Metodologie informatiche per le discipline umanistiche

SQL Database

Coordinatore: Prof. Paolo Nesi
Docenti: Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
Dr. Emanuele Bellini
ebellini@dsi.unifi.it

1

SQL

- SQL = Structured Query Language – Linguaggio di Interrogazione Strutturato
- Contiene sia istruzioni di DDL (Data Definition Language) sia di DML (Data Manipulation Language)

2

SQL - storia

- prima proposta SEQUEL (1974);
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 ca. "standard di fatto"
- standard (1986, poi 1989 e infine 1992, 1999)
- ne esistono diverse implementazioni
  - **lo standard è recepito solo in parte** (!!)

SQL interattivo vs embedded

Pur essendo considerato un linguaggio avanzato di database, in realtà SQL richiede la presenza di un linguaggio ospite come C, Visual Basic, Java, Cobol, Fortran, PhP, etc.. poichè al suo interno non prevede strutture di controllo.

Può essere usato in modo:

- Interattivo, digitando direttamente dal prompt SQL o dalle interfacce dei DBMS il comando che viene immediatamente elaborato, interpretato ed eseguito.

- In modalità embedded, in cui le istruzioni SQL vengono inserite all'interno di un programma.

SQL

SQL si definisce non procedurale in quanto l'utente deve solo specificare quali dati desidera elaborare e non la modalità con cui devono essere individuati.

Infatti i comandi SQL, utilizzati per definire, visualizzare e aggiornare dati specificano solamente l'operazione da eseguire sui dati, mentre viene lasciata al sistema la definizione della procedura ottimale per eseguire l'operazione richiesta

SQL

SQL considera il database come un insieme di tabelle in relazioni tra loro. Più precisamente un database SQL è una directory che contiene i seguenti tipi di file:

- Tabelle: che sono la struttra dle database

- Viste: o tabelle virtuali che consentono di costruire delle strutture formate da dati estratti dalle righe e colonne di una o più tabelle di base, che vengono aggiornate automaticamente al variare delle tabelle di base

- Sinomini o nomi alternativi per le tabelle

- Indici che consentono accessi più rapidi

Tipi di dato

Carattere: singoli caratteri o stringhe, anche di lunghezza variabile

Bit: singoli booleani o stringhe

Numerici, esatti(Integer) e approssimati (Float/Double)

Data, ora, intervalli di tempo

Boolean (Vero/Falso)

BLOB, CLOB (binary/character large object): per grandi immagini e testi



Tipi di dato – DBMS Access

Testo: singoli caratteri o stringhe, anche di lunghezza variabile

Memo: testi lunghi

Numerico

Data/ora

Valuta

Contatore

Si/No: in pratica valori booleani



QUERY

Le query permettono di interrogare una o più tabelle estraendo anche attraverso filtri, una selezione dei campi, presentandoli in forma schematica (struttura di vista), definita dall'utilizzatore.

Esistono due tipi di query:

Query di vista: utilizzate nelle interrogazioni dei database per selezionare record che soddisfano a determinate condizioni, per correlare campi di record di database diversi, per filtrare dati necessari ad altre applicazioni, come stampa di prospetti, di etichette di schede.

Query di aggiornamento utilizzate appunto per aggiornare I dati di un archivio in funzione delle condizioni specificate nel file di query, consentendo di aggiungere, sostituire o cancellare I dati stessi.

Creare un DB/ attivare un DB

Prima di iniziare a creare le tabelle che formano un database occorre avere a disposizione un database atto a contenrle, cioè na direcotry su disco in cui verranno memorizzate tutte le informazioni relative al database stesso.

Il comando è:
create database nome database;

Se invece il database è già creato basterà attivarlo con il comando:

start database nome database

Aggiungere una tabella

- Istruzione **CREATE TABLE**:
  - definisce uno schema di relazione e ne crea un'istanza vuota
  - permette di specificare attributi, domini e vincoli

Esempio: CREATE TABLE

CREATE TABLE nome tabella
(nome colonna1 tipo1
[, nome colonna2 tipo2]..);

Esempio

```
CREATE TABLE Impiegato(
    Matricola CHAR(6) PRIMARY KEY,
    Nome CHAR(20) NOT NULL,
    Cognome CHAR(20) NOT NULL,
    Stipendio NUMERIC(9) DEFAULT 0,
);
```

Eliminare una tabella

Questo comando cancella tutti gli indici e le viste associate alla tabella.

