Query SQL - Combinare più tabelle

Informatica@SEFA 2018/2019 - Lezione 23

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Mercoledì, 12 Dicembre 2018

Prodotto cartesiano

Definizione

In matematica il prodotto cartesiano di A e B è

$$A \times B = \{(a, b) \ : \ \text{for } a \in A \text{ and } b \in B\}$$

Il prodotto cartesiano di due **tabelle** T_1 e T_2 è la tabella ottenuta concatenando ogni riga di T_1 con ogni riga di T_2.

Il numero di righe di $T_1 \times T_2$ è quindi $|T_1| \cdot |T_2|$.

Esempio di di prodotto cartesiano

```
select * from T1,T2;                                                    1
```

A	B	C	D
1	2	a	b
1	2	c	d
1	2	e	f
3	4	a	b
3	4	c	d
3	4	e	f

Table: T1

A	B
1	2
3	4

Table: T2

C	D
a	b
c	d
e	f

Con selezione di colonne

```
select B,A,C from T1,T2;                                    1
```

B	A	C
2	1	a
2	1	c
2	1	e
4	3	a
4	3	c
4	3	e

Table: T1

A	B
1	2
3	4

Table: T2

C	D
a	b
c	d
e	f

Elementi ripetuti

```
select B,A,D from T1,T2;                    1
```

Table: T1

A	B
1	1
2	2

B	A	D
1	1	a
1	1	b
1	1	b
2	2	a
2	2	b
2	2	b

Table: T2

C	D
a	a
b	b
c	b

```
select distinct B,A,D from T1,T2;           1
```

B	A	D
1	1	a
1	1	b
2	2	a
2	2	b

Più di due tabelle

```
select * from T1,T2,T3;                                    1
```

A	B	C	D	E	F
1	1	2017-12-12	Roma	a	a
1	1	2017-12-12	Roma	b	b
1	1	2017-12-26	Praga	a	a
1	1	2017-12-26	Praga	b	b
2	2	2017-12-12	Roma	a	a
2	2	2017-12-12	Roma	b	b
2	2	2017-12-26	Praga	a	a
2	2	2017-12-26	Praga	b	b

Table: T1

A	B
1	1
2	2

Table: T2

C	D
2017-12-12	Roma
2017-12-26	Praga

Table: T3

E	F
a	a
b	b

Sintassi del prodotto cartesiano

```
select <EspressioniColonne>                                    1
       from Tabella1,Tabella2, ..., TabellaN;                  2
```

Selezione di righe

A che serve il prodotto cartesiano? Visto in questo modo sembra un'operazione abbastanza inutile e costosa (produce tabelle molto grandi).

- si usa una clausola `WHERE` per selezionare le righe che hanno dei dati in corrispondenza.

Corrispondenze senza senso

```
select * from Impiegati,Progetti;
```

Impiegato	Progetto	Codice	Nome	Budget
Rossi	A	A	Venere	20000
Rossi	A	B	Marte	10000
Neri	A	A	Venere	20000
Neri	A	B	Marte	10000
Neri	B	A	Venere	20000
Neri	B	B	Marte	10000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000

Corrispondenze con una logica

```
select * from Impiegati,Progetti                    1
  where Progetto=Codice;                            2
```

Impiegato	Progetto	Codice	Nome	Budget
Rossi	A	A	Venere	20000
Neri	A	A	Venere	20000
Neri	B	B	Marte	10000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000

Pulizia delle colonne

```
select Impiegato, Nome as Progetto, Budget
  from Impiegati,Progetti
  where Progetto=Codice;
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000

Riferimenti a tabelle esterne

L'uso che abbiamo fatto del prodotto cartesiano è tipico

- `Impiegati` ha riferimenti ai progetti
- `Progetto` ha le informazioni di ogni progretto

Abbiamo usato il prodotto cartesiano + selezione di righe, per sostituire i riferimenti ai progetti con le loro informazioni esplicite.

Discuteremo questi **riferimenti a tabelle esterne** successivamente.

Ambiguità nei nomi di attributi (I)

Nella base di dati del registro automobilistico:

- La tabella `Veicoli` ha un attributo `Cod_Combustibile`.
- La tabella `Combustibile` ha un attributo `Cod_Combustibile`.

Possiamo incrociare i dati per vedere quali veicoli usano quale tipo di carburante, ma ogni riferimento a `Cod_Combustibile` è ambiguo.

Ambiguità nei nomi di attributi (II)

