

Basi di dati

Capitolo 4: SQL: concetti base

1/125

SQL

- ora "nome proprio", originariamente "Structured Query Language"
- linguaggio con varie funzionalità:
 - contiene sia il DDL sia il DML
- ne esistono varie versioni
- vediamo gli aspetti essenziali, non i dettagli

2/125

SQL: "storia"

- prima proposta **SEQUEL** (1974);
- prime implementazioni in SQL/DS e Oracle (1981)
- dal 1983 circa "standard di fatto"
- standard (1986, poi 1989, 1992, 1999, 2003, 2006, 2008, 2011 ...)
 - recepito solo in parte !! (Vedi per un confronto <http://troels.arvin.dk/db/rdbms/>)

3/125

Definizione dei dati in SQL

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

4/125

CREATE TABLE, esempio

```
CREATE TABLE Impiegato(  
    Matricola CHAR(6) PRIMARY KEY,  
    Nome CHAR(20) NOT NULL,  
    Cognome CHAR(20) NOT NULL,  
    Dipart CHAR(15),  
    Stipendio NUMERIC(9) DEFAULT 0,  
  
    FOREIGN KEY(Dipart) REFERENCES  
        Dipartimento(NomeDip),  
    UNIQUE (Cognome, Nome)  
)
```

- DB2 vuole NOT NULL per la chiave primaria

5/125

Domini

- Domini elementari (predefiniti)
- Domini definiti dall'utente (semplici, ma riutilizzabili)

6/125

Domini elementari

- **Carattere**: singoli caratteri o stringhe, anche di lunghezza variabile
- **Numerici**, esatti e approssimati
- **Data, ora, intervalli di tempo**
- Introdotti in SQL:1999:
 - **Boolean**
 - **BLOB, CLOB** (binary/character large object): per grandi immagini e testi

7/125

Definizione di domini

- Istruzione **CREATE DOMAIN**:
 - definisce un dominio (semplice), utilizzabile in definizioni di relazioni, anche con vincoli e valori di default

8/125

CREATE DOMAIN, esempio

```
CREATE DOMAIN Voto
AS SMALLINT DEFAULT NULL
CHECK ( value >=18 AND value <= 30 )
```

- note:
 - Mimer OK
 - SQLServer, DB2 no

9/125

Vincoli intrarelazionali

- **NOT NULL**
- **UNIQUE** definisce chiavi
- **PRIMARY KEY**: chiave primaria
 - una sola, implica **NOT NULL**
 - DB2 non rispetta lo standard
- **CHECK**, vedremo più avanti

10/125

UNIQUE e PRIMARY KEY

- due forme:
 - nella definizione di un attributo, se forma da solo la chiave
 - come elemento separato

11/125

CREATE TABLE, esempio PRIMARY KEY nella definizione dell'attributo

```
CREATE TABLE Impiegato (
  Matricola CHAR(6) PRIMARY KEY,
  Nome CHAR(20) NOT NULL,
  Cognome CHAR(20) NOT NULL,
  Dipart CHAR(15),
  Stipendio NUMERIC(9) DEFAULT 0,
  FOREIGN KEY(Dipart) REFERENCES
    Dipartimento(NomeDip),
  UNIQUE (Cognome,Nome)
)
```

12/125

PRIMARY KEY, alternative

- 1) Matricola CHAR(6) PRIMARY KEY
- 2) Matricola CHAR(6),
...,
PRIMARY KEY (Matricola)

13/125

CREATE TABLE, esempio

Vincolo intrarelazionale UNIQUE

```
CREATE TABLE Impiegato(  
  Matricola CHAR(6) PRIMARY KEY,  
  Nome CHAR(20) NOT NULL,  
  Cognome CHAR(20) NOT NULL,  
  Dipart CHAR(15),  
  Stipendio NUMERIC(9) DEFAULT 0,  
  
  FOREIGN KEY(Dipart) REFERENCES  
    Dipartimento(NomeDip),  
  UNIQUE (Cognome, Nome)  
)
```

15/125

Vincoli interrelazionali

- CHECK, vedremo più avanti
- REFERENCES e FOREIGN KEY permettono di definire vincoli di integrità referenziale
- di nuovo, due sintassi
 - per singoli attributi
 - su più attributi
- E' possibile definire politiche di reazione alla violazione

