



## SQL: concetti base

### Tratto da:

Atzeni, Ceri, Paraboschi, Torlone  
Basi di dati (Capitolo 4)  
McGraw-Hill

1



## SQL

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

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, 1992, 1999, 2003, 2006, 2008, 2011, 2016 ...)
  - recepito solo in parte (!! Vedi <http://troels.arvin.dk/db/rdbms/> per un confronto)

3

## Definizione dei dati in SQL

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

4

## 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

## 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

6

## 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

7

## 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

8

## 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

9

## 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

10

## 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

11

## 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

12

## Domini

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

13

## 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

14

## Definizione di domini

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

15

## CREATE DOMAIN, esempio

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

- note:
  - Mimer OK
  - SQLServer, DB2 no

16



## 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

17

## UNIQUE e PRIMARY KEY

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

18

## 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)  
)
```

19

## PRIMARY KEY, alternative

Matricola CHAR(6) PRIMARY KEY

Matricola CHAR(6),  
...,  
PRIMARY KEY (Matricola)

20

## 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)  
)
```

21

## Attenzione: Non è la stessa cosa!

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,

22

## 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

23

## Infrazioni

<u>Codice</u>	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

<u>Matricola</u>	Cognome	Nome
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

24

## 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

25

## 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)  
)
```

26

## Modifiche degli schemi

ALTER DOMAIN

ALTER TABLE

DROP DOMAIN

DROP TABLE

...

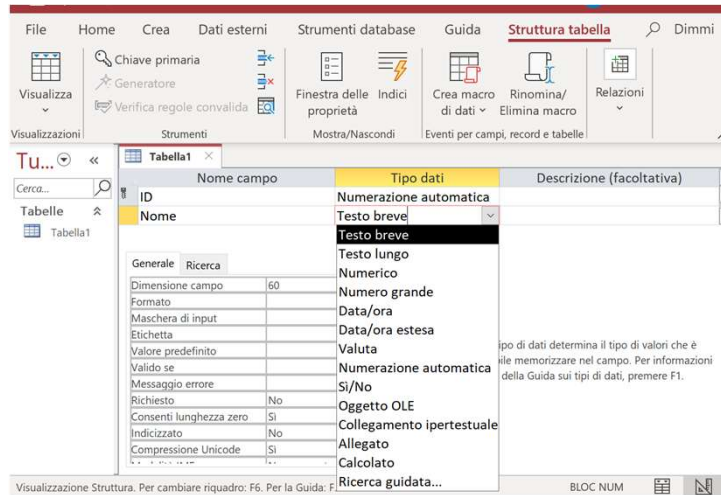
27

## 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**

28

- **DDL in pratica:** in molti sistemi si utilizzano strumenti diversi dal codice SQL per definire lo schema della base di dati
  - Esempio di creazione di una tabella in Access:



29

## SQL, operazioni sui dati

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

30

## Istruzione SELECT (versione base)

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

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

31

### 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

32



Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## Selezione e proiezione

- Nome e Reddito delle Persone con meno di trenta anni

$$\pi_{\text{Nome, Reddito}}(\sigma_{\text{Età} < 30}(\text{Persone}))$$

```
SELECT Nome, Reddito  
FROM Persone  
WHERE Età < 30
```

33

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## SELECT, abbreviazioni

```
SELECT Nome, Reddito  
FROM Persone  
WHERE Età < 30
```

```
SELECT P.Nome AS Nome,  
       P.Reddito AS Reddito  
FROM Persone AS P  
WHERE P.Età < 30
```

34

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## Selezione, senza proiezione

- Nome, età e Reddito delle Persone con meno di trenta anni

$\sigma_{\text{Età} < 30}(\text{Persone})$

```
SELECT *  
FROM   Persone  
WHERE  Eta < 30
```

35

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## SELECT, abbreviazioni

```
SELECT *  
FROM   Persone  
WHERE  Eta < 30
```

```
SELECT Nome, Eta, Reddito  
FROM   Persone  
WHERE  Eta < 30
```

36

## Proiezione, senza selezione

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- Nome e Reddito di tutte le Persone

$\pi_{\text{Nome, Reddito}}(\text{Persone})$

```
SELECT Nome, Reddito  
FROM Persone
```

37

## SELECT, abbreviazioni

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- $R(A,B)$

```
SELECT *  
FROM R
```

equivale (intuitivamente) a

```
SELECT X.A as A, X.B as B  
FROM R X  
WHERE true
```

38

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## Espressioni nella target list

