Basi di dati - Laboratorio

Corso di Laurea in Bioinformatica

Docente: Barbara Oliboni

Lezione 4

Contenuto della lezione

- Interrogazioni SQL
  - Join interni ed esterni
  - Uso di variabili tupla o ALIAS
  - Interrogazioni di tipo insiemistico
  - Interrogazioni nidificate
    - ANY ed ALL
  - Interrogazioni nidificate complesse
    - EXISTS e NOT EXISTS
- Viste SQL

Base di Dati usata negli esempi

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso

Insegnamento

Codice	Nome_ins	Numero_crediti

Docente

Matricola	Cognome	Nome	Telefono	Stipendio

Esame

Codice_ins	Studente	Voto

Docenza

Codice_ins	Docente	Numero_studenti

Join interni ed esterni

- SQL-2 ha introdotto una sintassi alternativa per l'espressione dei join che permette di distinguere le condizioni di join dalle condizioni di selezione sulle tuple.

inner,
right outer,
left outer,
full outer

```
SELECT AttrExpr [ [AS] Alias ]
       {, AttrExpr [ [AS] Alias ] }
FROM Tabella [ [AS] Alias ]
{ TipoJoin JOIN Tabella [ [AS] Alias ]
           ON CondizioneDiJoin }
[ WHERE AltraCondizione]
```

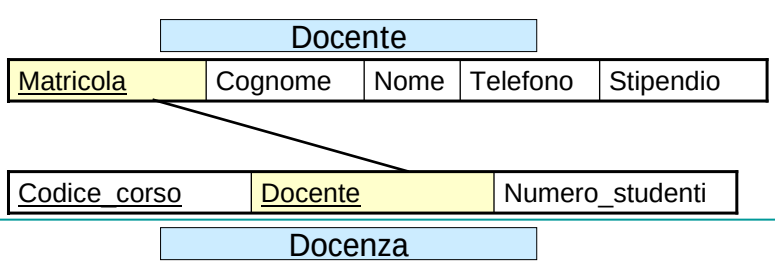
Join interni ed esterni

- INNER JOIN (JOIN INTERNO): rappresenta il tradizionale thetajoin dell'algebra relazionale. Con questo JOIN vengono selezionate solo le tuple del prodotto cartesiano per cui la condizione è vera.
- OUTER JOIN (JOIN ESTERNO): viene eseguito il JOIN mantenendo tutte le tuple che fanno parte di una o entrambe le tabelle.
  - LEFT JOIN: fornisce come risultato il join interno esteso con le tuple della relazione che compare a sinistra nel join per le quali non esiste una corrispondente tupla nella tabella di destra ("tuple escluse").
  - RIGHT JOIN: restituisce oltre al join interno, le "tuple escluse" della relazione di destra.
  - FULL JOIN: restituisce il join interno esteso con le "tuple escluse" di entrambe le relazioni.

Esempio 1

- Visualizzare cognome e nome dei docenti che tengono un corso con più di 100 studenti

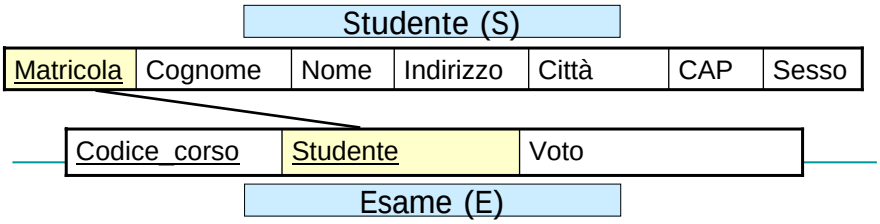
```
SELECT Docente.Cognome, Docente.Nome
FROM Docente INNER JOIN Docenza
      ON (Docente.Matricola= Docenza. Docente)
WHERE Docenza.Numero_studenti > 100;
```



