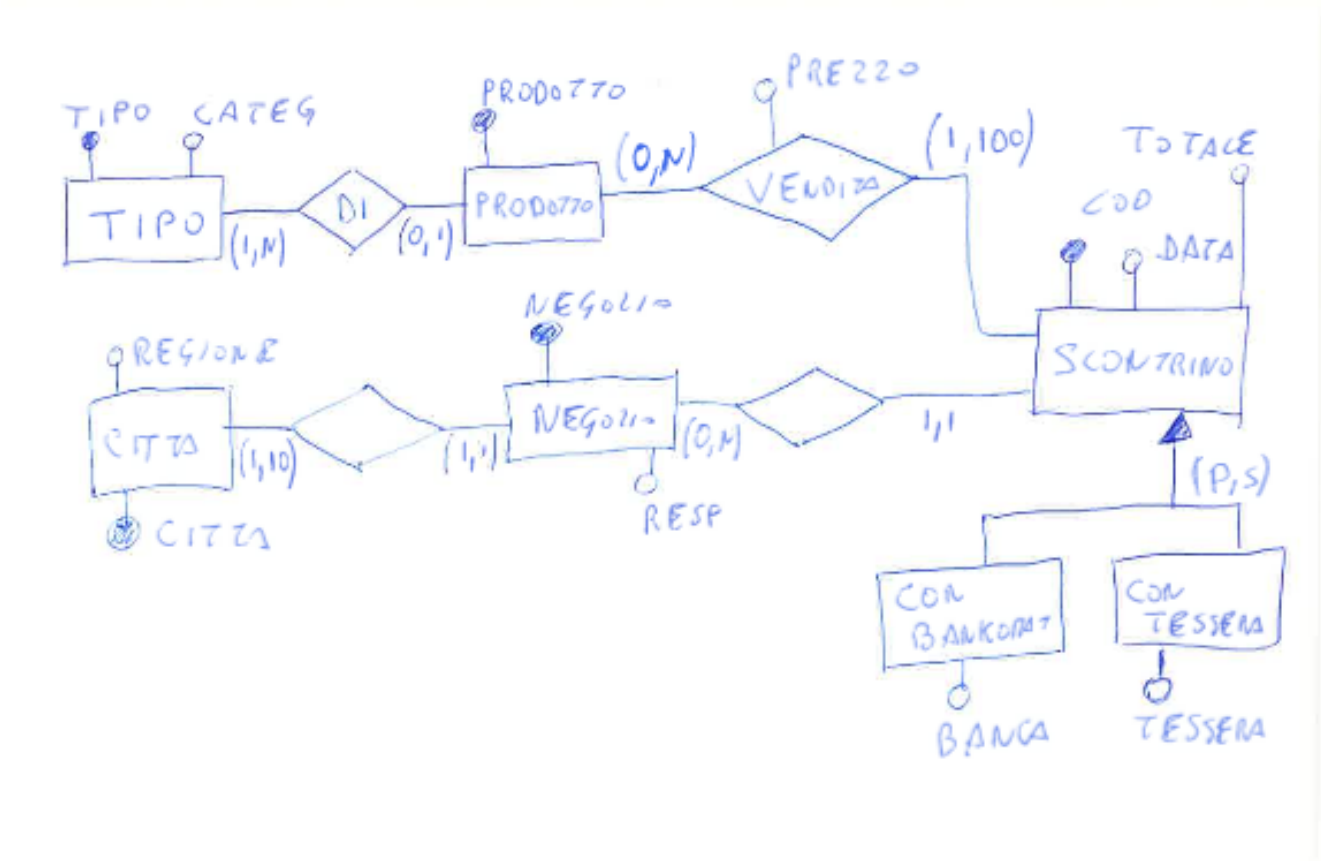
[redacted]

SCHEMA ER



Riferimenti per l'ER:

1) Libro di testo di Sistemi Informativi, Capitolo 1
2) Prove scritte con Soluzione (http://dbgroup.unimo.it/SIRE/ScrittiAnnoAccademico20112012.pdf) Capitolo 2
3) Dispense "Introduzione modello ER" (http://dbgroup.unimo.it/SIRE/2-ER-SI.pdf)

