

Università degli Studi di Trieste

Corso di Laurea Magistrale in
INGEGNERIA CLINICA

ESERCITAZIONE: IL LINGUAGGIO SQL

Corso di Informatica Medica

Docente Sara Renata Francesca MARCEGLIA



Dipartimento di Ingegneria e Architettura



UNIVERSITÀ
DEGLI STUDI DI TRIESTE



SQL: SINTASSI BASE

Funzioni Principali di un Linguaggio di Interrogazione Strutturato

(SQL: Structured Query Language)

- Definizioni di dati
- Istruzioni di aggiornamento
- Interrogazioni

Modello di Interrogazione generica

SELECT <lista di attributi>

FROM <lista delle tabelle>

WHERE <condizione>



DEFINIZIONE DELLO SCHEMA

- Definizione dei dati: definizione di uno **schema di base di dati** e di tutte le sue **componenti**;
- si usa l'istruzione SQL **create schema**;
- componenti di uno schema sono:
 - domini
 - tabelle
 - indici
 - asserzioni
 - viste
 - privilegi



ELEMENTI DELLO SCHEMA

- **dominio**: insieme dei valori ammissibili che una colonna (un attributo) può assumere;
- **tabella**: insieme ordinato di colonne ed eventuali vincoli relativi ai valori degli attributi stessi;
- **indice**: struttura dati ausiliaria, associata ad una tabella, che permette di rendere più efficiente l'esecuzione di interrogazioni sulla tabella mediante l'introduzione di un **ordinamento** sui valori di uno o più attributi;



ELEMENTI DELLO SCHEMA

- **asserzione**: rappresenta dei vincoli non legati ad attributi o tabelle, ma direttamente connessi allo schema della base di dati;
- **vista**: è una tabella “virtuale”, il cui contenuto è legato al contenuto di altre tabelle della base di dati;
- **privilegio**: definizione delle modalità di accesso ai dati da parte di un certo utente.



CREATE TABLE

- Creare la tabella PAZIENTE con attributi
 - Nome
 - Cognome
 - Data di nascita
 - Sesso

```
create table Nome_tabella  
(  
  Nome_colonna    Dominio  [Valore_di_default]          [Vincoli]  
  { Nome_colonna  Dominio  [Valore_di_default]          [Vincoli]}  
  Altri_vincoli  
)
```

CREATE TABLE



create table *Paziente*

(

Cognome *varchar(20)* *not null,*

Nome *varchar(20)* *not null,*

Data_di_nchar(9),

Sesso *char(1)*

)



CREATE TABLE

```
create table Paziente_Cardiochirurgia
(
  Cognome      varchar(20)      not null      ,
  Nome         varchar(20)      not null      ,
  Luogo_di_n  varchar(30)      not null      ,
  Data_di_n   date             not null      ,
  Sesso       char(1)          ,
  HID         char(10)         primary key ,
  Istante_int timestamp       ,
  Dur_int     interval hour(2) to second ,
  freq_card_media decimal(5,2) ,
  Ospedale_prov varchar(30) references Ospedale(Nome) ,
  Num_prec_int integer          ,
unique (Cognome, Nome, Luogo_di_n, Data_di_n) ,
foreign key (Cognome, Nome, Luogo_di_n, Data_di_n) references Anagrafica(Cognome,
  Nome, Luogo_di_nascita, Data_di_nascita)
)
```




IL COMANDO SELECT

Sintassi essenziale del comando SELECT

```
select Espressione_su_colonna [ [as] Nuovo_nome_colonna ]  
    {, Espressione_su_colonna [ [as]  
    Nuovo_nome_colonna ] }  
from Nome_tabella [ [as] alias ]  
    {, Nome_tabella [ [as] alias ] }  
[ where Condizione ]
```

DAL LINGUAGGIO NATURALE ALLA QUERY



Traduzione di una query di select in linguaggio naturale
(traduzione linguaggio dichiarativo in procedurale):

1. Tra le righe ottenute dal prodotto cartesiano delle tabelle elencate nella clausola **from**,
2. vengono considerate quelle righe che soddisfano la condizione espressa nella clausola **where**;
3. su tali righe vengono valutate le espressioni sulle colonne indicate nella clausole **select**.



