Università degli Studi di Padova
Corso di Laurea Magistrale in Bioingegneria

A.A. 2010-2011

INFORMATICA SANITARIA
(Lezione SQL 3)

Barbara Di Camillo
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Padova
Via Ognissanti 72, 35129 Padova
e-mail: barbara.dicamillo@dei.unipd.it

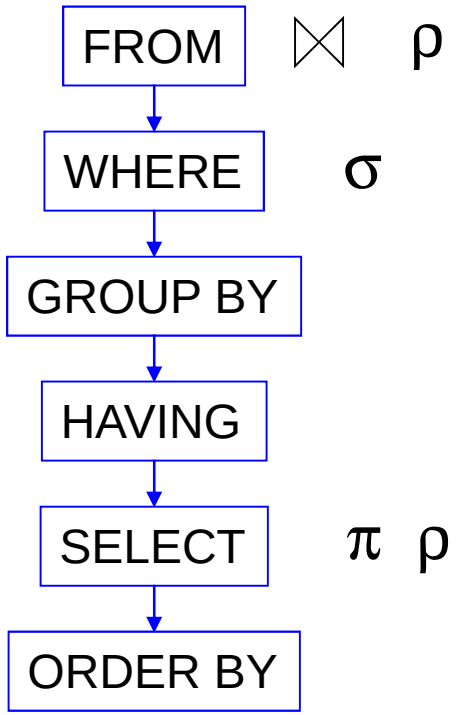
Interrogazione di Tabelle: l’istruzione SELECT e l’algebra relazionale

SELECT e l'algebra relazionale

```
SELECT rimborso
FROM drg
WHERE rimborso>1000
GROUP BY rimborso HAVING COUNT(*)>1
ORDER BY rimborso
```

$\pi_{rimborso}(\sigma_{rimborso>1000}(drg))$

- L'istruzione SELECT permette di implementare gli operatori di algebra relazionale (tranne l'UNIONE) e fornisce delle funzionalità aggiuntive



Operatori intersezione e differenza

2 tabelle (con lo stesso schema) → 1 tabella (mantiene lo schema)

Prescrizioni(codice, nome,costo)
Prescr_old(codice, nome,costo)

→

Prescrizioni ∩ Prescr_old

Prescrizioni - Prescr_old

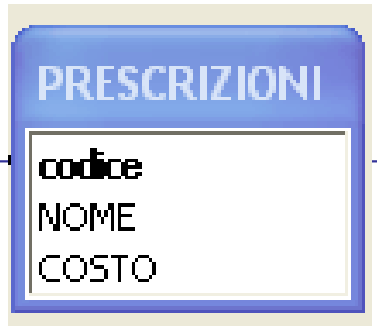
Agisco sulle righe; mantengo tutte le colonne

- **INTERSEZIONE (∩)**

```
SELECT Prescrizioni.*
FROM Prescrizioni, Prescr_old
WHERE
Prescrizioni.codice=Prescr_old.codice;
```

- **DIFFERENZA (-)**

```
SELECT Prescrizioni.*
FROM Prescrizioni
WHERE Prescrizioni.codice NOT IN
(SELECT codice FROM Prescr_old);
```



Alcune note…

- SELECT Prescrizioni.*
  FROM Prescrizioni, Prescr_old
  WHERE Prescrizioni.codice=Prescr_old.codice;

Indica la selezione di tutte le colonne

- SELECT prescrizioni.*
  FROM Prescrizioni
  WHERE Prescrizioni.codice NOT IN
  (SELECT codice FROM Prescr_old);

Specifica “full”:
tabella e attributo
Evita ambiguità e rende più facile la lettura!