SCHEMA RELAZIONALE

```
TCITTA(CITTA,REGIONE)

TNEGOZIO(NEGOZIO,RESPONS,CITTA)
      FK CITTA REFERENCES TCITTA NOT NULL

TSCONTRINO(COD,DATA,TOTALE,NEGOZIO)
      FK NEGOZIO REFERENCES TNEGOZIO NOT NULL

TCONBANKOMAT(COD,BANCA)
      FK COD REFERENCES TSCONTRINO

TCONTESSERA(COD,TESSERA)
      FK COD REFERENCES TSCONTRINO

TTIPO(TIPO,CATEG)

TPRODOTTO(PRODOTTO,TIPO)
      FK TIPO REFERENCES TTIPO

VENDITA(COD,PRODOTTO,PREZZO)
      FK COD REFERENCES TSCONTRINO
      FK PRODOTTO  REFERENCES TPRODOTTO
```

Una versione semplificata è quella in cui la Foreign Key viene indicata direttamente accanto al nome dell'attributo

```
TCITTA(CITTA,REGIONE)

TNEGOZIO(NEGOZIO,RESPONS,CITTA:TCITTA)

TSCONTRINO(COD,DATA,TOTALE,NEGOZIO:TNEGOZIO)

TCONBANKOMAT(COD:TSCONTRINO,BANCA)

TCONTESSERA(COD:TSCONTRINO,TESSERA)

TTIPO(TIPO,CATEG)

TPRODOTTO(PRODOTTO,TIPO:TTIPO)

VENDITA(COD:TSCONTRINO,PRODOTTO:TPRODOTTO,PREZZO)
```

Riferimenti per il modello relazionale e la progettazione logico-relazionale:

1) Libro di testo di Sistemi Informativi, Capitolo 2, sezione 2.1
2) Libro di testo di Sistemi Informativi, Capitolo 3
3) Prove scritte con Soluzione (http://dbgroup.unimo.it/SIRE/ScrittiAnnoAccademico20112012.pdf) Capitolo 3

SCHEMA RELAZIONALE IN SQL

Per implementare il DB in un DBMS (SQL SERVER nel nostro caso), lo schema relazionale è scritto in linguaggio SQL

```
CREATE TABLE TCITTA(
   CITTA CHAR(12),
   REGIONE CHAR(12),
   PRIMARY KEY(CITTA))

CREATE TABLE TNEGOZIO(
   NEGOZIO CHAR(12),
   RESPONS CHAR(12),
   CITTA CHAR(12) NOT NULL, -- LA FOREIGN KEY CITTA NON PUO' ESSERE NULL
   PRIMARY KEY(NEGOZIO),
   FOREIGN KEY(CITTA) REFERENCES TCITTA)

CREATE TABLE TSCONTRINO(
   COD INT,
   DATA DATETIME,
   TOTALE FLOAT,
   NEGOZIO CHAR(12) NOT NULL, -- LA FOREIGN KEY NEGOZIO NON PUO' ESSERE NULL
   PRIMARY KEY(COD),
   FOREIGN KEY(NEGOZIO) REFERENCES TNEGOZIO )

CREATE TABLE TCONBANKOMAT(
  COD INT,  BANCA CHAR(12),
  PRIMARY KEY(COD),
  FOREIGN KEY(COD) REFERENCES TSCONTRINO)

CREATE TABLE TCONTESSERA(
  COD INT,  TESSERA CHAR(12),
  PRIMARY KEY(COD),
  FOREIGN KEY(COD) REFERENCES TSCONTRINO)


CREATE TABLE TTIPO(
   TIPO CHAR(12), CATEG CHAR(12),
   PRIMARY KEY(TIPO))

CREATE TABLE TPRODOTTO(
   PRODOTTO CHAR(12),
   TIPO CHAR(12), -- LA FOREIGN KEY TIPO PUO' ESSERE NULL, CI SONO PRODOTTI SENZA TIPO
   PRIMARY KEY(PRODOTTO),
   FOREIGN KEY(TIPO) REFERENCES TTIPO )

CREATE TABLE VENDITA(
  COD INT
  PRODOTTO CHAR(12),
  PREZZO INT,
  PRIMARY KEY(COD,PRODOTTO),
  FOREIGN KEY(COD) REFERENCES TSCONTRINO,
  FOREIGN KEY(PRODOTTO ) REFERENCES TPRODOTTO )
```