Tabella “Anagrafica”

Definizione della Tabella “Anagrafica”

- **Anagrafica** (Cognome, Nome, Data_di_n, Sesso)

Cognome	Nome	Data_di_n	Sesso
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M



Queries 1.1, 1.2 e 1.3

1.1 Conoscere cognome e nome dei pazienti maschi considerati nella base di dati;

$1.1 \leftarrow \pi_{(\text{Cognome}, \text{Nome})} \sigma_{(\text{Sesso}='M')} \text{Anagrafica}$

1.2 Conoscere tutti i dati relativi ai pazienti il cui cognome è “Bianchi”;

$1.2 \leftarrow \sigma_{(\text{Cognome}='Bianchi')} \text{Anagrafica}$

1.3 Conoscere cognome, nome, anno di nascita dei pazienti considerati.

! Definiamo l'operatore $\text{year}(\text{'YYYY-MM-DD'}) \rightarrow \text{YYYY}$

$1.3 \leftarrow \pi_{(\text{Cognome}, \text{Nome}, \text{year}(\text{Data_di_n}))} \text{Anagrafica}$

Query 1.1 : implementazione

Conoscere cognome e nome
considerati nella base di dati
dei pazienti maschi

SELECT *Cognome, Nome*

FROM *Anagrafica*

WHERE (*Sesso* = 'M' **OR** *Sesso* = 'm');

RISULTATO:

Cognome	Nome	Data_di_n	Sesso
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

Cognome	Nome
Bianchi	Luca
Aldobrandi	Enrico

Query 1.2 : implementazione

Conoscere tutti i dati

relativi ai pazienti

il cui cognome è “Bianchi”

SELECT *

FROM *Anagrafica*

WHERE (*Cognome*=`Bianchi`);

RISULTATO:

Cognome	Nome	Data_di_n	Sesso
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

Cognome	Nome	Data_di_n	Sesso
Bianchi	Luca	1962-05-08	M



Query 1.3 : implementazione

Conoscere cognome, nome, anno di nascita
relativi ai pazienti

```
SELECT Cognome, Nome, year(Data_di_n) AS Anno  
FROM Anagrafica
```

RISULTATO:

Cognome	Nome	Data_di_n	Sesso
Bianchi	Luca	1962-05-08	M
Mascheroni	Marinella	1965-12-02	F
Strozzi	Giulia	1964-02-11	F
Aldobrandi	Enrico	1960-02-29	M

Cognome	Nome	Anno
Bianchi	Luca	1962
Mascheroni	Marinella	1965
Strozzi	Giulia	1964
Aldobrandi	Enrico	1960

Tabelle “Paziente” e “Terapia”

Definizione della Tabelle “Paziente” e “Terapia”

- **Paziente** (ID% , Cognome, Nome , Data_di_n)

ID	Cognome	Nome	Data_di_n
1	Bianchi	Luca	08-05-62
2	Mascheroni	Marinella	02-12-65
3	Strozzi	Giulia	11-02-64
4	Aldobrandi	Enrico	29-02-60

- **Terapia** (ID% , Nome% , Inizio_ter , Fine_Ter)

ID	Nome	Inizio_ter	Fine_Ter
1	beta-bloccante	1990-05-09	1991-07-10
2	aspirina	1991-03-21	1991-05-31
3	nitroglicerina	1995-01-01	1996-03-01
2	paracetamolo	1990-11-30	1992-11-23



Queries 2.4, 2.5 e 2.6

definiamo la Vista accessoria $\text{Paz_Ter} \leftarrow \sigma_{(P.ID=T.ID)} \text{Paziente } P \times \text{Terapia } T$

2.4 Conoscere tutte le terapie associate ad ogni paziente;

$$2.4 \leftarrow \pi_{(\text{Cognome}, P.\text{Nome}, P.\text{Data_di_n}, T.\text{Nome})} \text{Paz_Ter}$$

2.5 Conoscere cognome e nome dei pazienti trattati con aspirina;

$$2.5 \leftarrow \pi_{(\text{Cognome}, P.\text{Nome})} \sigma_{(T.\text{Nome}='aspirina')} \text{Paz_Ter}$$