17/125

CREATE TABLE, esempio

Due attributi come PRIMARY KEY

```
CREATE TABLE Impiegato(  
  Nome CHAR(20),  
  Cognome CHAR(20),  
  Dipart CHAR(15),  
  Stipendio NUMERIC(9) DEFAULT 0,  
  
  FOREIGN KEY(Dipart) REFERENCES  
    Dipartimento(NomeDip),  
  PRIMARY KEY (Cognome, Nome)  
)
```

14/125

Chiavi su più attributi, attenzione

Nome CHAR(20) NOT NULL,
Cognome CHAR(20) NOT NULL,
UNIQUE (Cognome, Nome),

Nome CHAR(20) NOT NULL UNIQUE,
Cognome CHAR(20) NOT NULL UNIQUE,

- Non è la stessa cosa!

16/125

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Vigili

Matricola	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

18/125

Infrazioni

Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto

Prov	Numero	Cognome	Nome
MI	39548K	Rossi	Mario
TO	E39548	Rossi	Mario
PR	839548	Neri	Luca

19/125

Modifiche degli schemi

ALTER DOMAIN
ALTER TABLE
DROP DOMAIN
DROP TABLE
...

21/125

DDL, in pratica

- In molti sistemi si utilizzano strumenti diversi dal codice SQL per definire lo schema della base di dati

23/125

CREATE TABLE, esempio

```
CREATE TABLE Infrazioni(  
  Codice CHAR(6) NOT NULL PRIMARY KEY,  
  Data DATE NOT NULL,  
  Vigile INTEGER NOT NULL  
    REFERENCES Vigili(Matricola),  
  Provincia CHAR(2),  
  Numero CHAR(6) ,  
  
  FOREIGN KEY(Provincia, Numero)  
    REFERENCES Auto(Provincia, Numero)  
)
```

20/125

Definizione degli indici

- è rilevante dal punto di vista delle prestazioni
- ma è a livello fisico e non logico
- in passato era importante perché in alcuni sistemi era l'unico mezzo per definire chiavi
- CREATE INDEX

22/125

SQL, operazioni sui dati

- interrogazione:
 - SELECT
- modifica:
 - INSERT, DELETE, UPDATE

24/125

Istruzione SELECT (versione base)

```
SELECT ListaAttributi  
FROM ListaTabelle  
[ WHERE Condizione ]
```

- clausola **SELECT** (chiamata *target list*)
- clausola **FROM**
- clausola **WHERE**

25/125

Selezione e proiezione

- Nome e reddito delle persone con meno di trenta anni

```
PROJNome, Reddito(SELEta<30(Persone))
```

```
SELECT nome, reddito  
FROM persone  
WHERE eta < 30
```

27/125

Selezione, senza proiezione

- Nome, età e reddito delle persone con meno di trenta anni

```
SELEta<30(Persone)
```

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

29/125

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

26/125

SELECT, abbreviazioni

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

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

28/125

SELECT, abbreviazioni

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

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

30/125

Proiezione, senza selezione

- Nome e reddito di tutte le persone

$PROJ_{Nome, Reddito}(Persone)$

```
select nome, reddito  
from persone
```

31/125

SELECT, abbreviazioni

- $R(A,B)$

```
select *  
from R
```

equivale (intuitivamente) a

```
select X.A as A, X.B as B  
from R X  
where true
```

32/125

Espressioni nella target list

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

33/125

Condizione complessa

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

34/125

Condizione "LIKE"

- Le persone che hanno un nome che inizia per 'A' e ha una 'd' come terza lettera

```
select *  
from persone  
where nome like 'A_d%'
```

35/125

Gestione dei 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

36/125

Gestione dei 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

SEL _{(Età > 40) OR (Età IS NULL)} (Impiegati)

37/125

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

SEL _{Età > 40 OR Età IS NULL} (Impiegati)

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

38/125

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

39/125

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			64

PROJ _{Cognome, Filiale} (Impiegati)

40/125

Proiezione, attenzione

- cognome e filiale di tutti gli impiegati

Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

PROJ _{Cognome, Filiale} (Impiegati)

41/125

```
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

42/125

Selezione, proiezione e join

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:
 - selezioni, proiezioni, ridenominazioni
- con più relazioni nella FROM si realizzano join (e prodotti cartesiani)

43/125

SQL e algebra relazionale

- $R1(A1,A2) \bowtie R2(A3,A4)$