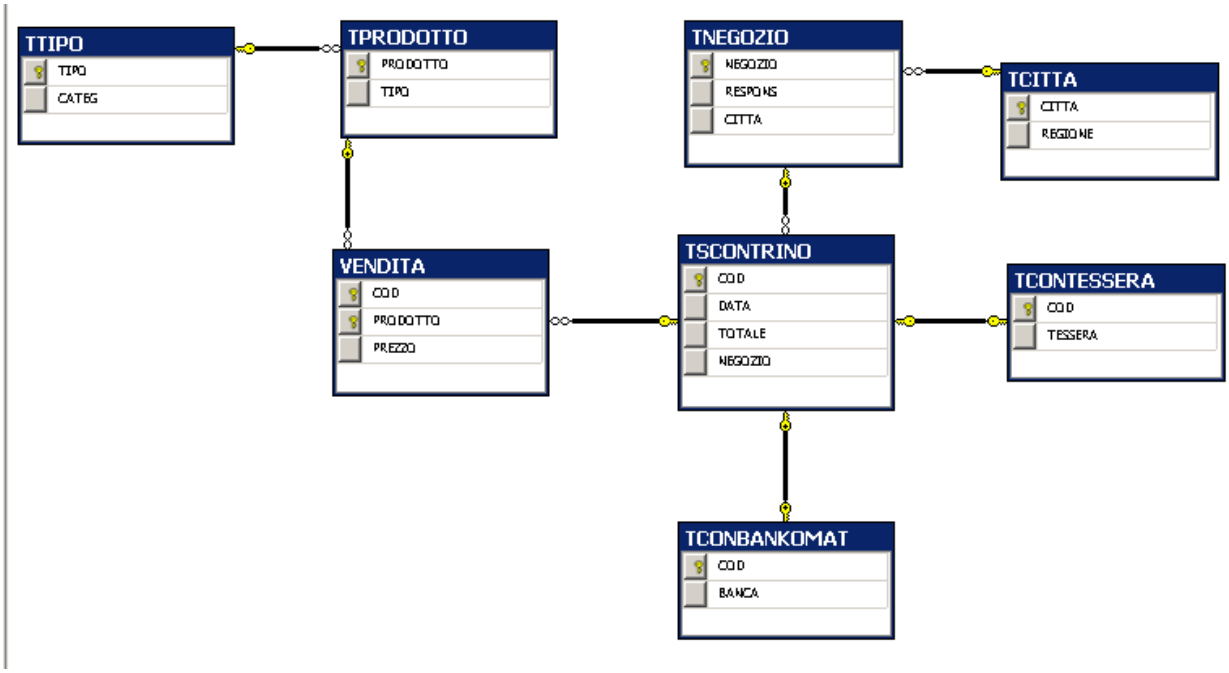
Nota: questa è la “versione base” in cui vengono riportati solo gli “elementi minimi indispensabili” definiti nello schema relazionale di partenza. Altri elementi possono essere aggiunti :
ad esempio ad una Foreign Key può essere dato un nome esplicito .

Riferimenti per l’SQL:

1) Libro di testo di Sistemi Informativi, Capitolo 4, sezioni 4.1, 4.2, 4.5
2) Prove scritte con Soluzione (http://dbgroup.unimo.it/SIRE/ScrittiAnnoAccademico20112012.pdf) Capitolo 4
3) Dispense “Richiami sul linguaggio SQL”

DIAGRAMMA RELAZIONALE IN SQL SERVER

E’ una rappresentazione grafica dello schema relazionale



Tale rappresentazione è automaticamente costruita da SQL SERVER e riporta tutti gli elementi dello schema relazionale dichiarato in SQL (in particolare chiave e foreign key).

D’altra parte il diagramma è anche uno **strumento di progetto**, ovvero può essere utilizzato per definire graficamente chiavi e foreign key. Ad esempio , potrei definire semplicemente

```
CREATE TABLE TCITTA(
   CITTA CHAR(12),
   REGIONE CHAR(12))

CREATE TABLE TNEGOZIO(
   NEGOZIO CHAR(12),
   RESPONS CHAR(12),
   CITTA CHAR(12) NOT NULL)
```

E poi *completarlo* con il diagramma relazionale.

Modello ER , del **modello relazionale** e della **progettazione logica** nel corso di Sistemi Informativi Avanzati.