2.6 Conoscere cognome, nome, data di nascita, terapia per quei pazienti la cui terapia è finita prima dell'anno 1992.

$$2.6 \leftarrow \pi_{(\text{Cognome}, P.\text{Nome}, \text{Data_di_n}, T.\text{Nome})} \sigma_{(\text{year}(\text{Fine_ter}) < 1990)} \text{Paz_Ter}$$



Query 2.4 : implementazione

Conoscere tutte

le terapie associate ad ogni paziente

```
SELECT Cognome, P.Nome AS Nome, Data_di_n, T.Nome AS  
Terapia
```

```
FROM Paziente AS P, Terapia AS T
```

```
WHERE ( P.ID=T.ID );
```

RISULTATO:

Cognome	Nome	Data_di_n	Terapia
Bianchi	Luca	08-05-62	beta-bloccante
Mascheroni	Marinella	02-12-65	aspirina
Strozzi	Giulia	11-02-64	nitroglicerina
Mascheroni	Marinella	02-12-65	paracetamolo



Query 2.5 : implementazione

Conoscere cognome e nome
dei pazienti
trattati con aspirina

```
SELECT Cognome, P.Nome AS Nome  
FROM Paziente AS P, Terapia AS T  
WHERE ( P.ID=T.ID AND T.Nome='Aspirina' );
```

RISULTATO:

Cognome	Nome
Mascheroni	Marinella



Query 2.6 : implementazione

Conoscere cognome, nome, data di nascita, terapia
per quei pazienti
la cui terapia è finita prima dell'anno 1992

```
SELECT Cognome, P.Nome AS Nome  
FROM Paziente AS P, Terapia AS T  
WHERE ( P.ID=T.ID AND year(T.Fine_ter)<1992);
```

RISULTATO:

Cognome	Nome	Data_di_n	Terapia
Bianchi	Luca	08-05-62	beta-bloccante
Mascheroni	Marinella	02-12-65	aspirina

Queries 2.7, 2.8 e 2.9

2.7 Conoscere nome e cognome dei pazienti che hanno avuto terapie con inizio nell'anno 1995 o nell'anno 1990, senza considerare in quest'ultimo caso i pazienti nati prima del 1965;

$$2.7 \leftarrow \pi_{(\text{Cognome}, \text{P.Nome})} \sigma_{([\text{year}(\text{Inizio_ter})=1995] \text{ OR } [\text{year}(\text{Inizio_ter})=1990 \text{ AND } \text{year}(\text{Data_di_n}) > 1964])} \text{Paz_Ter}$$

2.8 Conoscere nome e cognome dei pazienti, senza considerare quelli nati prima del 1965, che hanno avuto terapie con inizio nell'anno 1995 o nell'anno 1990;

$$2.8 \leftarrow \pi_{(\text{Cognome}, \text{P.Nome})} \sigma_{([\text{year}(\text{Data_di_n}) > 1964] \text{ AND } [\text{year}(\text{Inizio_ter})=1995] \text{ OR } [\text{year}(\text{Inizio_ter})=1990])} \text{Paz_Ter}$$

2.9 Conoscere cognome e nome dei pazienti che hanno terapie iniziate prima del 1993.

$$2.9 \leftarrow \pi_{(\text{Cognome}, \text{P.Nome})} \sigma_{(\text{year}(\text{Inizio_ter}) < 1993)} \text{Paz_Ter}$$

Non è del tutto vera la precedente rappresentazione: in algebra relazionale non esistono istanze duplicate



Query 2.7 : implementazione

Conoscere nome e cognome

dei pazienti che hanno avuto terapie

con inizio nell'anno 1995 o nell'anno 1990, senza considerare in quest'ultimo caso i pazienti nati prima del 1965

```
SELECT Cognome, P.Nome AS Nome  
FROM Paziente AS P, Terapia AS T  
WHERE ( [P.ID=T.ID ] AND [[year(T.Inizio_ter)=1995] OR  
[year(T.Inizio_ter)=1990 AND year(P.Data_di_n)>1964 ]]);
```

RISULTATO:

Cognome	Nome
Mascheroni	Marinella
Strozzi	Giulia



Query 2.8 : implementazione

Conoscere nome e cognome,
dei pazienti, senza considerare quelli nati prima del 1965, che hanno
avuto terapie
con inizio nell'anno 1995 o nell'anno 1990

```
SELECT Cognome, P.Nome AS Nome  
FROM Paziente AS P, Terapia AS T  
WHERE ( [P.ID=T.ID ] AND [year(P.Data_di_n)>1964] AND  
[year(T.Inizio_ter)=1990 OR year(T.Inizio_ter)=1995]);
```

RISULTATO:

Cognome	Nome
Mascheroni	Marinella



Query 2.9 : implementazione

Conoscere cognome e nome
dei pazienti che hanno avuto terapie
iniziate prima del 1993

```
SELECT Cognome, P.Nome AS Nome  
FROM Paziente AS P, Terapia AS T  
WHERE ( [P.ID=T.ID ] AND [year(T.Inizio_ter)<1993]);
```

RISULTATO:

Cognome	Nome
Bianchi	Luca
Mascheroni	Marinella
Mascheroni	Marinella



JOIN

Sintassi

```
SELECT Espressione_su_colonna [ [AS] Nuovo_nome_colonna]
        {, Espressione_su_colonna [ [AS] Nuovo_nome_colonna] }
FROM Nome_tabella [ [AS] alias]
        {, < Nome_tabella [ [AS] alias] |
        Tipo_join join Nome_tabella [ [AS] alias] on Condizione_di_join > }
[ WHERE Condizione]
```

Tipo_join può essere:

- inner
- left (outer)
- right (outer)
- full (outer)



JOIN

Tipo `join` può essere:

- `inner` (join interno fra le due tabelle) corrisponde al theta-join del modello relazionale, dove la condizione viene espressa in `Condizione_di_join`;

JOIN



Tipo `join` può essere:

- `left (outer)` (join esterno sinistro) viene valutato, con la condizione espressa in `Condizione_di_join`, il join interno sulle due tabelle, e tale risultato è arricchito con le righe della tabella di sinistra che non hanno righe nella tabella di destra (aggiunte righe a destra);

JOIN



Tipo `join` può essere:

- **right (outer)** (join esterno destro) viene valutato, con la condizione espressa in `Condizione_di_join`, il join interno sulle due tabelle, e tale risultato è arricchito con le righe della tabella di destra che non hanno righe nella tabella di sinistra (aggiunte righe a sinistra);

JOIN



Tipo `join` può essere:

- **full (outer)** (join esterno completo) viene valutato, con la condizione espressa in `Condizione_di_join`, il join interno sulle due tabelle, e tale risultato è arricchito con le righe di entrambe le tabelle che non hanno righe corrispondenti nell'altra;



JOIN

Condizione_di_join rappresenta la condizione da verificare per stabilire la relazione fra le tuple presenti nelle due tabelle



Queries 2.10, 2.11 e 2.12

definiamo la Vista accessoria $\text{Paz_JOIN_Ter} \leftarrow \text{Paziente } P \bowtie_{(P.ID=T.ID)} \text{Terapia } T$

2.10 Conoscere cognome e nome dei pazienti che hanno terapie iniziate prima del 1993, ma senza duplicati;

Equivalente alla 2.9 (l'unicità delle tuple è garantita dall'algebra relazionale)

2.11 Conoscere tutte le terapie associate ad ogni paziente;

$2.11 \leftarrow \pi_{(\text{Cognome}, P.\text{Nome}, P.\text{Dta_di_n}, T.\text{Nome})} \text{Paz_JOIN_Ter}$

2.12 Conoscere i dati dei pazienti e delle loro terapie, considerando anche quei pazienti che al momento non hanno terapie riportate nella base di dati.