Con gli operatori insiemistici devo usare una condizione sulla chiave.
Se la chiave è composita userò la parola chiave AND.
Ad esempio, se la chiave fosse (codice,costo):

```
SELECT Prescrizioni.*
FROM Prescrizioni, Prescr_old
WHERE
Prescrizioni.codice=Prescr_old
.codice
AND
Prescrizioni.costo=Prescr_old.
costo;
```

Operatore unione

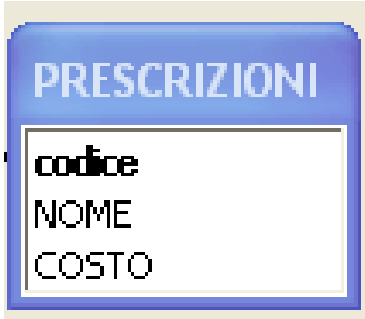
2 tabelle (con lo stesso schema) → 1 tabella (mantiene lo schema)

Prescrizioni(codice, nome,costo)
Prescr_old(codice, nome,costo) → Prescrizioni ∪ Prescr_old

Agisco sulle righe; mantengo tutte le colonne

- **UNIONE (∪)**

```
(SELECT * FROM Prescrizioni)
UNION
(SELECT * FROM Prescr_old);
```



Operatore di ridenominazione,

1 tabella → 1 tabella (con schema uguale o diverso)

- **RIDENOMINAZIONE (ρ)**

RIDENOMINAZIONE TABELLA

```
SELECT *
INTO Prescrizioni_new
FROM Prescrizioni;

SELECT *
FROM Prescrizioni AS Prescrizioni_new;
```

Utile per fare self-join, come vedremo…

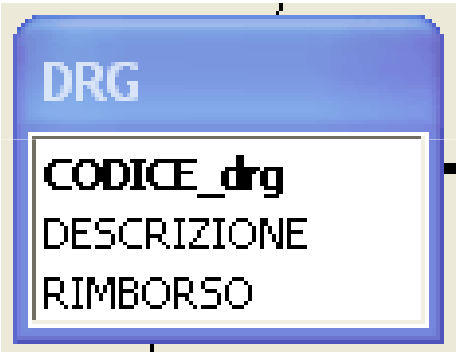
RIDENOMINAZIONE COLONNE

```
SELECT  codice AS id, nome, costo
FROM Prescrizioni;
```

NOTA: l'output di una query è una tabella "virtuale" (se non ho usato INTO <nome-nuova-tabella>)

Operatore AS

- AS si usa anche per dare il nome a nuove colonne oltre che per rinominare colonne già esistenti
- In genere conviene usarlo quando si fanno *elaborazioni sui dati*



Esempi:

- SELECT rimborso/1000 FROM drg ;
- SELECT rimborso/1000 AS migliaia_di_Euro FROM drg ;
- SELECT drg.* , round(rimborso/1000) AS migliaia_di_Euro FROM drg ;

Operatori di proiezione e selezione

1 tabella → 1 tabella (con schema uguale o diverso)

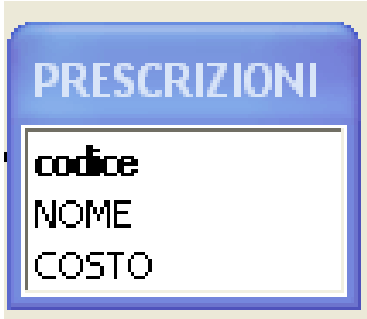
- **PROIEZIONE (π)**

  Estraggo colonne!!!

```
SELECT DISTINCT codice, nome
FROM Prescrizioni;
```

- **SELEZIONE (σ)**

  Estraggo righe!!!




```
SELECT *
FROM Prescrizioni
WHERE costo<100;
```

Esempi

- SELECT cognome, nome FROM generale
  WHERE nascita IS NULL;

$\pi_{nome,\ cognome}\ (\sigma_{nascita=NULL}\ (generale))$

- SELECT nome,cognome,nascita
  FROM generale
  WHERE provincia="PD";