Esempio 2

- Visualizzare il cognome degli studenti che hanno preso 30 nell'esame con codice uguale a "INF01".

```
SELECT Cognome
FROM Studente S INNER JOIN Esame E
        ON (S.Matricola = E.Studente)
WHERE E.Voto = 30;
```



Uso di variabili tupla o ALIAS (1)

- SQL consente di associare un nome alternativo alle tabelle che compaiono come argomento della clausola FROM.
- Il nome viene usato per far riferimento alla tabella nel contesto dell'interrogazione e viene detto ALIAS.

```
SELECT S.Cognome
FROM Studente S
WHERE  S.Città = 'Padova';
```

- In questo caso (in cui la tabella compare una sola volta nell'interrogazione) l'ALIAS viene considerato come uno pseudonimo.

Uso di variabili tupla o ALIAS (2)

- Ogni volta che si introduce un ALIAS per una tabella si dichiara una variabile di tipo tabella in cui viene copiato il contenuto della tabella di cui è ALIAS.

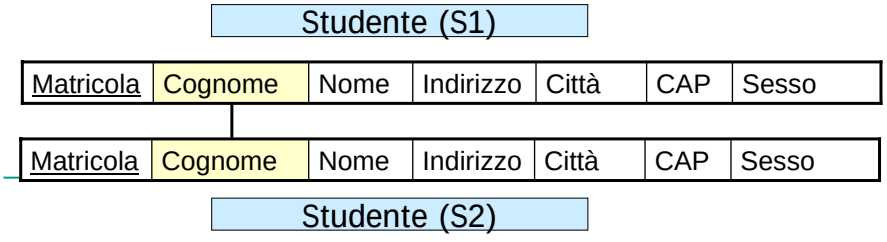
```
SELECT S1.Cognome
FROM Studente S1, Studente S2
WHERE S1.Cognome = S2.Cognome
AND S2.Città = 'Padova';
```

- In questo caso (quando una tabella compare più volte) l'ALIAS viene considerato come una nuova variabile.

Esempio 3

- Trovare il cognome di tutti gli studenti che hanno lo stesso cognome di uno studente che abita a Padova.

```
SELECT S1.Cognome
FROM Studente S1, Studente S2
WHERE S1.Cognome = S2.Cognome
AND S2.Città = 'Padova';
```



Interrogazioni di tipo insiemistico

- SQL mette a disposizione anche degli operatori insiemistici.
- Gli operatori insiemistici si possono utilizzare solo al livello più esterno di una query, operando sul risultato di SELECT.
- Gli operatori disponibili sono:
  - UNION
  - INTERSECT
  - EXCEPT (MINUS)

  e assumono come default di eseguire sempre una eliminazione dei duplicati (a meno dell'uso di ALL).

```
SelectSQL { < UNION | INTERSECT | EXCEPT >
                [ALL] SelectSQL }
```

Esempio 4

- Determinare i cognomi e le città degli studenti.

```
SELECT Cognome
FROM Studente
   UNION
SELECT Città
FROM Studente;
```

Esempio 4: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M
VR0004	Verona	Mario	Via K	Padova	52100	M
VR0005	Bianchi	Luca	Via Z	Verona	37135	M

Risultato

Cognome
Rossi
Neri
Verdi
Verona
Bianchi
Padova

I duplicati vengono eliminati

Esempio 5

- Determinare i cognomi e le città degli studenti maschi mantenendo i duplicati.

```
SELECT Cognome
FROM Studente
WHERE Sesso = `M'
   UNION ALL
SELECT Città
FROM Studente
WHERE Sesso = `M';
```

Esempio 5: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M
VR0004	Verona	Mario	Via K	Padova	52100	M
VR0005	Bianchi	Luca	Via Z	Verona	37135	M

Risultato

Cognome
Rossi
Verdi
Verona
Bianchi
Verona
Verona
Padova
Verona

I duplicati NON vengono eliminati

Esempio 6

- Determinare i cognomi degli studenti che sono anche città.

```
SELECT Cognome
FROM Studente
  INTERSECT
SELECT Città
FROM Studente;
```