$2.12 \leftarrow \text{Paziente } P \bowtie_{(P.ID=T.ID)} \text{Terapia } T$



Query 2.10 : implementazione

Conoscere cognome e nome

dei pazienti che hanno terapie

iniziate prima del 1993, ma senza duplicati;

```
SELECT DISTINCT Cognome, P.Nome AS Nome
```

```
FROM Paziente AS P, Terapia AS T
```

```
(oppure FROM Paziente P JOIN Terapia T ON P.ID=T.ID)
```

```
WHERE ( [P.ID=T.ID ] AND [year(T.Inizio_ter)<1993]);
```

RISULTATO:

Cognome	Nome
Bianchi	Luca
Mascheroni	Marinella



Query 2.11 : implementazione

Conoscere tutte le terapie associate ad ogni paziente
dei pazienti che hanno terapie

```
SELECT Cognome, P.Nome AS Nome, Data_di_n, T.Nome AS Terapia  
FROM Paziente AS P INNER JOIN Terapia AS T ON P.ID=T.ID;
```

RISULTATO:

Cognome	Nome	Data_di_n	Terapia
Bianchi	Luca	1962-05-08	beta-bloccante
Mascheroni	Marinella	1965-12-02	aspirina
Mascheroni	Marinella	1965-12-02	paracetamolo
Strozzi	Giulia	1964-02-11	nitroglicerina



Query 2.12 : implementazione

Conoscere i dati dei pazienti e delle loro terapie, considerando anche quei pazienti che al momento non hanno terapie riportate nella base di dati

```
SELECT Cognome, P.Nome AS Nome, Data_di_n, T.Nome AS Terapia  
FROM Paziente AS P LEFT [OUTER] JOIN Terapia AS T ON  
P.ID=T.ID;
```

RISULTATO:

Cognome	Nome	Data_di_n	Terapia
Bianchi	Luca	1962-05-08	beta-bloccante
Mascheroni	Marinella	1965-12-02	aspirina
Mascheroni	Marinella	1965-12-02	paracetamolo
Strozzi	Giulia	1964-02-11	nitroglicerina
Aldobrandi	Enrico	1960-02-29	NULL

Operatori Aggregati

SQL mette a disposizione **5 operatori aggregati** che permettono di operare su più righe di una tabella, e **non hanno riscontro nell'algebra relazionale:**

count (< * | [distinct | all] *Lista_colonne* >)

* conta il numero delle righe per le colonne in *Lista_colonne*;

distinct non considera i duplicati;

all (default) non vengono considerati i valori nulli;

Operatori Aggregati

sum (< [distinct | all] *Espressione_su_colonne* >)

avg (< [distinct | all] *Espressione_su_colonne* >)

Per sum ed avg si devono considerare espressioni che rappresentano valori numerici o durate temporali.

min (< [distinct | all] *Espressione_su_colonne* >)

max (< [distinct | all] *Espressione_su_colonne* >)

Per min ed max si devono considerare valori sui quali è definito un criterio di ordinamento.

distinct non considera i duplicati;

all non vengono considerati i valori nulli;



Queries 2.13, 2.14 e 2.15

2.13 Conoscere cognome, nome, e le terapie (a coppie) di quei pazienti che hanno avuto più terapie concomitanti;

2.13 $\leftarrow \pi_{(\text{Cognome}, \text{Nome}, \text{Ter}_1, \text{Ter}_2)} \rho_{(\text{P.Nome}, \text{T1.Nome}, \text{T2.Nome} \leftarrow \text{Nome}, \text{Ter}_1, \text{Ter}_2)} \sigma_{(\text{P.ID} = \text{T1.ID} \text{ AND } \text{P.ID} = \text{T2.ID} \text{ AND } \text{T1.Inizio_ter} \leq \text{T2.Inizio_ter} \text{ AND } \text{T1.Inizio_ter} \leq \text{T2.Fine_ter} \text{ AND NOT } \text{T1.Nome} = \text{T2.Nome})}$ Paziente P \times Terapia T1 \times Terapia T2

2.14 Ottenere l'elenco ordinato alfabeticamente per cognome dei pazienti considerati nella tabella "Paziente";

L'operazione di ordinamento non è presente nell'algebra relazionale

2.15 Ottenere il numero dei pazienti con almeno una terapia; Gli operatori aggregati non sono presenti nell'algebra relazionale



Query 2.13 : implementazione

Conoscere cognome, nome, e le terapie (a coppie) di quei pazienti che hanno avuto più terapie concomitanti

```
SELECT Cognome, P.Nome AS Nome, T1.Nome AS Ter_1, T2.Nome  
AS Ter_2  
  
FROM Paziente P, Terapia T1, Terapia T2  
WHERE P.ID = T1.ID AND P.ID = T2.ID AND  
((T1.Inizio_ter <= T2.Inizio_ter AND T2.Inizio_ter <= T1.Fine_ter)  
OR  
(T2.Inizio_ter <= T1.Inizio_ter AND T1.Inizio_ter <= T2.Fine_ter))  
  
AND NOT T1.Nome = T2.Nome;
```