```
select Targa,Descrizione_Combustibile as Combustibile          1
  from Veicoli, Combustibili as C                              2
  where Veicoli.Cod_Combustibile = C.Cod_Combustibile;         3
```

Targa	Combustibile
A123456X	Benzina
B256787Y	Gasolio
C76589AG	Benzina
C78905GT	Benzina
C845905Z	GPL
CFT340VB	Gasolio
D239765W	Benzina
DD4567XX	Benzina
DGH789JC	Gasolio
DH79567H	Metano
ERG567NM	Metano
F96154NH	GPL
FGH673FV	Benzina
XCH56GJK	Benzina
XF5789CY	Benzina

Vincoli `foreign key`

Riferimenti a tabelle esterne

Una tabella può avere delle colonne **foreign key**

- riferimenti a colonne di una tabella **esterna**
- sono “puntatori” a righe della tabella esterna

```
create table Impiegati (                                    1
     Impiegato nvarchar(20),                                2
     Progetto nchar(1) references Progetti(Codice)          3
);                                                          4
```

La colonna `Progetto` nella tabella `Impiegato` contiene i riferimenti alle righe della tabella `Progetti` che hanno il corrispondente valore nella colonna codice.

Vincolo di integrità referenziale

```
create table Impiegati (                                          1
     Impiegato nvarchar(20),                                      2
     Progetto nchar(1) references Progetti(Codice)                3
);                                                                4
```

- La colonna riferita deve essere `unique`
  E.g. `Codice` in `Progetto` deve essere `unique`
- Il valore nella colonna può essere
  - NULL
  - Un valore **esistente** nella colonna riferita.

  E.g. `("Bianchi",NULL)` può essere inserito
  E.g. `("Bianchi",'C')` non può essere inserito

Esempio

```
PRAGMA foreign_keys = ON;                                    1
select * from Progetti;                                      2
insert into Impiegati values ('Bianchi','C');                3
insert into Impiegati values ('Bianchi',NULL);               4
select * from Progetti;                                      5
```

```
Codice      Nome        Budget
----------  ----------  ----------
A           Venere      20000
B           Marte       10000
Error: FOREIGN KEY constraint failed
Impiegato   Progetto
----------  ----------
Rossi       A
Neri        A
Neri        B
Bianchi     NULL
```

Una osservazione su SQLite

Per compatibilità con le versioni precedenti, SQLite non rispetta i vincoli **foreign key** a meno che questa caratteristica non sia attivata con il comando

```
PRAGMA foreign_keys = ON;
```

Foreign key multicolonna

Le righe di una tabella esterna potrebbero essere identificabili solo utilizzando colonne multiple. È possibile definire un vincolo di integrità referenziale adeguato.

```
create table Studenti (                                          1
      matricola       nvarchar(20),                              2
      id_università   integer;,                                  3
      [...omissis...]                                            4
                                                                 5
      primary key(matricola,id_università)                       6
);                                                               7
create table BorsaDiStudio (                                     8
      cod_pratica integer primary key,                           9
      studente    nvarchar(20),                                  10
      università  integer,                                       11
      [...omissis..]                                             12
                                                                 13
      foreign key(studente,università)                           14
          references Studenti(matricola,id_università)           15
);                                                               16
```

Mantenimento dell’integrità referenziale

I vincoli di **integrità referenziale** impediscono di inserire dei riferimenti a entità che non esistono nella tabella esterna (e.g. nessun impiegato può far parte di un progetto che non esiste).

Eliminazione/Modifica del dato riferito.

E.g. Se il progetto “Venere” viene eliminato/modificato?

Comandi **opzionali** a **foreign key** per scegliere se

- il riferimento viene messo a NULL o ad un default
- l’intera riga referente viene eliminata/aggiornata.
- la modifica viene vietata

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000

Eliminazione/Modifica del dato riferito (2)