```
select distinct R1.A1, R2.A4
from R1, R2
where R1.A2 = R2.A3
```

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

44/125

SQL e algebra relazionale, 2

- $R1(A1,A2) \bowtie R2(A3,A4)$

```
select R1.A1, R2.A4
from R1, R2
where R1.A2 = R2.A3
```

$PROJ_{A1,A4} (SEL_{A2=A3} (R1 \bowtie R2))$

45/125

- 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 ...
```

46/125

```
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
```

$REN_{B1,B2 \leftarrow A1,A4} ($
 $PROJ_{A1,A4} (SEL_{A2=A3 \text{ AND } A4=C1} ($
 $R1 \bowtie R2$
 $JOIN REN_{C1,C2 \leftarrow A1,A2} (R1))))$

47/125

SQL: esecuzione delle interrogazioni

- Le espressioni SQL sono dichiarative e noi ne stiamo vedendo la semantica
- 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

48/125

SQL: specifica 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 ...)

49/125

Selezione, proiezione e join

- I padri di persone che guadagnano più di 20

```
PROJPadre(paternita
  JOINFiglio = Nome
    SELReddito > 20(persone))
```

51/125

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

```
PROJNome, Reddito, RP(SELReddito > RP
  (RENNP, EP, RP ← Nome, Eta, Reddito(persone)
    JOINNP = Padre
      (paternita JOINFiglio = Nome persone)))
```

53/125

Maternità

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

Paternità

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

50/125

Selezione, proiezione e join

- I padri di persone che guadagnano più di 20

```
PROJPadre(paternita
  JOINFiglio = Nome
    SELReddito > 20(persone))
```

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

52/125

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

```
PROJNome, Reddito, RP(SELReddito > RP
  (RENNP, EP, RP ← Nome, Eta, Reddito(persone)
    JOINNP = Padre
      (paternita JOINFiglio = Nome persone)))
```

```
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
```

54/125

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
```

55/125

Join esplicito

- Padre e madre di ogni persona

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

56/125

Join esplicito

- Padre e madre di ogni persona

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

57/125

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

58/125

SELECT con join esplicito, sintassi

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

59/125

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

JOIN ESPLICITO

JOIN e CONDIZIONI nella clausola WHERE

60/125

- 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 join paternita on p.nome = padre)
join persone f on figlio = f.nome
where f.reddito > p.reddito
```

61/125

Ulteriore estensione: join naturale (meno diffuso)

PROJ_{Figlio,Padre,Madre}(paternita
JOIN_{Figlio = Nome} REN_{Nome=Figlio}(maternita))

paternita JOIN maternita

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

```
select madre, figlio, padre
from maternita natural join paternita
```

mimer OK DB2 no

63/125

Outer join

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

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

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

- Che cosa produce ?

65/125

- 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 join paternita on p.nome = padre)
join persone f on figlio = f.nome
where f.reddito > p.reddito
```

```
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
```

62/125

Join esterno: "outer join"

- Padre e, se nota, madre di ogni persona

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

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

- **outer** e' opzionale

64/125

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
```

66/125

```
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

67/125

Unione, intersezione e differenza

- La **select** da sola non permette di fare unioni; serve un costrutto esplicito:

```
select ...
union [all]
select ...
```

- i **duplicati vengono eliminati** (a meno che si usi **all**); anche dalle proiezioni!

68/125

```
select A, B
from R
union
select A, B
from S
```

```
select A, B
from R
union all
select A, B
from S
```

69/125

Notazione posizionale!

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

- quali nomi per gli attributi del risultato?
 - inventati o nessuno
 - quelli del primo operando
 - ...

70/125

Figlio	
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

71/125

Figlio		Padre	
Sergio	Franco	Sergio	Franco
Luigi	Olga	Luigi	Olga
Luigi	Filippo	Luigi	Filippo
Franco	Andrea	Franco	Andrea
Franco	Aldo	Franco	Aldo
Luisa	Maria	Luisa	Maria
Luisa	Luigi	Luisa	Luigi
Anna	Olga	Anna	Olga
Anna	Filippo	Anna	Filippo
Maria	Andrea	Maria	Andrea
Maria	Aldo	Maria	Aldo

72/125

Notazione posizionale, 2

```
select padre, figlio  
from paternita  
union  
select figlio, madre  
from maternita
```