Verifico che l'inizio di T2 sia compreso nell'intervallo di T1 oppure che l'inizio di T1 sia compreso nell'intervallo di T2

RISULTATO:

Cognome	Nome	Ter_1	Ter_2
Mascheroni	Marinella	paracetamolo	aspirina



Query 2.14 : implementazione

Ottenere l'elenco ordinato alfabeticamente per cognome dei pazienti considerati nella tabella "Paziente"

```
SELECT Cognome, Nome  
FROM Paziente  
ORDER BY Cognome ASC;
```

RISULTATO:

Cognome	Nome
Aldobrandi	Enrico
Bianchi	Luca
Mascheroni	Marinella
Strozzi	Giulia



Query 2.15 : implementazione

Ottenere il numero dei pazienti
con almeno una terapia

```
SELECT count(DISTINCT Cognome, P.Nome) AS N_pz_con_ter  
FROM Paziente P, Terapia T  
WHERE P.ID=T.ID;
```

RISULTATO:

<u>N_pz_con_ter</u>
3



Queries 2.16, 2.17, 2.18 e 2.19

2.16 Ottenere il numero delle terapie registrate nella base di dati, considerando la sola tabella “Terapia”;

Gli operatori aggregati non sono presenti nell'algebra relazionale

2.17 Ottenere l'indicazione, per ogni paziente con almeno una terapia, del numero di terapie e del codice;

Gli operatori aggregati non sono presenti nell'algebra relazionale

2.18 Ottenere l'indicazione, per ogni paziente con terapie, del numero di terapie e dei dati anagrafici del paziente;

Gli operatori aggregati non sono presenti nell'algebra relazionale

2.19 Ottenere l'indicazione, per ogni paziente con almeno due terapie, del numero di terapie e dei dati anagrafici del paziente;

Gli operatori aggregati non sono presenti nell'algebra relazionale



Query 2.16 : implementazione

Ottenere il numero delle terapie registrate nella base di dati, considerando
la sola tabella “Terapia”

```
SELECT count(*) AS N_ter  
FROM Terapia;
```

RISULTATO:

N_ter
4



Query 2.17 : implementazione

Ottenere l'indicazione, per ogni paziente con almeno una terapia, del numero di terapie e del codice

```
SELECT P.ID AS Paz_ID, count(*) AS Num_ter  
FROM Paziente P, Terapia T  
WHERE P.ID=T.ID  
GROUP BY P.ID;
```

RISULTATO:

Paz_ID	Num_ter
1	1
2	2
3	1



Query 2.18 : implementazione

Ottenere l'indicazione, per ogni paziente con terapie, del numero di terapie e dei dati anagrafici del paziente

```
SELECT Cognome, P.Nome AS Nome, count(*) AS Num_ter  
FROM Paziente P, Terapia T  
WHERE P.ID=T.ID  
GROUP BY P.ID, Cognome, P.Nome;
```

RISULTATO:

Cognome	NomePaz_ID	Num_ter
Bianchi	Luca	1
Mascheroni	Marinella	2
Strozzi	Giulia	1



Query 2.19 : implementazione

Ottenere l'indicazione, per ogni paziente con almeno due terapie, del numero di terapie e dei dati anagrafici del paziente

```
SELECT Cognome, P.Nome AS Nome, count(*) AS Num_ter
FROM Paziente P, Terapia T
WHERE P.ID=T.ID
GROUP BY P.ID, Cognome, P.Nome
HAVING COUNT(*) >=2;
```

RISULTATO:

Cognome	NomePaz_ID	Num_ter
Mascheroni	Marinella	2



SELECT di SQL: sintassi

Select_di_SQL ::=

SELECT *Espressione_su_colonna* [**AS** *Nuovo_nome_colonna*]
 {, *Espressione_su_colonna* [**AS** *Nuovo_nome_colonna*] }