DROPE TABLE nome tabella

Aggiunta di una colonna in una tabella

Se ad un certo punto ci si accorge di aver tralasciato in fase di analisi, un campo significativo, è possibile modificare la struttura di una tabella, aggiungendo uno o più colonne con il comando:

ALTER TABLE nome tabella
ADD (nome colonna1 tipo1[,nomecol2 tipo2]..);

Inserimento di dati in una tabella

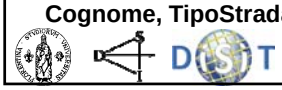
Una volta definita la struttura di una tabella si passa all'operazione di inseriemento dei dati sotto forma di nuove righe con il comando insert nel formato:

INSERT INTO nome tabella
VALUES (elenco valori relativi alle colonne);

Esempio

Insert into ANAGRAFE values ("Emanuele", "Bellini", "Via", "Pinco Pallino");

In questo caso si prevde che la tabella ANAGRAFE sia fatta dei campi: Nome, Cognome, TipoStrada, NomeStrada nella sequenza riportata



Modifica di dati in una tabella

UPDATE nome tabella
SET nome colonna=espressione
[,nome colonna2=espressione2]
[WHERE condizione di ricerca]

Esempio

update ANAGRAFE
Set nome="Luca"
Where nome="Emanuele" and cognome="Bellini"

Cancellare dati in una tabella

DELETE FROM nome tabella
[WHERE condizione di ricerca]

Esempio

Delete from ANAGRAFE
Where nome="Emanuele" and cognome="Bellini"

NB: La cancellazione non è recuperabile (salvo usare in backup del DB), quindi prima di lanciare il comando verificare bene che le condizioni della "where" soddisfino l criteri di selezione. Lo si può verificare usando una Select con gli stessi criteri e verificare che l record presentati siano quelli che si vuoel eliminare.

17

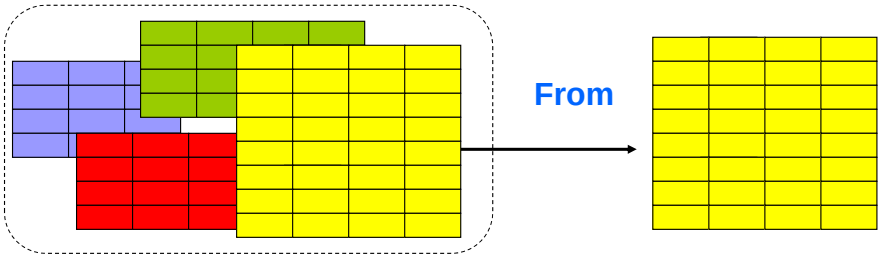
Select

SELECT [all/distinct] lista colonne/*
FROM nome tabella/vista
[WHERE condizione di ricerca]
[GROUP BY nome colonna [,nome colonna..]]
[HAVING condizione di ricerca]
[ORDER BY nome colonna/intero [asc/desc]
[,nome colonna/intero [asc/desc]…]

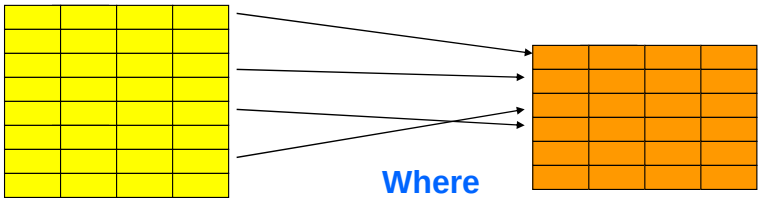
L'esecutore, quando incontra un comando select, esamina le clausole in esso contenute in un ordine che non rispetta quell con cui le clausole stesse compaiono nel comando. Infatti l'ordine di elaborazione segue una logica che, partendo dalle tabelle iniziali (clausola from), ad ogni passo (esecuzione delle clausole) Riduce o trasforma le tabelle precedenti in altre tabelle intermedie che rappresentano un punto di partenza per la clausola successiva.