```
select padre, figlio  
from paternita  
union  
select madre, figlio  
from maternita
```

73/125

Notazione posizionale, 3

- Anche con le ridenominazioni non cambia niente:

```
select padre as genitore, figlio  
from paternita  
union  
select figlio, madre as genitore  
from maternita
```

- Corretta:

```
select padre as genitore, figlio  
from paternita  
union  
select madre as genitore, figlio  
from maternita
```

74/125

Differenza

```
select Nome  
from Impiegato  
except  
select Cognome as Nome  
from Impiegato
```

solo DB2

- vedremo che si può esprimere con `select` nidificate

75/125

Intersezione

```
select Nome  
from Impiegato  
intersect  
select Cognome as Nome  
from Impiegato
```

solo DB2

- equivale a

```
select I.Nome  
from Impiegato I, Impiegato J  
where I.Nome = J.Cognome
```

76/125

Interrogazioni nidificate

- le condizioni atomiche permettono anche
 - il confronto fra **un attributo** (o più, vedremo poi) e il **risultato di una sottointerrogazione**
 - **quantificazioni esistenziali**

77/125

- nome e reddito del padre di Franco

```
select Nome, Reddito  
from Persone, Paternita  
where Nome = Padre and Figlio = 'Franco'
```

78/125

- nome e reddito del padre di Franco

```
select Nome, Reddito
from Persone, Paternita
where Nome = Padre and Figlio = 'Franco'
```

```
select Nome, Reddito
from Persone
where Nome = ( select Padre
               from Paternita
               where Figlio = 'Franco')
```

79/125

Interrogazioni nidificate, commenti

- La forma nidificata è “meno dichiarativa”, ma talvolta più leggibile (richiede meno variabili)
- La forma piana e quella nidificata possono essere **combinare**
- Le sottointerrogazioni **non** possono contenere operatori insiemistici (“l’unione si fa solo al livello esterno”); la limitazione non è significativa

81/125

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

notare la **distinct**

83/125

- nome e reddito del padre di Franco

```
select Nome, Reddito
from Persone, Paternita
where Nome = Padre and Figlio = 'Franco'
```

```
select Nome, Reddito
from Persone
where Nome = ( select Padre
               from Paternita
               where Figlio = 'Franco')
```

Uno solo!!!!

80/125

Interrogazioni nidificate, operatori

- ANY (almeno uno)
- ALL (tutti)
- IN (“presenza”)
- EXISTS (operatore esistenziale)

82/125

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

```
select Nome, Reddito
from Persone
where Nome in (select Padre
               from Paternita
               where Figlio = any (select Nome
                                   from Persone
                                   where Reddito > 20))
```

notare la **distinct**

84/125

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

```
select Nome, Reddito
from Persone
where Nome in (select Padre
from Paternita
where Figlio = any (select Nome
from Persone
where Reddito > 20))
```

"IN" ≡ "= ANY"
"not IN" ≡ "<> ALL"

notare la **distinct**

85/125

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

```
select Nome, Reddito
from Persone
where Nome in (select Padre
from Paternita, Persone
where Figlio = Nome
and Reddito > 20)
```

87/125

- Nome e reddito dei padri di persone che guadagnano più di 20, **con indicazione del reddito del figlio**

```
select distinct P.Nome, P.Reddito, F.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

```
select Nome, Reddito, ???
from Persone
where Nome in (select Padre
from Paternita
where Figlio = any (select Nome
from Persone
where Reddito > 20))
```

89/125

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
and F.Reddito > 20
```

86/125

Interrogazioni nidificate, commenti, 2

- La prima versione di SQL prevedeva **solo la forma nidificata** (o strutturata), con una sola relazione in ogni clausola FROM.
- Insoddisfacente:
 - la dichiaratività è limitata
 - non si possono includere nella target list *attributi di relazioni nei blocchi interni*

88/125

Interrogazioni nidificate, commenti, 3

- regole di **visibilità**:
 - non** è possibile fare riferimenti a variabili definite in blocchi più interni
 - se un nome di variabile è omissso, si assume riferimento alla variabile più **"vicina"**
- in un blocco si può fare riferimento a variabili definite in **blocchi più esterni**; la semantica base (prodotto cartesiano, selezione, proiezione) non funziona più, vedremo presto

90/125

Quantificazione esistenziale

- Ulteriore tipo di condizione
 - **EXISTS** (Sottoespressione)
 - risultato il valore **True** se e solo se la sottoespressione restituisce un risultato **non vuoto**

91/125

- Le persone che hanno almeno un figlio