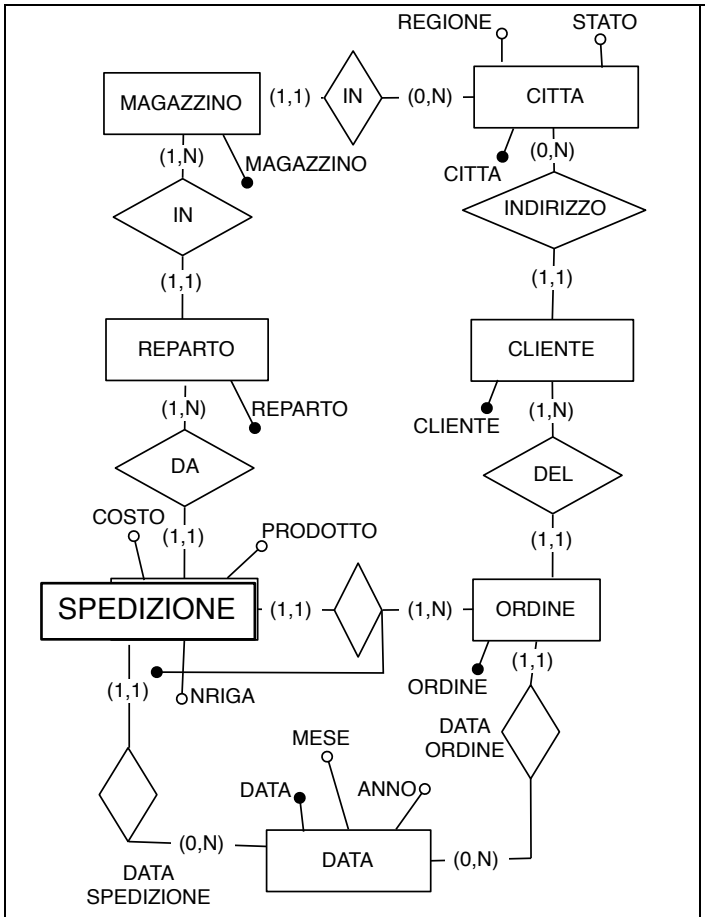
Il DB Operazionale (DBO) è il punto di partenza della progettazione del DW, quindi

1) occorre saper leggere ed interpretare lo schema ER del DBO (se disponibile)
2) se lo schema ER non è disponibile occorre ricavarlo con un processo inverso alla progettazione logica (Reverse Engineering)
3) occorre saper leggere ed interpretare lo schema relazionale del DBO (se disponibile)
4) se lo schema relazionale *è privo/incompleto* di chiavi e foreign key, occorre individuarle tramite un processo di Data Profiling (formulazione di particolari interrogazioni sul DBO)
5) occorre saper formulare semplici modifiche dei dati per correggere errori presenti nel DBO

Il DW verrà implementato come un DB relazionale

1) verrà studiata la teoria di progettazione logica per ottenere lo schema relazionale del DB (star schema e/o snowflake schema)
2) bisogna definire le corrispondenti tabelle in SQL per costruire tale schema relazionale
3) occorre saper formulare interrogazioni sul DBO (soprattutto join e raggruppamento) allo scopo di alimentare il DW

ESEMPIO : Schema ER delle spedizioni e relativo schema relazionale




```
CITTA(CITTA,REGIONE,STATO)

DATA(DATA,MESE,ANNO)

CLIENTE(CLIENTE,CITTA:CITTA)

ORDINE(ORDINE,CLIENTE:CLIENTE,DATA:DATA)


MAGAZZINO(MAGAZZINO,CITTA:CITTA)

REPARTO(REPARTO,MAGAZZINO:MAGAZZINO)

SPEDIZIONE(NRIGA,ORDINE:ORDINE,
           PRODOTTO,
           REPARTO:REPARTO
           DATASPEDIZIONE:DATA,
           COSTO)
```