$\pi_{nome,\ cognome,\ nascita}(\sigma_{provincia=PD}(generale))$

- SELECT nome,cognome,nascita
  FROM generale
  WHERE provincia="PD" AND sesso="F";

$\pi_{nome,\ cognome,\ nascita}(\sigma_{provincia=PD\ AND\ sesso=F}\ (generale))$

- SELECT codice_fisc, reparto
  FROM dipendenti
  WHERE codice_fisc IN (SELECT codice_fisc FROM ricoveri);

$\pi_{codice\_fisc,\ reparto}\ (\sigma_{codice\_fisc\ \in\ (\pi\ codice\_fisc(ricoveri))}(dipendenti))$

Operatori di JOIN
(prod.cartesiano, join naturale, theta join)

2 tabelle (con schemi anche diversi)→ 1 tabella (con schema “join”)

RIPASSO:

•Il **prodotto cartesiano** di due relazioni è la relazione che si ottiene semplicemente **affiancando** ad ogni tupla della prima tabella tutte le tuple della seconda tabella.

•La **join naturale** è un operazione che concatena dati in relazioni diverse sulla base di **valori uguali** in **attributi con lo stesso nome e lo stesso significato** (in genere coinvolge una chiave esterna e una primaria). *Se non ci sono campi in comune degenera nel prodotto cartesiano.*

•La **theta-join** è un’operazione che consente di correlare due relazioni sulla base del **confronto** tra i valori delle relazioni relativi ad **attributi con nome diverso ma con significato comune.**
Se la condizione è l’uguaglianza si parla in particolare di **equi-join.**

Prodotto Cartesiano

Esprimiamo l'operazione Prodotto cartesiano con l'istruzione SELECT in cui:

- dopo SELECT usiamo * (selezioniamo tutte le colonne)
- nella clausola FROM viene referenziata più di una tabella
- Non specifichiamo la clausola WHERE

```
SELECT *
    FROM Prescrizioni,drg;
```

Join Naturale e Theta join

In SQL esprimiamo tutte e due le operazioni di Join nello stesso modo

- Se la condizione di JOIN è un'egualglianza tra attributi con lo stesso nome e lo stesso campo ho un join naturale
- Se la condizione di JOIN è tra attributi con nome diverso (ma lo stesso campo) ho un theta-join (se poi la condizione è l'eguaglianza ho un equi-join)
- Va specificato se uso un INNER JOIN o un OUTER (LEFT o RIGHT) JOIN:

```
SELECT *
FROM <nome_tab_left>
INNER JOIN | LEFT JOIN | RIGHT JOIN    < nome_tab_right>
ON <condizione di JOIN>;
```

Join Naturale e Theta join

```
SELECT *
   FROM generale INNER JOIN dipendenti
   ON generale.codice_fisc=dipendenti.codice_fisc;
```

Generale ⋈ Dipendenti

```
SELECT *
   FROM interventi INNER JOIN dipendenti
   ON interventi.cf_medico=dipendenti.codice_fisc;
```

Interventi $\bowtie_{cf\_medico=codice\_fisc}$ Dipendenti

```
SELECT *
 FROM interventi INNER JOIN  visite
   ON     interventi.data_intervento< visite.data ;
```

Interventi $\bowtie_{Data\_intervento<data}$ Visite

```
SELECT *
 FROM interventi INNER JOIN  visite
   ON     interventi.codice_fisc = visite.cf_paziente
WHERE data_intervento<data;
```

Interventi $\bowtie_{cf\_medico=codice\_fisc \text{ AND } Data\_intervento<data}$ Visite

INNER e OUTER JOIN

```
SELECT *
   FROM generale INNER JOIN dipendenti
    ON  generale.codice_fisc=dipendenti.codice_fisc;
```

Generale ⋈ Dipendenti

```
SELECT *
   FROM generale LEFT JOIN dipendenti
    ON  generale.codice_fisc=dipendenti.codice_fisc;
```