```
SELECT Reddito/2 AS RedditoSemestrale  
FROM   Persone  
WHERE  Nome = 'Luigi'
```

39

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## Condizione complessa

```
SELECT *  
FROM   Persone  
WHERE  Reddito > 25  
AND    (Eta < 30 OR Eta > 60)
```

40

## Condizione “LIKE”

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- 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*'
```

41

## Utilizzo della Condizione “LIKE” in Access

Character	Da usare per abbinare
? o _ (carattere di sottolineatura)	Qualsiasi carattere singolo
* o %	Zero o più caratteri
#	Qualsiasi cifra singola (0 - 9)
[elencocaratteri]	Qualsiasi carattere singolo incluso in elencocaratteri (es. [abcd] e [a-d] corrispondono ad un qualsiasi carattere tra a, b or c)
[!elencocaratteri]	Qualsiasi carattere singolo non incluso in elencocaratteri
[a-zA-Z0-9].	Qualsiasi carattere alfanumerico
[A-Z]	Qualsiasi lettera maiuscola nell'intervallo da A a Z. <b>Nota:</b> Quando si specifica un intervallo di caratteri, i caratteri devono essere disposti in ordine crescente. Ad esempio, [Z-A] non è un criterio valido.

42

## 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

$\sigma_{(Età > 40) \text{ OR } (Età \text{ IS NULL})}$  (Impiegati)

43

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

$\sigma_{Età > 40 \text{ OR } Età \text{ IS NULL}}$  (Impiegati)

```
SELECT *  
FROM Impiegati  
WHERE Eta > 40 or Eta is NULL
```

44

## Proiezione in SQL e in AR

- cognome e filiale di tutti gli impiegati:

```
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

$\pi_{\text{Cognome, Filiale}}(\text{Impiegati})$

45

## Selezione, proiezione e JOIN

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- Istruzioni SELECT con una sola relazione nella clausola FROM permettono di realizzare:

- selezioni, proiezioni, ridenominazioni

```
SELECT DISTINCT Nome AS Cittadino, Reddito  
FROM   Persone  
WHERE  Eta < 30
```

- con più relazioni nella FROM si realizzano JOIN (e prodotti cartesiani)

46

## 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

47

## 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 ...)

48



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

49

## Esempi

- I padri di Persone che guadagnano più di 20

Maternità(Madre, Figlio)

Paternità(Padre, Figlio)

Persone(Nome, Età, Reddito)

50

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- Le Persone che guadagnano più dei rispettivi padri; mostrare nome, Reddito e Reddito del padre

51

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## SELECT, con ridenominazione del risultato

```
SELECT f.Nome AS Persona,  
       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
```

52

## Join esplicito

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- 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
```

53

## SELECT con JOIN esplicito, sintassi

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

54

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- 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
```

55

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## JOIN naturale (meno diffuso)

Paternita ⋈ 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

56

## 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 è opzionale
- RIGHT JOIN, FULL JOIN

57

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

## Ordinamento del risultato

- Nome e Reddito delle Persone con meno di trenta anni in ordine alfabetico

```
SELECT Nome, Reddito  
FROM Persone  
WHERE Età < 30  
ORDER BY Nome
```

58

```
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

59

## 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!

60

```
SELECT A, B
FROM R
UNION
SELECT A, B
FROM S
```

```
SELECT A, B
FROM R
UNION ALL
SELECT A, B
FROM S
```

61

## Notazione posizionale!

```
Maternità(Madre, Figlio)
Paternità(Padre, Figlio)
Persone(Nome, Età, Reddito)
```

```
SELECT Padre, Figlio
FROM paternita
UNION
SELECT Madre, Figlio
FROM maternita
```

- Importante, che
  - 1) Il numero di attributi in ciascun SELECT coincide e
  - 2) gli attributi corrispondenti sono compatibili
- Quali nomi per gli attributi del risultato?
  - Di solito, quelli del primo operando

62

```
SELECT Padre, Figlio
FROM paternita
UNION
SELECT Madre, Figlio
FROM maternita
```

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

63

## Differenza

```
SELECT Nome
FROM Impiegato
EXCEPT
SELECT Cognome
FROM Impiegato
```

solo DB2

- vedremo che si può esprimere con **SELECT** nidificate

64



## Intersezione

```
SELECT Nome  
FROM Impiegato  
INTERSECT  
SELECT Cognome  
FROM Impiegato
```

solo DB2

- equivale a

```
SELECT I.Nome  
FROM Impiegato I, Impiegato J  
WHERE I.Nome = J.Cognome
```