18

Database



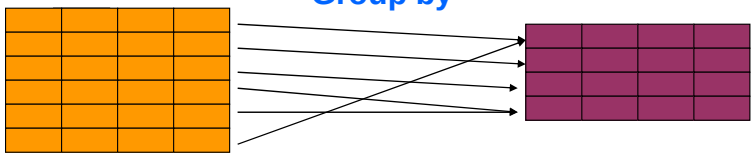
La clausola **from** definise la tabella da cui accedere per l'estrazione dei dati



La clausola **where** definisce la tabella intermedia formata solo da quelle righe che soddisfano la condizione imposta.

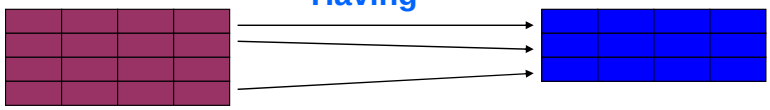
19

Group by



La clausola **group by** definisce una tabella intermedia in cui sono raggruppate le righe in base ai valori uguali di certe colonne

Having



La clausola **having** costruisce la tabella intermedia formata da solo quelle righe che soddisfano la condizione imposta

Select

La clausola **Select** costruisce la tabella con le sole colonne

20

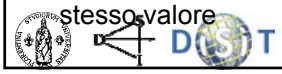
La clausola where

Where condizione di ricerca

Dove condizione di ricerca rappresenta una qualsiasi espressione logica

```
Select Cognome, voto
From ANAGRAFE, DIPLOMAT
Where ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA
```

Questo comando produrrà la tabella contenene in ogni rga il cognome dello studente ed il voto da lui sostenuto. In pratica la clausola where, partendo dalla tabella di combinazione Prodotta dalla clausola from, ha selezionato solo le righe n cui I campi MATRICOLA Presentavano lo stesso valore



Operatori

Matematici: +, - , *, /

Relazione: <, >, =, <> (diverso da), <=(minore o uguale a) , >=(maggiore o uguale a)

Logici: and (congiunzione), or (disgiunzione), not (negazione)

Stringa: + concatenamento
se A="a " e B="b" allora A+B fornirà "a b"

Operatori SQL - Between

esp Between esp1 and esp 2

Equivale all'espressione esp compresto tra esp1 e esp2

Select cognome, nome from ANAGRAFE, DIPLOMAT
Where (ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA)
and (DIPLOMAT.VOTO between 41 and 48);

Forinsce l'elenco dei diplomati con voto compreso tra il 42 e il 48

Operatori SQL - In

esp in (val1, val2,..)

Equivale all'espressione esp assume uno dei valori riportati tra parentesi

Select cognome, nome from ANAGRAFE, DIPLOMAT
Where (ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA)
And
(ANAGRAFE.CITTA IN ("Milano", "Firenze"));

Forinsce l'elenco dei diplomati abitanti a Firenze e Milano

Operatori SQL - like

like

Consente delle maschere di ricerca utilizzando I caratteri jolly **%** e _
La stringa di ricerca che contiene caratteri jolly va racchiusa tra aplici e non Tra virgolette

Select cognome, nome from ANAGRAFE, DIPLOMAT
Where (ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA)
And
(ANAGRAFE.COGNOME like 'A%'),
Fornisce l'elenco ti tutti I diplomati che iniziano con la lettera A

25

Operatori SQL - like

like

Select cognome, nome from ANAGRAFE, DIPLOMAT
Where (ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA)
And
(ANAGRAFE.COGNOME like '_I%'),

Fornisce l'elenco ti tutti i diplomati la cui seconda lettera è la I

26

Clausola having

Having condizione di ricerca

Assomiglia alla clausola **where**, am mentre la **where** elabora la tabella intermedia prodotta dalla **from**, la **having** elabora quella restituita dalla **group by**

Clausola order by

Order by nome colonna asc/desc

Select cognome, nome from ANAGRAFE, DIPLOMAT
Where (ANAGRAFE.MATRICOLA=DIPLOMAT.MATRICOLA)
Order by Voto desc;

Restituisce l'elenco dei diplomati, ordinato rispetto al voto

Funzioni

Sono funzioni aggregate sql che possono essere utilizzate nelle espressioni contenute nel comando **select**

Avg([all/distinct] nome colonna)

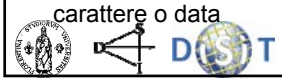
Calcola la media aritmetica dei valori di una colonna numerica nelle righe selezionate