Esempio 6: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M
VR0004	Verona	Mario	Via K	Padova	52100	M
VR0005	Bianchi	Luca	Via Z	Verona	37135	M

Risultato

Cognome
Verona

Esempio 7

- Determinare i cognomi degli studenti che non sono anche città.

```
SELECT Cognome
FROM Studente
  EXCEPT
SELECT Città
FROM Studente;
```

Esempio 7: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M
VR0004	Verona	Mario	Via K	Padova	52100	M
VR0005	Bianchi	Luca	Via Z	Verona	37135	M

Risultato

Cognome
Rossi
Neri
Verdi
Bianchi

Interrogazioni nidificate

- SQL ammette il confronto di un valore (ottenuto come risultato di una espressione valutata sulla singola riga) con il risultato dell'esecuzione di una interrogazione SQL.
- L'interrogazione che viene usata nel confronto viene definita direttamente nel predicato interno alla clausola WHERE.
- Confrontiamo un attributo con il risultato di una interrogazione.
  - PROBLEMA di disomogeneità: confronto tra un valore e un insieme di valori (risultato dell'interrogazione).
  - SOLUZIONE: uso delle parole chiave ALL ed ANY per estendere i normali operatori di confronto (=, >, …).

ANY ed ALL

- La parola chiave ANY specifica che la tupla corrente T soddisfa la condizione se risulta vero il confronto (con l'operatore specificato) tra il valore dell'attributo su T ed ALMENO UNO dei valori restituiti dall'interrogazione.
- La parola chiave ALL specifica che T soddisfa la condizione solo se TUTTI i valori restituiti dall'interrogazione nidificata rendono vero il confronto.
- NB: la sintassi richiede che lo schema della tabella restituita dall'interrogazione nidificata sia costituito da un solo attributo e sia compatibile in tipo con l'attributo con cui avviene il confronto.

Esempio 8

- Visualizzare il cognome dei docente che guadagnano meno dello stipendio medio

```
SELECT Cognome
FROM Docente
WHERE Stipendio < (SELECT AVG(Stipendio)
                   FROM Docente);
```

Confronto tra un valore e un valore: basta <

Docente

Matricola	Nome	Telefono	Stipendio
DIN001	A. A.	045 …	2500
DIN002	B. B.	045 …	1900
DIN003	C. C.	045 …	3000

Risultato 1

AVG(Stipendio)
2466

Risultato

Nome
B.B.

Esempio 9

- Trovare il cognome di tutti gli studenti di Verona che hanno lo stesso cognome di uno studente che abita a Padova.

```
SELECT Cognome
FROM Studente
WHERE Città = 'Verona'
AND Cognome = ANY (SELECT Cognome
                   FROM Studente
                   WHERE Città = 'Padova') ;
```

Confronto tra un valore e un insieme: serve ANY

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0004	Bianchi	Mario	Via K	Padova	52100	M
VR0005	Bianchi	Luca	Via Z	Verona	37135	M
VR0006	Rossi	Paolo	Via W	Verona	37132	M
VR0007	Rosa	Maria	Via J	Padova	52020	F

Risultato 1

Cognome
Bianchi
Rosa

Risultato

Cognome
Bianchi

Esempio 10

- Visualizzare l'elenco degli studenti che hanno almeno un voto maggiore o uguale del voto medio di ogni studente.

```
SELECT DISTINCT Cognome, Voto
FROM Studente S, Esame E
WHERE S.Matricola = E.Studente
 AND Voto >= ALL (SELECT AVG(Voto)
                  FROM Esame
                  GROUP BY Studente)
```