LINGUAGGIO SQL – INTERROGAZIONI DI JOIN

Schema semplificato per le query di join

```
CREATE TABLE SC(
   COD INT, DATA DATE, TOTALE FLOAT,
   PRIMARY KEY(COD))

CREATE TABLE CB(
  COD INT,  BANCA CHAR(10),
  PRIMARY KEY(COD),
  FOREIGN KEY(COD) REFERENCES SC)

CREATE TABLE CT(
  COD INT,  TESSERA CHAR(10),
  PRIMARY KEY(COD),
  FOREIGN KEY(COD) REFERENCES SC)


INSERT INTO SC VALUES(1,'09-26-2013',31)
INSERT INTO SC VALUES(2,'09-27-2013',20)
INSERT INTO SC VALUES(3,'09-26-2013',31)
INSERT INTO SC VALUES(4,'09-30-2013',null)
INSERT INTO SC VALUES(5,'09-30-2013',10)


INSERT INTO CB VALUES(1,'B1')
INSERT INTO CB VALUES(2,'B10')

INSERT INTO CT VALUES(1,'TA')
INSERT INTO CT VALUES(3,'TA')
INSERT INTO CT VALUES(5,'TB')

-- A) SCONTRINI CON BANKOMAT (RIPORTA ANCHE LA BANCA)
SELECT SC.*, BANCA
     FROM SC
     JOIN CB ON (SC.COD=CB.COD)

-- JOIN e' sinonimo di INNER JOIN (JOIN INTERNO)

SELECT SC.*, BANCA
     FROM SC
     INNER JOIN CB ON (SC.COD=CB.COD)


-- B) SCONTRINI CON TESSERA (RIPORTA ANCHE LA TESSERA)
SELECT SC.*,  TESSERA
FROM SC
     JOIN CT ON (SC.COD=CT.COD)

-- C) SCONTRINI CON BANKOMAT (RIPORTA ANCHE LA BANCA) E CON TESSERA (RIPORTA ANCHE LA
TESSERA)
SELECT SC.*, BANCA, TESSERA
FROM SC
      JOIN CB ON (SC.COD=CB.COD)
     JOIN CT ON (SC.COD=CT.COD)


-- D) SCONTRINI SENZA TESSERA
            -- E' UNA "DIFFERENZA" TRA TUTTI GLI SCONTRINI E QUELLI IN TESSERA
            ==> LEFT JOIN CON WHERE .... IS NULL
SELECT SC.*
FROM SC
```

```
     LEFT JOIN CT ON (SC.COD=CT.COD)
WHERE CT.COD IS NULL


-- IL LEFT JOIN APPARTIENE ALLA CATEGORIA DEI "JOIN ESTERNI" QUINDI LEFT JOIN E'
SINONIMO DI LEFT OUTER JOIN

SELECT SC.*
FROM SC
     LEFT OUTER JOIN CT ON (SC.COD=CT.COD)
WHERE CT.COD IS NULL



-- E) SCONTRINI SENZA BANKOMAT
SELECT SC.*
FROM SC
     LEFT JOIN CB ON (SC.COD=CB.COD)
WHERE CB.COD IS NULL




--- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA

SELECT SC.*, BANCA
FROM SC
      LEFT JOIN CB ON (SC.COD=CB.COD)


--- TUTTI GLI SCONTRINI ; SE CON TESSERA ANCHE LA TESSERA

SELECT SC.*,  TESSERA
FROM SC
      LEFT JOIN CT ON (SC.COD=CT.COD)

--- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA, SE CON TESSERA ANCHE LA
TESSERA

SELECT SC.*, BANCA, TESSERA
FROM SC
      LEFT JOIN CB ON (SC.COD=CB.COD)
     LEFT JOIN CT ON (SC.COD=CT.COD)



--- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA, SE CON TESSERA ANCHE LA
TESSERA
--- PER SEMPLIFICARE SI PUO' FARE IN DUE PASSI

--- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA
CREATE VIEW SCB AS
SELECT SC.*, BANCA
FROM SC
      LEFT JOIN CB ON (SC.COD=CB.COD)

--- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA, SE CON TESSERA ANCHE LA
TESSERA

SELECT SCB.*,  TESSERA
FROM SCB
      LEFT JOIN CT ON (SCB.COD=CT.COD)



-- PER CODIFICARE I VALORI NULLI SI PUO' USARE COALESCE
```

```
SELECT SC.*,
COALESCE(BANCA,'NOBANKOMAT') AS BANCA,
COALESCE(TESSERA,'NOTESSERA') AS TESSERA