```
select *
  from Persone
 where exists ( select *
                from Paternita
                where Padre = Nome) or
        exists ( select *
                from Maternita
                where Madre = Nome)
```

92/125

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
  from Paternita Z
 where not exists (
    select *
    from Paternita W, Persone
    where W.Padre = Z.Padre
    and W.Figlio = Nome
    and Reddito <= 20)
```

93/125

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
  from Paternita
 where not exists (
    select *
    from Persone
    where Figlio = Nome
    and Reddito <= 20)
```

NO!!!

94/125

Semantica delle espressioni “correlate”

- L'interrogazione interna viene eseguita **una volta per ciascuna ennupla** dell'interrogazione esterna

95/125

Visibilità

- scorretta:

```
select *
  from Impiegato
 where Dipart in (select Nome
                  from Dipartimento D1
                  where Nome = 'Produzione') or
        Dipart in (select Nome
                  from Dipartimento D2
                  where D2.Citta = D1.Citta)
```

96/125

Disgiunzione e unione (ma non sempre)

```
select * from Persone where Reddito > 30
union
select F.*
from Persone F, Paternita, Persone P
where F.Nome = Figlio and Padre = P.Nome
and P.Reddito > 30

select *
from Persone F
where Reddito > 30 or
exists (select *
        from Paternita, Persone P
        where F.Nome = Figlio and Padre = P.Nome
        and P.Reddito > 30)
```

97/125

Differenza e nidificazione

```
select Nome from Impiegato
except
select Cognome as Nome from Impiegato
```

98/125

Differenza e nidificazione

```
select Nome from Impiegato
except
select Cognome as Nome from Impiegato

select Nome
from Impiegato I
where not exists (select *
                  from Impiegato
                  where Cognome = I.Nome)
```

99/125

Operatori aggregati

- Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple:
 - COUNT conteggio,
 - MIN minimo,
 - MAX massimo,
 - AVG media,
 - SUM totale
 - sintassi base (semplificata):
Funzione ([DISTINCT] *)
Funzione ([DISTINCT | ALL] Attributo)

100/125

Operatori aggregati

- Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple:
 - COUNT(* | [DISTINCT] Attributo)
 - MIN | MAX (Attributo)
 - SUM ([DISTINCT] Attributo numerico)
 - AVG ([DISTINCT | ALL] Attributo numerico)

DISTINCT elimina duplicati
ALL trascura i valori nulli

101/125

Operatori aggregati: COUNT

- Il numero di figli di Franco

```
select count(*) as NumFigliDiFranco
from Paternita
where Padre = 'Franco'
```

- l'operatore aggregato (count) viene applicato al risultato dell'interrogazione:

```
select *
from Paternita
where Padre = 'Franco'
```

102/125

Paternità

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

NumFigliDiFranco

2

103/125

Altri operatori aggregati

- SUM, AVG, MAX, MIN
- Media dei redditi dei figli di Franco

```
select avg(reddito)
from persone join paternita on nome=figlio
where padre='Franco'
```

105/125

Operatori aggregati e valori nulli

```
select avg(reddito) as redditomedio
from persone
```

Persone	Nome	Età	Reddito
	Andrea	27	30
	Aldo	25	NULL
	Maria	55	36
	Anna	50	36

107/125

COUNT DISTINCT

```
select count(*) from persone
```

```
select count(distinct reddito) from persone
```

Persone	Nome	Età	Reddito
	Andrea	27	21
	Aldo	25	35
	Maria	55	21
	Anna	50	35

104/125

COUNT e valori nulli

```
select count(*) from persone
```

```
select count(reddito) from persone (ALL è di default)
```

```
select count(distinct reddito) from persone
```

Persone	Nome	Età	Reddito
	Andrea	27	21
	Aldo	25	NULL
	Maria	55	21
	Anna	50	35

106/125

Operatori aggregati e target list

- un'interrogazione scorretta:

```
select nome, max(reddito)
from persone
```

- di chi sarebbe il nome? La target list deve essere omogenea

```
select min(eta), avg(reddito)
from persone
```

108/125

Massimo e nidificazione

- La persona (o le persone) con il reddito massimo