Confronto tra un valore e un insieme: serve ALL

Esempio 10: risultato

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0003	30

Risultato 1

AVG(Voto)
28
28
30

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0003	30

Risultato

Cognome	Voto
Verdi	30

Esempio 11

- Visualizzare il cognome degli studenti che hanno preso almeno un 30

```
SELECT Cognome
FROM Studente
WHERE Matricola = ANY (SELECT Studente
                       FROM Esame
                       WHERE Voto=30)
```

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0003	30

Risultato 1

Studente
VR0003

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Risultato

Cognome
Verdi

Interrogazioni nidificate

- Nelle interrogazioni nidificate viste finora si assume che l'interrogazione nidificata venga eseguita prima di analizzare le tuple dell'interrogazione esterna.
- Se l'interrogazione nidificata fa riferimento al contesto dell'interrogazione che la racchiude questo NON è possibile.
  - Interrogazioni nidificate complesse

Interrogazioni nidificate complesse

- L'interrogazione nidificata fa riferimento al contesto dell'interrogazione che la racchiude
  - tramite una variabile definita nell'ambito della query esterna ed usata nell'ambito della query interna
    - Tale legame si dice "PASSAGGIO DI BINDING"
- In questo caso non vale più l'interpretazione semplice data precedentemente alle interrogazioni nidificate.
  - Vale a dire l'interrogazione nidificata DEVE essere valutata per ogni tupla dell'interrogazione esterna che si sta valutando.

Interrogazioni nidificate complesse: esecuzione

1. Viene costruito il prodotto cartesiano delle tabelle
2. Le condizione che appaiono nella clausola WHERE vengono applicate a ciascuna riga del prodotto cartesiano
3. Vengono mantenute solo le righe per cui la condizione viene valutata vera
4. L'interrogazione nidificata (che compare all'interno di un predicato) viene valutata separatamente per ogni riga prodotta dalla valutazione dell'interrogazione più esterna.

EXISTS

- L'operatore logico EXISTS ammette come parametro un'interrogazione nidificata e restituisce il valore vero solo se l'interrogazione fornisce un risultato non vuoto.
- L'operatore logico EXISTS può essere usato in modo significativo solo quando si ha un passaggio di binding tra l'interrogazione esterna e quella nidificata argomento dell'operatore.

Esempio 12

- Visualizzare i dati degli studenti che hanno degli omonimi (stesso nome e cognome, diversa matricola).

```
SELECT *
FROM Studente S
WHERE EXISTS (SELECT *
                FROM Studente S1
                WHERE S.Nome = S1.Nome
                AND S.Cognome = S1.Cognome
                AND S.Matricola <> S1.Matricola )
```

Esempio 13

- Visualizzare la matricola e il cognome degli studenti che hanno superato più di un esame.

```
SELECT Matricola, Cognome
FROM Studente
WHERE EXISTS (SELECT COUNT(*)
                FROM Esame
                 WHERE Studente = Matricola
                 HAVING COUNT(*)>1)
```

Esempio 13: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0002	25

COUNT(*)
1
2
0

Esempio 13: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0002	25

Risultato

Matricola	Cognome
VR0002	Neri

NOT EXISTS

- L'operatore logico NOT EXISTS ammette come parametro un'interrogazione nidificata e restituisce il valore vero solo se l'interrogazione fornisce un risultato vuoto.

Esempio 14

- Visualizzare matricola e cognome degli studenti che non hanno superato alcun esame.

```
SELECT Matricola, Nome
FROM Studente
WHERE NOT EXISTS (SELECT *
                  FROM Esame
                  WHERE Studente=Matricola)
```

Esempio 14: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0002	25

Esempio 14: risultato

Studente

Matricola	Cognome	Nome	Indirizzo	Città	CAP	Sesso
VR0001	Rossi	Marco	Via X	Verona	37129	M
VR0002	Neri	Maria	Via W	Verona	37132	F
VR0003	Verdi	Paolo	Via Y	Verona	37121	M