```
CREATE TABLE child_table                                                1
(                                                                       2
  column1 datatype1,                                                    3
  column2 datatype2,                                                    4
  ...                                                                   5
                                                                        6
  FOREIGN KEY (child_col1, ..., child_col_n)                            7
  REFERENCES parent_table (parent_col1, ..., parent_col_n)              8
  -- opzionale (uno a scelta)                                           9
  ON DELETE NO ACTION                                                   10
  ON DELETE CASCADE                                                     11
  ON DELETE SET NULL                                                    12
  ON DELETE SET DEFAULT                                                 13
  -- opzionale (uno a scelta)                                           14
  ON UPDATE NO ACTION                                                   15
  ON UPDATE CASCADE                                                     16
  ON UPDATE SET NULL                                                    17
  ON UPDATE SET DEFAULT                                                 18
);                                                                      19
```

Operatori di Join

Torniamo al prodotto cartesiano

```
select Impiegato, Nome as Progetto, Budget                    1
  from Impiegati,Progetti                                     2
  where Progetto=Codice;                                      3
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B
Bianchi	NULL

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000
C	Giove	30000

Operatore Join

L'operazione che abbiamo visto è tipica, Abbiamo risolto i riferimenti da una tabella all'altra, incrociando i dati con

- prodotto cartesiano;
- selezione di righe.

Esiste una sintassi alternativa:

```
select Impiegato, Nome as Progetto, Budget                1
  from Impiegati join Progetti                            2
  on Progetto=Codice;                                     3
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000

Nella sintassi `JOIN` si può usare `WHERE`

```
select Targa,Cilindrata,                                          1
       Descrizione_Combustibile as Combustibile                   2
from Veicoli join Combustibili                                    3
on Veicoli.Cod_Combustibile = Combustibili.Cod_Combustibile       4
where Cilindrata > 1500;                                          5
```

Targa	Cilindrata	Combustibile
A123456X	1796	Benzina
C78905GT	1998	Benzina
DD4567XX	1581	Benzina
DGH789JC	1590	Gasolio
DH79567H	1589	Metano
ERG567NM	1598	Metano
F96154NH	1781	GPL
XCH56GJK	1918	Benzina
XF5789CY	1587	Benzina

Sintassi alternativa: USING

```
select Targa,Cilindrata,                                    1
       Descrizione_Combustibile as Combustibile             2
from Veicoli join Combustibili using(Cod_Combustibile)      3
where Cilindrata > 1500;                                    4
```

Targa	Cilindrata	Combustibile
A123456X	1796	Benzina
C78905GT	1998	Benzina
DD4567XX	1581	Benzina
DGH789JC	1590	Gasolio
DH79567H	1589	Metano
ERG567NM	1598	Metano
F96154NH	1781	GPL
XCH56GJK	1918	Benzina
XF5789CY	1587	Benzina

Sintassi alternativa: NATURAL JOIN

```
select Targa,Cilindrata,                                    1
       Descrizione_Combustibile as Combustibile             2
from Veicoli natural join Combustibili                      3
where Cilindrata > 1500;                                    4
```

Targa	Cilindrata	Combustibile
A123456X	1796	Benzina
C78905GT	1998	Benzina
DD4567XX	1581	Benzina
DGH789JC	1590	Gasolio
DH79567H	1589	Metano
ERG567NM	1598	Metano
F96154NH	1781	GPL
XCH56GJK	1918	Benzina
XF5789CY	1587	Benzina

Join vs Prodotto cartesiano

- la condizione di join
- la selezione successiva delle righe

```
select * from A join B                    1
on <CondizioneJoin>                       2
where <Condizione>;                       3
```

vs

```
select * from A, B                        1
where <CondizioneJoin> and <Condizione>;  2
```

La prima sintassi è più leggibile. Si potrebbe pensare che specificare il join esplicitamente permetta di eseguire la query più velocemente. Questo potrebbe essere vero in **alcune** implementazioni.

Join tra più tabelle