Count([*/distinc] nome colonna)

Conta il numero di righe selezionate in una query

Max([all/distinc] nome colonna)

Trova il valore massimo in una colonna specificata, colonna di tipo numerico, carattere o data



Funzioni

Min([all/distinc] nome colonna)

Trova il valore minimo in una colonna specificata del tipo prima indicato

Sum([all/distinc] nome colonna)

Somma I valori di una colonna numerica nelle righe selezionate

Vincoli intra-relazionali

- **NON NULLO**
- **UNICO**
- **CHIAVE PRIMARIA** (una sola, implica UNICO e NON NULLO)

Vincoli interrelazionali

- INTEGRITA‘ REFERENZIALE: i valori presenti per un attributo in una certa tabella (slave) devono essere già presenti in un‘altra tabella di riferimento (master)
- E' possibile definire politiche di reazione alla violazione (azioni „compensative“)
- In assenza di politiche specifiche l‘operazione (cancellazione o modifica) che causerebbe la violazione non viene consentita

Esempio

- Definiamo le tabelle relative agli schemi
- Studente(Matricola, Nome, Cognome, DataNascita)
- Corso(Codice, Titolo, Docente)
- Esame(Studente, Corso, Data, Voto)
- Quali sono i vincoli che devono essere soddisfatti ?

Esempio

```
select nome, età,
   reddito
   from persone
   where eta < 30
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

Nome	Età	Reddito	
Andrea	27	21	eta < 30 ? si
Aldo	25	15	eta < 30 ? si
Maria	55	42	eta < 30 ? no
Anna	50	35	eta < 30 ? no
Filippo	26	30	eta < 30 ? si
Luigi	50	40	eta < 30 ? no
Franco	60	20	eta < 30 ? no
Olga	30	41	eta < 30 ? no
Sergio	85	35	eta < 30 ? no
Luisa	75	87	eta < 30 ? no

```
select nome,
età, reddito
from persone
where eta < 30
```

Uso espressioni aritmetiche

```
SQL> SELECT ename, sal, sal+300
     FROM   emp;

ENAME            SAL   exp
---------- --------- ---------
KING            5000      5300
BLAKE           2850      3150
CLARK           2450      2750
JONES           2975      3275
MARTIN          1250      1550
ALLEN           1600      1900
...
14 rows selected.
```

Precedenza operatori

```
SQL> SELECT ename, sal, 12*sal+100
  2  FROM   emp;

ENAME            SAL exp
---------- --------- ----------
KING            5000      60100
BLAKE           2850      34300
CLARK           2450      29500
JONES           2975      35800
MARTIN          1250      15100
ALLEN           1600      19300
...
14 rows selected.
```



Uso delle parentesi

```
SQL> SELECT ename, sal, 12*(sal+100)
  2  FROM   emp;

ENAME            SAL exp
---------- --------- -----------
KING            5000       61200
BLAKE           2850       35400
CLARK           2450       30600
JONES           2975       36900
MARTIN          1250       16200
...
14 rows selected.
```



Alias delle colonne

- Ridenominare il nome di una colonna
  - Implementa l'operatore ρ (Ridenominazione) dell'algebra relazionale
- L'alias deve seguire immediatamente il nome di una colonna (SENZA VIRGOLA)
  - può essere usata opzionalmente la parola chiave AS tra il nome della colonna e l'alias
- Richiede doppio apice se l'alias ha degli spazi



Esempio

```
SQL> SELECT ename AS name, sal salary
  2  FROM   emp;
```

```
NAME            SALARY
------------ ---------
...
```

```
SQL> SELECT ename "Name",
  2         sal*12 "Annual Salary"
  3  FROM   emp;
```

```
Name         Annual Salary
------------ ------------
...
```



SELECT abbreviazioni

- select nome, reddito

from persone

where eta < 30

```
select p.nome as nome,
                    p.reddito as reddito
from persone p
where p.eta < 30
```

SELECT con abbreviazioni

- ```
  select nome, età, reddito
  from persone
  where eta < 30
  ```

- ```
  select *
  from persone
  where eta < 30
  ```

Selezione con espressioni

```
select Reddito/2 as redditoSemestrale
from Persone
where Nome = 'Luigi'
```

Selezione con condizione complessa

```
select *
from persone
where reddito > 25
      and (eta < 30 or eta > 60)
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

RISULTATO INTERROGAZIONE PRECEDENTE ??