65

## Interrogazioni nidificate

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

```
SELECT Nome, Reddito  
FROM Persone  
WHERE Nome = ( SELECT Padre  
                FROM Paternita  
                WHERE Figlio = 'Franco')
```

- quantificazioni esistenziali

66

## Interrogazioni nidificate

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- 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' )
```

67

## Interrogazioni nidificate

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- nome e Reddito del padre di Franco e del padre di Olga

```
SELECT Nome, Reddito
FROM Persone, Paternita
WHERE Nome = Padre
AND ( Figlio = 'Franco' OR Figlio = 'Olga' )
```

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome = ANY ( SELECT Padre
                   FROM Paternita
                   WHERE Figlio = 'Franco'
                   OR Figlio = 'Olga' )
```

68

## Interrogazioni nidificate

Maternità(Madre, Figlio)  
Paternità(Padre, Figlio)  
Persone(Nome, Età, Reddito)

- nome e Reddito del padre di Franco e del padre di Olga

```
SELECT Nome, Reddito
FROM   Persone, Paternita
WHERE  Nome = Padre
AND    ( Figlio = 'Franco' OR Figlio = 'Olga' )
```

```
SELECT Nome, Reddito
FROM   Persone
WHERE  Nome IN ( SELECT Padre
                  FROM   Paternita
                  WHERE  Figlio = 'Franco'
                  OR     Figlio = 'Olga' )
```

69

## Interrogazioni nidificate

- nome e Reddito del padre di Franco

```
SELECT Nome, Reddito
FROM   Persone
WHERE  Nome = ( SELECT Padre
                FROM   Paternita
                WHERE  Figlio = 'Franco' )
```

```
SELECT Nome, Reddito
FROM   Persone
WHERE  Nome IN ( SELECT Padre
                 FROM   Paternita
                 WHERE  Figlio = 'Franco' )
```

70

## Interrogazioni nidificate, commenti

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

71

- 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))
```

72

- 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)
```

73

## 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

74

- 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))
```

75

### 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

76

## Quantificazione esistenziale

- Ulteriore tipo di condizione
  - **EXISTS** ( Sottoespressione )

77

- Le Persone che hanno almeno un figlio

```
SELECT *
FROM Persone P
WHERE EXISTS (
    SELECT *
    FROM Paternita
    WHERE Padre = P.Nome) OR
    EXISTS (
    SELECT *
    FROM Maternita
    WHERE Madre = P.Nome)
```

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

78

## Intersezione e EXISTS

```
SELECT CF
FROM Studente
INTERSECT
SELECT CF
FROM Lavoratore
```

```
SELECT CF
FROM Studente S
WHERE EXISTS (SELECT *
              FROM Lavoratore L
              WHERE L.CF = S.CF)
```

79

## Intersezione e IN

```
SELECT CF
FROM Studente
INTERSECT
SELECT CF
FROM Lavoratore
```

```
SELECT CF
FROM Studente S
WHERE CF IN (SELECT CF
            FROM Lavoratore L)
```

```
SELECT CF
FROM Studente S JOIN Lavoratore L ON S.CF=L.CF
```

80



## Differenza e NOT EXISTS

```
SELECT CF  
FROM Studente  
EXCEPT  
SELECT CF  
FROM Lavoratore
```

```
SELECT CF  
FROM Studente S  
WHERE NOT EXISTS (SELECT *  
                  FROM Lavoratore L  
                  WHERE L.CF = S.CF)
```

81

## Differenza e NOT IN

```
SELECT CF  
FROM Studente  
EXCEPT  
SELECT CF  
FROM Lavoratore
```

```
SELECT CF  
FROM Studente S  
WHERE S.CF NOT IN (SELECT CF  
                  FROM Lavoratore L )
```

82

## Semantica delle espressioni “correlate”

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

83

I padri i cui figli guadagnano **tutti** più di 20

*I padri per i quali **non esiste** neanche un figlio che guadagna  $\leq 20$*

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
  SELECT *
  FROM Paternita W, Persone P
  WHERE W.Padre = Z.Padre
  AND W.Figlio = P.Nome
  AND P.Reddito <= 20)
```

ok

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
  SELECT *
  FROM Persone P
  WHERE Z.Figlio = P.Nome
  AND P.Reddito <= 20)
```

no

84

Una dopo l'altra si considerano le righe della tabella paternità:

- Fissata la riga  $r$ , essa determina il valore di **Z. Padre** e di **Z. Figlio**. Si calcola la sottoquery utilizzando il valore di **Z. Figlio**. Se il suo risultato non è una tabella vuota, allora il valore di **Z. Padre** farà parte del risultato della query.
- Quindi, di fatto la sottoquery considera il Reddito di un solo figlio, quello della riga  $r$ .

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
  SELECT *
  FROM Persone P
  WHERE Z.Figlio = P.Nome
  AND P.Reddito <= 20)
```