Generale ⋈ LEFT Dipendenti

```
SELECT *
   FROM generale RIGHT JOIN dipendenti
    ON  generale.codice_fisc=dipendenti.codice_fisc;
```

Generale ⋈ RIGHT Dipendenti

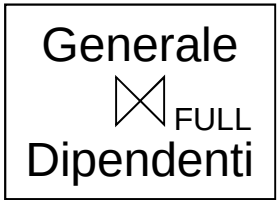
```
SELECT *
   FROM generale FULL JOIN dipendenti
    ON  generale.codice_fisc=dipendenti.codice_fisc;
```

Generale ⋈ FULL Dipendenti

FULL JOIN in Access

In Access non posso usare la parola chiave FULL (non è definita); quindi faccio un'unione tra un left e un right JOIN:

```
(SELECT *  FROM generale LEFT JOIN dipendenti
 ON  generale.codice_fisc=dipendenti.codice_fisc)
UNION
(SELECT *  FROM generale RIGHT JOIN dipendenti
ON  generale.codice_fisc=dipendenti.codice_fisc);
```



SELF JOIN tramite SELECT

- Si possono inoltre utilizzare dei nomi interni (detti range variables) per denotare le tabelle e/o le colonne
- L'uso delle range variables è indispensabile quando si debbano usare le stesse colonne della stessa tabella più volte, come nelle SELF JOIN

Esempio di Join della tabella con se stessa (SELF JOIN)
Trovare i coetanei (nome, cognome e nascita) nella tabella "generale".

```
SELECT DISTINCT gen1.nome, gen1.cognome,gen1.nascita
FROM generale AS gen1 INNER JOIN generale AS gen2
ON gen1.nascita=gen2.nascita
WHERE gen1.codice_fisc<>gen2.codice_fisc;
```

$$\pi(\sigma_{gen1.codice\_fisc<>gen2.codice\_fisc}(\rho_{gen1}(generale) \bowtie_{gen1.nascita=gen2.nascita} \rho_{gen2}(generale))$$

Esercizio (difficile)

Si consideri la tabella ALBERO (si escludono omonimie)

ALBERO

Genitore	Figlio
Luca Rossi	Anna Rossi
Maria Pavan	Anna Rossi
Giorgio Rossi	Luca Rossi
Silvia Di Carlo	Maria Pavan
Enzo Pavan	Maria Pavan

Attraverso operazioni di ridenominazione, join, proiezione, produrre la tabella sotto

Nonno	Nipote
Giorgio Rossi	Anna Rossi
Silvia Di Carlo	Anna Rossi
Enzo Pavan	Anna Rossi

Esercizio (difficile)

ALBERO

Genitore	Figlio
Luca Rossi	Anna Rossi
Maria Pavan	Anna Rossi
Giorgio Rossi	Luca Rossi
Silvia Di Carlo	Maria Pavan
Enzo Pavan	Maria Pavan

Genitore	Figlio
Luca Rossi	Anna Rossi
Maria Pavan	Anna Rossi
Giorgio Rossi	Luca Rossi
Silvia Di Carlo	Maria Pavan
Enzo Pavan	Maria Pavan

```
SELECT alb1.genitore AS nonno, alb2.figlio AS nipote
FROM (albero AS alb1) INNER JOIN (albero AS alb2)
ON alb1.figlio=alb2.genitore;
```

Uso delle join annidate

```
SELECT visite.data, generale.nome, generale.cognome, prescrizioni.nome,
    prescrizioni.costo
FROM (visite INNER JOIN prescrizioni ON visite.prescrizione=prescrizioni.codice)
    INNER JOIN generale ON visite.cf_medico=generale.codice_fisc;
```

La join annidata corrisponde a una tabella virtuale

- che è join tra la tabella **visite** e la tabella **prescrizioni**
- che è coinvolta in una join con la tabella **generale**