RICORDIAMO:
LE CONDIZIONI VENGONO VERIFICATE RIGA PER RIGA

Operatore LIKE

- LIKE è usato per effettuare ricerche *wildcard* di una stringa di valori.
- Le condizioni di ricerca possono contenere sia letterali, caratteri o numeri.
  - % denota zero o più caratteri.
  - _ denota un carattere.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE 'S%';
```

Predicato BETWEEN

- Espr1 [NOT] BETWEEN Espr2 AND Espr3
- Equivale a
  - [NOT] Espr2 ≤Espr1 AND Espr1≤Espr3

Uso operatore IN

- E' usato per selezionare righe che hanno un attributo che assume valori contenuti in una lista.

```
SQL> SELECT   empno, ename, sal, mgr
  2  FROM     emp
  3  WHERE    mgr IN (7902, 7566, 7788);
```

EMPNO	ENAME	SAL	MGR
7902	FORD	3000	7566
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788

Gestione valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

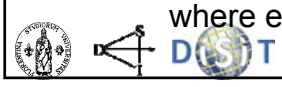
Gestione valori nulli

Impiegati

Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

- Gli impiegati la cui età è o potrebbe essere maggiore di 40

```
select *
from impiegati
where eta > 40 or eta is null
```



Gestione valori nulli

- Per verificare il valore nullo non si usano i normali operatori di confronto (= uguale e <> diverso)
- Operatori speciali
  - IS NULL e IS NOT NULL

WHERE stipendio <> NULL

WHERE stipendio IS NOT NULL

Selezione, proiezione e join

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:
  - selezioni, proiezioni, ridenominazioni (per modificare temporaneamente il nome di un attributo o di una tabella)
- con più relazioni nella FROM si realizzano prodotti cartesiani e join

SQL e algebra relazionale

- R1(A1,A2) R2(A3,A4)

```
SELECT R1.A1, R2.A4
FROM   R1, R2
WHERE  R1.A2 = R2.A3
```

- prodotto cartesiano (FROM)
- selezione (WHERE)
- proiezione (SELECT)



SQL e algebra relazionale

- R1(A1,A2) R2(A3,A4)

```
SELECT R1.A1, R2.A4
FROM   R1, R2
WHERE  R1.A2 = R2.A3
```

$$\pi_{A1,A4} (\sigma_{A2=A3} (R1 \text{ JOIN } R2))$$



- possono essere necessarie ridenominazioni
  - nel prodotto cartesiano
  - nella target list

```
SELECT X.A1 AS B1, ...
FROM   R1 X, R2 Y, R1 Z
WHERE  X.A2 = Y.A3 AND ...
```

- Self JOIN su R1

```
SELECT X.A1 AS B1, Y.A4 AS B2
FROM   R1 X, R2 Y, R1 Z
WHERE  X.A2 = Y.A3 AND Y.A4 = Z.A1
```

$$\delta_{B1,B2 \leftarrow A1,A4} (\pi_{A1,A4} (\sigma_{A2 = A3 \text{ AND } A4 = C1} (R1 \text{ JOIN } R2 \text{ JOIN } \delta_{C1,C2 \leftarrow A1,A2} (R1))))$$

SQL: esecuzione delle interrogazioni

- Le espressioni SQL sono dichiarative = descrivono il risultato da raggiungere
- In pratica, i DBMS eseguono le operazioni in modo efficiente, ad esempio:
  - eseguono le selezioni al più presto
  - se possibile, eseguono join e non prodotti cartesiani

SQL: specifca delle interrogazioni

- La capacità dei DBMS di "ottimizzare" le interrogazioni, rende (di solito) non necessario preoccuparsi dell'efficienza quando si specifica un'interrogazione
- È perciò più importante preoccuparsi della chiarezza (anche perché così è più difficile sbagliare …)

Proiezione, attenzione

- **cognome e filiale di tutti gli impiegati**

Matricola	Cognome	Filiale	Stipendio
7309	Neri	Napoli	55
5998	Neri	Milano	64
9553	Rossi	Roma	44
5698	Rossi	Roma	64

select
cognome, filiale
from impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma
Rossi	Roma

select **distinct**
cognome, filiale
from impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

Il risultato non è di tipo insiemistico, cioè può contenere duplicati
Per eliminare i duplicati si usa **distinct**

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

61

Selezione, proiezione e join

- I padri di persone che guadagnano più di venti milioni