```
select Targa, Nome_Modello as Modello,                    1
       Nome_Fabbrica as Fabbrica                          2
from Veicoli as V join Modelli as M join Fabbriche as F   3
on V.Cod_Modello = M.Cod_Modello and                      4
   F.Cod_Fabbrica = M.Cod_Fabbrica;                       5
```

Targa	Modello	Fabbrica
A123456X	Mondeo	Ford
B256787Y	Panda	Fiat
C76589AG	Panda	Fiat
C78905GT	Coupè	Fiat
C845905Z	Ducato	Fiat
CFT340VB	Clio	Renault
D239765W	Vespa	Piaggio
DD4567XX	Brava	Fiat
DGH789JC	Civic	Honda
DH79567H	Megane	Renault
ERG567NM	Laguna	Renault
F96154NH	Golf	Volkswagen
FGH673FV	Seicento	Fiat
XCH56GJK	V-10	Volvo
XF5789CY	Corolla	Toyota

Join tra più tabelle (natural join)

```
select Targa, Nome_Modello as Modello,                    1
       Nome_Fabbrica as Fabbrica                          2
from Veicoli natural join Modelli natural join Fabbriche;  3
```

Targa	Modello	Fabbrica
A123456X	Mondeo	Ford
B256787Y	Panda	Fiat
C76589AG	Panda	Fiat
C78905GT	Coupè	Fiat
C845905Z	Ducato	Fiat
CFT340VB	Clio	Renault
D239765W	Vespa	Piaggio
DD4567XX	Brava	Fiat
DGH789JC	Civic	Honda
DH79567H	Megane	Renault
ERG567NM	Laguna	Renault
F96154NH	Golf	Volkswagen
FGH673FV	Seicento	Fiat
XCH56GJK	V-10	Volvo
XF5789CY	Corolla	Toyota

Self-Join

Cerchiamo le coppie di macchine con lo stesso modello.

```
select V1.Targa as 'Targa 1',V2.Targa as 'Targa 2'          1
from Veicoli as V1 join Veicoli as V2                       2
on V1.Cod_Modello = V2.Cod_Modello                          3
where V1.Targa < V2.Targa;                                  4
```

Targa 1	Targa 2
B256787Y	C76589AG

Un esempio più articolato

Scriviamo una query SQL che produce le coppie di targhe di veicoli che provengono dalla stessa fabbrica. Vediamo prima il risultato.

Targa 1	Targa 2	Fabbrica
B256787Y	C76589AG	Fiat
B256787Y	DD4567XX	Fiat
B256787Y	C845905Z	Fiat
B256787Y	C78905GT	Fiat
B256787Y	FGH673FV	Fiat
C76589AG	DD4567XX	Fiat
C76589AG	C845905Z	Fiat
C76589AG	C78905GT	Fiat
C76589AG	FGH673FV	Fiat
C78905GT	DD4567XX	Fiat
C78905GT	C845905Z	Fiat
C78905GT	FGH673FV	Fiat
C845905Z	DD4567XX	Fiat
C845905Z	FGH673FV	Fiat
DD4567XX	FGH673FV	Fiat
CFT340VB	DH79567H	Renault
CFT340VB	ERG567NM	Renault
DH79567H	ERG567NM	Renault

La query

È un join di 5 tabelle: due copie di `Veicoli`, due di `Modelli` e una di `Fabbriche`.

```
select V1.Targa as 'Targa 1',V2.Targa as 'Targa 2',          1
       Nome_Fabbrica as Fabbrica                             2
from      Veicoli as V1 join Veicoli as V2                   3
     join Modelli as M1 join Modelli as M2                   4
     join Fabbriche                                          5
on     V1.Cod_Modello = M1.Cod_Modello                       6
   and V2.Cod_Modello = M2.Cod_Modello                       7
   and M1.Cod_Fabbrica = M2.Cod_Fabbrica                     8
   and M1.Cod_Fabbrica = Fabbriche.Cod_Fabbrica              9
where V1.Targa < V2.Targa                                    10
order by Fabbrica;                                           11
```

La query (variazioni)

La query può essere semplificata (anche se così è meno chiara da leggere), usanod i `natural join` e le parentesi tra i join.

- notate la clausola using tra parentesi
- al suo posto avrei potuto usare il natural join

```
select V1.Targa as 'Targa 1',                                    1
       V2.Targa as 'Targa 2',                                    2
       Nome_Fabbrica as 'Fabbrica'                               3
from  (Veicoli as V1 natural join Modelli natural join           4
    Fabbriche)
      join                                                       5
      (Veicoli as V2 join Modelli using(Cod_Modello))            6
using(Cod_Fabbrica)                                              7
where V1.Targa < V2.Targa                                        8
order by Fabbrica;                                               9
```

Varianti del Join

Righe scartate

Il join visto finora ”dimentica” la riga di una tabella se non è in corrispondenza con una riga dell’altra.