FROM SC
      LEFT JOIN CB ON (SC.COD=CB.COD)
     LEFT JOIN CT ON (SC.COD=CT.COD)


-- ============================================================

-- NOTA
-- C) SCONTRINI CON BANKOMAT (RIPORTA ANCHE LA BANCA) E CON TESSERA (RIPORTA ANCHE LA
TESSERA)
SELECT SC.*, BANCA, TESSERA
FROM SC
      JOIN CB ON (SC.COD=CB.COD)
     JOIN CT ON (SC.COD=CT.COD)

-- IL JOIN E' UN OPERATORE BINARIO VALUTATO DA SINISTRA A DESTRA
-- QUINDI NELLA PRECEDENTE QUERY SI EFFETTUA PRIMA IL JOIN TRA SC E CB E QUINDI IL JOIN
CON CT
-- OVVERO EQUIVALE A SCRIVERE

SELECT SC.*, BANCA, TESSERA
FROM (SC
      JOIN CB ON (SC.COD=CB.COD))
     JOIN CT ON (SC.COD=CT.COD)

-- D'ALTRA PARTE IL JOIN E' ASSOCIATIVO
-- QUINDI E' EQUIVALENTE FARE PRIMA IL JOIN TRA CB E CT E QUINDI IL JOIN CON SC
-- OVVERO SCRIVERE

SELECT SC.*, BANCA, TESSERA
FROM SC
      JOIN (CB JOIN CT ON (CB.COD=CT.COD)) ON (SC.COD=CB.COD)


-- CONSIDERIAMO ORA IL LEFT JOIN
-- TUTTI GLI SCONTRINI ; SE CON BANKOMAT ANCHE LA BANCA, SE CON TESSERA ANCHE LA
TESSERA

SELECT SC.*, BANCA, TESSERA
FROM SC
      LEFT JOIN CB ON (SC.COD=CB.COD)
     LEFT JOIN CT ON (SC.COD=CT.COD)

-- ANCHE IL LEFT JOIN E' UN OPERATORE BINARIO VALUTATO DA SINISTRA A DESTRA
-- QUINDI NELLA PRECEDENTE QUERY SI EFFETTUA PRIMA IL LEFT JOIN TRA SC E CB E QUINDI IL
LEFT JOIN CON CT
-- OVVERO EQUIVALE A SCRIVERE

SELECT SC.*, BANCA, TESSERA
FROM (SC
      LEFT JOIN CB ON (SC.COD=CB.COD))
      LEFT JOIN CT ON (SC.COD=CT.COD)

-- D'ALTRA PARTE IL LEFT JOIN -- A DIFFERENZA DEL JOIN  -- NON E' ASSOCIATIVO
-- QUINDI NON E' EQUIVALENTE FARE PRIMA IL JOIN TRA CB E CT E QUINDI IL JOIN CON SC
-- VERIFICHIAMOLO

SELECT SC.*, BANCA, TESSERA
FROM SC
      LEFT JOIN (CB LEFT JOIN CT ON (CB.COD=CT.COD)) ON (SC.COD=CB.COD)

-- ============================================================
```

```
-- ================================================================


-- F) SCONTRINI CON BANKOMAT OPPURE CON TESSERA
-- è UNA "UNIONE" TRA IL RISULTATO DELLA QUERY A E QUELLO DELLA QUERY B

   -- A) SCONTRINI CON BANKOMAT (RIPORTA ANCHE LA BANCA)
   -- B) SCONTRINI CON TESSERA (RIPORTA ANCHE LA TESSERA)

-- ALLORA SI DEFINISCONO DUE VISTE CORRISPONDENTI A TALI INTERROGAZIONI

CREATE VIEW A AS
SELECT SC.*, BANCA
     FROM SC
     JOIN CB ON (SC.COD=CB.COD)

CREATE VIEW B AS
SELECT SC.*,  TESSERA
FROM SC
     JOIN CT ON (SC.COD=CT.COD)

-- QUINDI PER FARE L'UNIONE SI UTILIZZA IL FULL JOIN TRA A E B


SELECT COALESCE(A.COD,B.COD) AS COD,
COALESCE(A.TOTALE,B.TOTALE) AS TOTALE,
COALESCE(A.DATA,B.DATA) AS TOTALE,
TESSERA,BANCA
FROM A
     FULL JOIN B ON (A.COD=B.COD)

-- IL FULL JOIN APPARTIENE ALLA CATEGORIA DEI "JOIN ESTERNI" QUINDI FULL JOIN E'
SINONIMO DI FULL OUTER JOIN

SELECT COALESCE(A.COD,B.COD) AS COD,
COALESCE(A.TOTALE,B.TOTALE) AS TOTALE,
COALESCE(A.DATA,B.DATA) AS TOTALE,
TESSERA,BANCA
FROM A
     FULL OUTER JOIN B ON (A.COD=B.COD)
```