FROM *Nome_tabella* [**AS** *alias*]

 {, < *Nome_tabella* [**AS** *alias*] |

Tipo_join **JOIN** *Nome_tabella* [**AS** *alias*]

ON *Condizione_di_join* > }

[**WHERE** *Condizione*]

[**GROUP BY** *Nome_colonna* {, *Nome_colonna* }]

[**HAVING** *Condizione_su_righe_aggregate*]

[**ORDER BY** *Nome_colonna* [**ASC** | **DESC**]

 {, *Nome_colonna* [**ASC** | **DESC**] }]



Operazioni di Insiemistica

Sono operazioni legate alla manipolazione di insiemi, quindi di insiemi di tuple.

Sono 3 principali:

- union
- intersect
- except

La sintassi d'uso prevede che congiungano due query di selezione

Select_di_SQL {< **union** | **intersect** | **except** > [**all**] Select_di_SQL}



Operazioni di Insiemistica

Union : svolge funzione analoga all'unione insiemistica, restituendo tutte le righe di entrambe le tabelle (rimuovendo i duplicati)

Intersect : svolge funzione analoga all'intersezione insiemistica , restituendo soltanto le righe comuni ad entrambe le tabelle

Except : svolge funzione analoga alla differenza insiemistica , restituendo soltanto le righe presenti nella prima tabella ma non presenti nella seconda

Il parametro **all** permette di mantenere gli eventuali duplicati

Tabelle “Paz_dia_1” e “Paz_dia_2”

Definizione della Tabelle “Paz_dia_1” e “Paz_dia_1”

- **Paz_dia_1 (ID%, Cognome, Data, Diagn, Medico_curante)**

ID	Cognome	Data	Diagn	Medico_curante
1123	Aldobrandi	12/11/1985	I.M.A.	Porpora
1763	Bianchi	31/03/1979	I.M.A.	De Esculapi
2156	Mascheroni	22/02/1982	B.B.S.	Conte

- **Paz_dia_2 (ID%, Cognome, Data, Diagn, Medico_curante)**

ID	Cognome	Data	Diagn	Medico_curante
1123	Aldobrandi	12/11/1985	I.M.A.	Porpora
2347	Danieli	31/05/1970	Angina	Derossi
8895	Riva	17/05/1993	Ipertensione	Sassi



Queries 2.20 e 2.21

2.20 Ottenere l'insieme dei cognomi dei pazienti considerati nelle tabelle "Paz_dia_1" e "Paz_dia_2" ;

$$2.20 \leftarrow \pi_{(\text{Cognome})} \text{Paz_dia_1} \cup \text{Paz_dia_2}$$

2.21 Ottenere il cognome dei pazienti considerati nella tabella "Paz_dia_1" che hanno patologie delle quali non soffre nessun paziente considerato nella tabella "Paz_dia_2"

L'operatore ALL non è presente nell'algebra relazionale



Query 2.20 : implementazione

Ottenere l'insieme dei cognomi dei pazienti considerati nelle tabelle
“Paz_dia_1” e “Paz_dia_2”

```
SELECT Cognome  
FROM Paz_dia_1  
UNION  
SELECT Cognome  
FROM Paz_dia_2;
```

RISULTATO:

Cognome
Aldobrandi
Bianchi
Mascheroni
Danieli
Riva



Query 2.21 : implementazione

Ottenere il cognome dei pazienti considerati nella tabella “Paz_dia_1” che hanno patologie delle quali non soffre nessun paziente considerato nella tabella “Paz_dia_2”

```
SELECT Cognome
FROM Paz_dia_1
WHERE Diagn <> ALL (SELECT Diagn
                    FROM Paz_dia_2
                    );
```

RISULTATO:

Cognome
Mascheroni

Testo di Riferimento

F. Pincioli, C. Combi, G. Pozzi

**BASI DI DATI PER L'INFORMATICA
MEDICA - CONCETTI LINGUAGGI
APPLICAZIONI [Cap. 5]**

Pàtron Editore, Bologna 1998

Per Approfondire

P. Atzeni, S. Ceri, S. Paraboschi e R. Torlone

BASI DI DATI - Seconda Edizione

McGraw-Hill, Italia 1999