85

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

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
  SELECT *
  FROM Persone P
  WHERE Z.Figlio = P.Nome
  AND P.Reddito <= 20)
```

86

Una dopo altra si considerano le righe della tabella paternità:

- Fissata la riga  $r$ , essa determina il valore di **Z. Padre** e di **Z. Figlio**. Si calcola la sottoquery utilizzando il valore di **Z. Padre**. Se il suo risultato non è una tabella vuota, allora il valore di **Z. Padre** farà parte del risultato della query.
- La sottoquery considera il Reddito di tutti i figli del padre **Z. Padre** grazie all'utilizzo della seconda copia della tabella *Paternità*, qui chiamata **W**.

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
    SELECT *
    FROM Paternita W, Persone P
    WHERE W.Padre = Z.Padre
    AND W.Figlio = P.Nome
    AND P.Reddito <= 20)
```

87

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

```
SELECT DISTINCT Z.Padre
FROM Paternita Z
WHERE NOT EXISTS (
    SELECT *
    FROM Paternita W, Persone P
    WHERE W.Padre = Z.Padre
    AND W.Figlio = P.Nome
    AND P.Reddito <= 20)
```

88

## 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)
```

89

## 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)
```

90

## Operatori aggregati

- Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple:
  - conteggio, minimo, massimo, media, totale
  - sintassi base (semplificata):

COUNT ( \* )  
COUNT ( [ DISTINCT ] Attributo/i )  
FUNZIONE ( [ DISTINCT ] AttrEspr ), dove  
FUNZIONE è MIN, MAX, AVG, o SUM

91

## 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'
```

92

Paternità

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

NumFigliDiFranco

2

93

## 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

94

## 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'
```

95

## COUNT e valori nulli

```
SELECT count(*) FROM Persone
```

```
SELECT count(Reddito) FROM Persone
```

```
SELECT count(distinct Reddito) FROM Persone
```

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

96



## 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

97

## 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
```

98

## Massimo e nidificazione

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

```
SELECT *  
FROM Persone  
WHERE Reddito = ( SELECT max(Reddito)  
                  FROM Persone)
```

99

## Operatori aggregati e raggruppamenti

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

100

## Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

```
SELECT Padre, count(*) AS NumFigli
FROM Paternita
GROUP BY Padre
```

Paternita	Padre	Figlio	Padre	NumFigli
	Sergio	Franco	Sergio	1
	Luigi	Olga	Luigi	2
	Luigi	Filippo	Luigi	2
	Franco	Andrea	Franco	2
	Franco	Aldo	Franco	2

101

## Semantica di interrogazioni con operatori aggregati e raggruppamenti

1. interrogazione senza **GROUP BY** e senza operatori aggregati

```
SELECT *
```

```
FROM Paternita
```

2. si raggruppa e si applica l'operatore aggregato a ciascun gruppo

102

## 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
```

103

## Condizioni sui gruppi

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

```
SELECT p.Padre, avg(f.Reddito)  
FROM Persone f JOIN Paternita p ON p.Figlio = f.Nome  
GROUP BY p.Padre  
HAVING avg(f.Reddito) > 25
```

104

## WHERE o HAVING?

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

```
SELECT p.Padre, avg(f.Reddito)
FROM Persone f JOIN Paternita p ON p.Figlio = f.Nome
WHERE f.Eta < 30
GROUP BY p.Padre
HAVING avg(f.Reddito) > 20
```

105

## Group by e valori nulli

```
SELECT B, count (*)
FROM R
GROUP BY B
```

B	
11	2
null	2

A	B
1	11
2	11
3	null
4	null

```
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

106

## Operazioni di aggiornamento

- operazioni di
  - inserimento: **INSERT**
  - eliminazione: **DELETE**
  - modifica: **UPDATE**
- di una o più entuple di una relazione
- sulla base di una condizione che può coinvolgere anche altre relazioni

107

## Inserimento

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

oppure

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

108

```
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)
```

109

## 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

110

## Eliminazione di ennuple

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

111

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

112



## 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**

113

## Modifica di ennuple

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

114



```
UPDATE Persone SET Reddito = 45  
WHERE Nome = 'Piero'
```

```
UPDATE Persone  
SET Reddito = Reddito * I.I  
WHERE Eta < 30
```