```
select distinct paternità.padre
from persone, paternita
where persone.reddito > 20
```

Prodotto cartesiano

Prodotto cartesiano di due tabelle =
tutte le possibili combinazioni di tuple

62

Persone

Paternità

Padre	Figlio	Nome	Età	Reddito
Sergio	Franco	Andrea	27	21
Sergio	Franco	Aldo	25	15
Sergio	Franco	Maria	55	42
Sergio	Franco	Anna	50	35
Sergio	Franco	Filippo	26	30
Sergio	Franco	Luigi	50	40
Sergio	Franco	Franco	60	20
Sergio	Franco	Olga	30	41
Sergio	Franco	Sergio	85	35
Sergio	Franco	Luisa	75	87
Luigi	Olga	Andrea	27	21
Luigi	Olga	Aldo	25	15
…	…	…	…	…

63

Persone

Paternità

Padre	Figlio	Nome	Età	Reddito
Sergio	Franco	Andrea	27	21
Sergio	Franco	Aldo	25	15
Sergio	Franco	Maria	55	42
Sergio	Franco	Anna	50	35
Sergio	Franco	Filippo	26	30
Sergio	Franco	Luigi	50	40
Sergio	Franco	Franco	60	20
Sergio	Franco	Olga	30	41
Sergio	Franco	Sergio	85	35
Sergio	Franco	Luisa	75	87
Luigi	Olga	Andrea	27	21
Luigi	Olga	Olga	30	41
…	…	…	…	…

64

Condizioni di join

```
select distinct paternità.padre
from persone, paternita
where paternità.figlio = persone.nome
and reddito > 20
```

Join = sottoinsieme del prodotto cartesiano di due tabelle

```
select distinct padre
from persone, paternita
where figlio = nome and reddito > 20
```

Ogni nome di attributo compare in una sola tabella

Padre e madre di ogni persona

```
select paternita.figlio,padre, madre
from maternita, paternita
where paternita.figlio = maternita.figlio
```

Attenzione ai nomi di attributi che compaiono in più tabelle

Ridenominazione

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
select f.nome, f.reddito, p.reddito
from persone p, paternita, persone f
where p.nome = padre and
      figlio = f.nome and
      f.reddito > p.reddito
```

Ridenominazione: è più comodo usare abbreviazioni per i nomi delle tabella

Occorre distinguere occorrenze diverse della stessa tabella

SELECT, con ridenominazione del risultato

```
select figlio, f.reddito as reddito,
       p.reddito as redditoPadre
from persone p, paternita, persone f
where p.nome = padre and figlio = f.nome
      and f.reddito > p.reddito
```

Join esplicito

- Padre e madre di ogni persona

```
select paternita.figlio,padre, madre
from maternita, paternita
where paternita.figlio = maternita.figlio

select madre, paternita.figlio, padre
from maternita join paternita on
        paternita.figlio = maternita.figlio
```

- Non è sempre implementato

SELECT con join esplicito, sintassi

SELECT …
FROM Tabella { … JOIN Tabella ON CondDiJoin }, …
[WHERE AltraCondizione]

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
select f.nome, f.reddito, p.reddito
from persone p, paternita, persone f
where p.nome = padre and
      figlio = f.nome and
      f.reddito > p.reddito
```

```
select f.nome, f.reddito, p.reddito
from persone p join paternita on p.nome = padre
      join persone f on figlio = f.nome
where f.reddito > p.reddito
```

Ordinamento del risultato

- Nome e reddito delle persone con meno di trenta anni **in ordine alfabetico**

```
select nome, reddito
from persone
where eta < 30
order by nome
```

select nome, reddito
from persone
where eta < 30

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

select nome, reddito
from persone
where eta < 30
order by nome

Persone

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30

73

Metodologie informatiche per le discipline umanistiche

SQL Database

Coordinatore: Prof. Paolo Nesi
Docenti: Prof. Paolo Nesi
http://www.dsi.unifi.it/~nesi
Dr. Emanuele Bellini
ebellini@dsi.unifi.it

74