```
select *  
from persone  
where reddito = ( select max(reddito)  
                 from persone)
```

109/125

Operatori aggregati e raggruppamenti

- Le funzioni possono essere applicate a **partizioni** delle relazioni
- Clausola **GROUP BY**:
GROUP BY listaAttributi

110/125

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

```
select Padre, count(*) AS NumFigli  
from paternita  
group by Padre
```

paternita

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

Padre	NumFigli
Sergio	1
Luigi	2
Franco	2

111/125

Semantica di interrogazioni con operatori aggregati e raggruppamenti

- interrogazione senza **group by** e **senza** operatori aggregati

```
select *  
from paternita
```

- si raggruppa e si applica l'**operatore aggregato** a ciascun gruppo

112/125

Raggruppamenti e target list

scorretta

```
select padre, avg(f.reddito), p.reddito  
from persone f join paternita on figlio = f.nome join  
persone p on padre =p.nome  
group by padre
```

corretta

```
select padre, avg(f.reddito), p.reddito  
from persone f join paternita on figlio = f.nome join  
persone p on padre =p.nome  
group by padre, p.reddito
```

113/125

Condizioni sui gruppi

- I padri i cui figli hanno un reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
select padre, avg(f.reddito)  
from persone f join paternita on figlio = nome  
group by padre  
having avg(f.reddito) > 25
```

114/125

WHERE o HAVING?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 20

```
select padre, avg(f.reddito)
from persone f join paternita on figlio = nome
where eta < 30
group by padre
having avg(f.reddito) > 20
```

115/125

Operazioni di aggiornamento

- operazioni di
 - inserimento: **insert**
 - eliminazione: **delete**
 - modifica: **update**
- di una o più ennuple di una relazione
- sulla base di una condizione che può coinvolgere anche altre relazioni

117/125

```
INSERT INTO Persone VALUES ('Mario',25,52)
```

```
INSERT INTO Persone(Nome, Eta, Reddito)
VALUES('Pino',25,52)
```

```
INSERT INTO Persone(Nome, Reddito)
VALUES('Lino',55)
```

```
INSERT INTO Persone ( Nome )
SELECT Padre
FROM Paternita
WHERE Padre NOT IN (SELECT Nome
                     FROM Persone)
```

119/125

Group by e valori nulli

```
select *
from R
```

A	B
1	11
2	11
3	null
4	null

```
select B, count (*)
from R
group by B
```

B	
11	2
null	2

```
select A, count (*)
from R
group by A
```

A	
1	1
2	1
3	1
4	1

```
select A, count (B)
from R
group by A
```

A	
1	1
2	1
3	0
4	0

116/125

Inserimento

```
INSERT INTO Tabella [ ( Attributi ) ]
VALUES( Valori )
```

oppure

```
INSERT INTO Tabella [ ( Attributi ) ]
SELECT ...
```

118/125

Inserimento, commenti

- l'**ordinamento** degli attributi (se presente) e dei valori è significativo
- le due liste debbono avere lo **stesso numero** di elementi
- se la lista di attributi è omessa, si fa riferimento a tutti gli attributi della relazione, secondo l'**ordine con cui sono stati definiti**
- se la lista di attributi non contiene tutti gli attributi della relazione, per gli altri viene inserito un **valore nullo** (che deve essere permesso) o un **valore di default**

120/125

Eliminazione di ennuple

```
DELETE FROM Tabella  
[ WHERE Condizione ]
```

121/125

Eliminazione di ennuple (esempio)

```
DELETE FROM Persone  
WHERE Eta < 35  
  
DELETE FROM Paternita  
WHERE Figlio NOT in ( SELECT Nome  
FROM Persone)  
  
DELETE FROM Paternita
```

122/125

Eliminazione, commenti

- elimina le ennuple che soddisfano la condizione
- può causare (se i vincoli di integrità referenziale sono definiti con politiche di reazione **cascade**)
eliminazioni da altre relazioni
- ricordare: se la where viene omessa, si intende **where true**

123/125

Modifica di ennuple

```
UPDATE NomeTabella  
SET Attributo = < Espressione |  
SELECT ... |  
NULL |  
DEFAULT >  
[ WHERE Condizione ]
```

124/125

Modifica di ennuple (esempi)

```
UPDATE Persone  
SET Reddito = 45  
WHERE Nome = 'Piero'  
  
UPDATE Persone  
SET Reddito = Reddito * 1.1  
WHERE Eta < 30
```

125/125