Esame

Codice_corso	Studente	Voto
INF02	VR0001	28
INF02	VR0002	28
INF01	VR0002	25

Risultato

Matricola	Cognome
VR0003	Verdi

Viste SQL

- Le viste sono tabelle “virtuali” il cui contenuto dipende dal contenuto delle altre tabelle della base di dati.
- In SQL le viste vengono definite associando un nome ed una lista di attributi al risultato dell’esecuzione di un’interrogazione.
- Nell’interrogazione che definisce la vista possono comparire anche altre viste. SQL non ammette però:
  - dipendenze ricorsive
  - dipendenze immediate (definire una vista in termini di se stessa)
  - dipendenze transitive (V1 definita usando V2, V2 usando V3, …, Vn usando V1)

Viste SQL (2)

- Si definisce una vista usando il comando:

  CREATE VIEW NomeVista [(ListaAttributi)]

  AS SELECTSQL

  [WITH [LOCAL|CASCADED] CHECK OPTION]

- L’interrogazione SQL deve restituire un insieme di attributi pari a quello contenuto nello schema e l’ordine della target list deve corrispondere all’ordine degli attributi dello schema della vista.

- SQL permette che una vista sia aggiornabile solo quando una sola tupla di ciascuna tabella di base corrisponde a una tupla della vista.

Viste SQL (3)

```
CREATE VIEW NomeVista [(ListaAttributi)]
AS SELECTSQL
[WITH [LOCAL|CASCADED] CHECK OPTION]
```

- La clausola CHECK OPTION può essere utilizzata solo nel contesto di viste aggiornabili.
- La clausola CHECK OPTION specifica che possono essere ammessi aggiornamenti solo sulle righe della vista e dopo gli aggiornamenti le righe devono continuare ad appartenere alla vista.
  Esempio: assegnare ad un attributo della lista un valore che rende falso uno dei predicati di selezione.

Viste SQL (4)

```
CREATE VIEW NomeVista [(ListaAttributi)]
AS SELECTSQL
[WITH [LOCAL|CASCADED] CHECK OPTION]
```

- L'opzione LOCAL o CASCADED specifica, nel caso una vista sia definita in termini di altre viste, se il controllo del fatto che le righe vengano rimosse dalla vista
  - debba essere effettuato solo all'ultimo livello (si controlla che la modifica non faccia violare la condizione della vista più esterna)
  - o se deve essere propagato a tutti i livelli di definizione (si controlla che le righe su cui si apportano le modifiche non scompaiano dalla vista a causa della violazione di una qualsiasi delle condizioni delle viste coinvolte)
- L'opzione di default è CASCADED

Esempio 1

- Definire la vista che contiene la matricola e il cognome dei docenti che tengono un corso con più di 100 studenti

```
CREATE VIEW Docenti100Studenti(Matricola,Cognome)
AS SELECT Docente.Matricola, Docente.Cognome
FROM Docente, Docenza
WHERE Docenza.Numero_studenti > 100
AND Docente.Matricola= Docenza.Docente;
```

Esempio 2

- Definire la vista che contiene il nome dei corsi tenuti da ogni docente

```
CREATE VIEW
DocenteCorso(CognomeDocente,NomeInsegnamento)
AS SELECT D.Cognome, I.Nome_ins
FROM Docente D, Insegnamento I, Docenza Dz
WHERE D.Matricola = Dz.Docente
AND Dz.Codice_ins = I.Codice
```

Esempio 3

- Si vuole determinare qual è il docente che tiene corsi con la massima somma di crediti

```
CREATE VIEW CreditiDocente(CognomeDocente,TotCreditiCorsi)
AS SELECT D.Cognome, SUM(I.Numero_crediti)
FROM Docente D, Insegnamento I, Docenza Dz
WHERE D.Matricola = Dz.Docente
AND I.Codice = Dz.Codice_ins
GROUP BY D.Cognome

SELECT Nome, TotCreditiCorsi
FROM CreditiDocente
WHERE TotCreditiCorsi = (SELECT MAX(ToTCreditiCorsi)
                         FROM CreditiDocente)
```