```
select Impiegato, Nome as Progetto, Budget          1
  from Impiegati join Progetti                      2
  on Progetto=Codice;                               3
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B
Bianchi	NULL

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000
C	Giove	30000

Join esterni

Vediamo **outer join** (join esterno)

- completo
- sinistro
- destro

che gestiscono diversamente quelle righe.

Join esterno completo

Warning: non è supportato da SQLite.

```
select Impiegato, Nome as Progetto, Budget
  from Impiegati full outer join Progetti
  on Progetto=Codice;
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000
Bianchi	NULL	NULL
NULL	Giove	30000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B
Bianchi	NULL

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000
C	Giove	30000

Join esterno sinistro

```
select Impiegato, Nome as Progetto, Budget                    1
  from Impiegati left join Progetti -- 'left outer join'      2
  on Progetto=Codice;                                         3
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000
Bianchi	NULL	NULL

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B
Bianchi	NULL

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000
C	Giove	30000

Join esterno destro

Warning: non è supportato da SQLite.

```
select Impiegato, Nome as Progetto, Budget                    1
  from Impiegati right join Progetti -- 'right outer join'    2
  on Progetto=Codice;                                         3
```

Impiegato	Progetto	Budget
Rossi	Venere	20000
Neri	Venere	20000
Neri	Marte	10000
NULL	Giove	30000

Table: `Impiegati`

Impiegato	Progetto
Rossi	A
Neri	A
Neri	B
Bianchi	NULL

Table: `Progetti`

Codice	Nome	Budget
A	Venere	20000
B	Marte	10000
C	Giove	30000

Specificare il tipo di join

- `inner join` (`inner` è opzionale)
- `full outer join` (`full` oppure `outer` sono opzionali)
- `left outer join` (`outer` è opzionale)
- `right outer join` (`outer` è opzionale)

Query insiemistiche

Query insiemistiche

Esiste la possibilità di effettuare operazioni di

- unione
- intersezione
- differenza

tra query omogenee, ovvero con stesso numero di colonne e tipi compatibili.

```
<selectExpr> union <selectExpr>
<selectExpr> intersect <selectExpr>
<selectExpr> except <selectExpr>
```

Unione

```
select * from Impiegati
union
select * from Operai;
```

Nome	Stipendio
Bianchi	33000
Covelli	20000
Mambelli	23000
Neri	28000
Rossi	35000
Verdi	NULL

Table: `Impiegati`

Nome	Stipendio
Rossi	35000
Neri	28000
Bianchi	33000
Verdi	NULL

Table: `Operai`

Nome	Salario
Covelli	20000
Mambelli	23000
Neri	28000

Intersezione

```
select * from Impiegati                                    1
intersect                                                  2
select * from Operai;                                      3
```

Nome	Stipendio
Neri	28000

Table: `Impiegati`

Nome	Stipendio
Rossi	35000
Neri	28000
Bianchi	33000
Verdi	NULL

Table: `Operai`

Nome	Salario
Covelli	20000
Mambelli	23000
Neri	28000

Differenza

```
select * from Impiegati                    1
except                                     2
select * from Operai;                      3
```

Nome	Stipendio
Bianchi	33000
Rossi	35000
Verdi	NULL

Table: `Impiegati`

Nome	Stipendio
Rossi	35000
Neri	28000
Bianchi	33000
Verdi	NULL

Table: `Operai`

Nome	Salario
Covelli	20000
Mambelli	23000
Neri	28000

Nomi dei campi

A voler essere formalmente corretti i nomi delle colonne andrebbero uniformati prima di usare un'operazione insiemistica. Se non lo si fa SQL prende i nomi dalla prima `select`.

```
select Nome,Stipendio as Retribuzione from Impiegati          1
union                                                         2
select Nome,Salario as Retribuzione from Operai;              3
```

Nome	Retribuzione
Bianchi	33000
Covelli	20000
Mambelli	23000
Neri	28000
Rossi	35000
Verdi	NULL

Letture

Capitoli 5 e 6 del manuale SQL.