Uso delle join annidate

```
SELECT visite.data, generale.nome, generale.cognome, prescrizioni.nome,
    prescrizioni.costo
FROM (visite INNER JOIN prescrizioni ON visite.prescrizione=prescrizioni.codice)
    INNER JOIN generale ON visite.cf_medico=generale.codice_fisc;
```

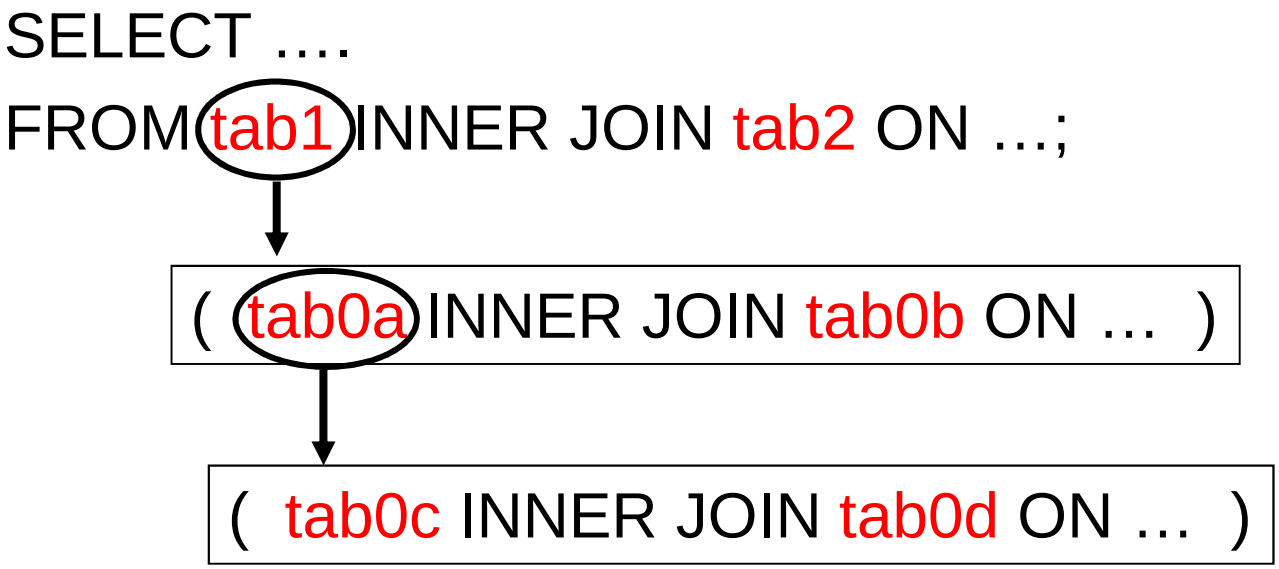
La condizione nella Join principale coinvolge la tabella **generale** e la tabella virtuale ottenuta dalla join annidata. Uso la specifica full per indicare la colonna della tabella virtuale a cui faccio riferimento.

La condizione imposta nella Join annidata coinvolge le due tabelle **visite** a **prescrizioni**

Tabella virtuale ottenuta con la join annidata:

VISITE.CF_PAZIENTE	VISITE.DATA	VISITE.ALTEZZA	VISITE.PESO	VISITE.TEMPERATURA	VISITE.P_DIASTOLICA	VISITE.P_SISTOLICA	VISITE.PRESCRIZIONE	VISITE.NOTE	VISITE.CF_MEDICO	PRESCRIZIONE.codice	PRESCRIZIONE.NOME	PRESCRIZIONE.COSTO
BRTPGR46T10A059G	12-Oct-02	180	80	38	80	120	1		ZNDLCU65H58A465M	1	emocromo	100
RCCCMN25P62B345T	06-Jan-00	160	75	39	75	110	3		DRSNNA66E48G224K	3	penicillina	50
MGLNMR99T52L577W	02-May-01	100	35	36	80	120	4		PLVJRN68C08D383W	4	antibiotici	40
BRTPGR46T10A059G	12-Oct-02	180	80	38	80	120	5		ZNDLCU65H58A465M	5	anticoagulanti	60
RCCCMN25P62B345T	10-Oct-00	160	70	37	80	100	6		TSTSST60D26G224A	6	dialisi	300
TNTNDR51C12A001A	16-Aug-04	175	90	37	90	140	7		DRSVNT62A61G570R	7	RX torace	100
RNZRNZ97T14L349T	06-Jun-03	110	45	38	70	100	9		ZNDLCU65H58A465M	9	TC encefalo	150
TNTNDR51C12A001A	16-Aug-04	175	90	37	90	140	10		DRSVNT62A61G570R	10	antiipertensivi	75
LMPDRA32P22A726F	07-Dec-97	170	68	38	110	190	10	iperteso	DRNPLA65D29A161R	10	antiipertensivi	75

Uso delle join annidate



- ✓ Le join vanno annidate in posizione corrispondente alla prima tabella (dopo FROM)
- ✓ Le join annidate sono racchiuse tra parentesi tonde
- ✓ Va sempre utilizzata la notazione full: nome_tabella.nome_colonna

Schema Riassuntivo

	funzioni e operatori	parole chiave
SELECT	proiezione	DISTINCT | ALL
	ridenominazione colonna	AS
	ridenominazione tabella (creazione)	INTO
	operazioni sui dati	,+, -, /, *
	funzioni di aggregazione	SUM, AVG, COUNT, MIN, MAX
FROM	ridenominazione tabella	AS
	Join	INNER JOIN | LEFT JOIN | RIGHT JOIN, ON
WHERE**	operatori di uguaglianza/ disuguaglianza	>,<,=,<=,>=,<>,
	operatori logici	NOT, AND, NULL
	altri operatori	BETWEEN, IN, LIKE, IS NULL
	predicati	IN, EXIST, ANY, ALL

* in rosso sono indicati gli operatori corrispondenti all'algebra relazionale

** la clausola WHERE corrisponde all'operatore SELEZIONE in algebra relazionale

Gli operatori insiemistici intersezione e differenza non corrispondono ad un comando SQL, ma sono implementabili con l'istruzione SELECT.
Per l'unione bisogna usare il comando UNION

Esercizio 1

Operazioni di Join e Query annidate

Trovare Nome e Cognome dei pazienti ricoverati nel reparto di medicina interna

Mi faccio restituire nei vari livelli di annidamento:

1. Codice del reparto ‘medicina interna’
2. CF dei pazienti ricoverati a medicina interna
3. Nome e cognome dei pazienti con CF ricavato al punto 2

```
SELECT nome, cognome FROM generale
WHERE codice_fisc IN
(SELECT codice_fisc FROM ricoveri
WHERE reparto = (SELECT codice FROM reparti WHERE
nome="medicina interna"));
```

Realizzare la stessa query usando un Join tra generale e ricoveri e una sola query annidata

Esercizio 2

Operazioni di Join e Query annidate

Qual è il nome e cognome e il reparto dei dipendenti dell’ospedale che sono stati ricoverati?

Suggerimento:

- JOIN tra generale e dipendenti per ricavare nome, cognome e reparto
- Con la condizione che i dipendenti siano stati ricoverati

Esercizio 3

Operazioni di Join e Query annidate

a) Creare una rubrica “rubrica_dipendenti” con il cognome, nome, indirizzo, comune provincia e telefono dei dipendenti attualmente assunti

b) Trovare il costo effettivo di ognuna delle prescrizioni effettuate in seguito a una visita (indicare data della visita, cf del medico, nome e costo della prescrizione)

c) Trovare il costo effettivo di ognuna delle prescrizioni effettuate in seguito a una visita (indicare data della visita, nome del medico, nome e costo della prescrizione)