Laurea in Informatica
A.A. 2021-2022

Corso "Base di Dati"

Concetti di Base di SQL

Prof. Massimiliano de Leoni

Università degli Studi di Padova

Structured Query Language: SQL

- Linguaggio con varie funzionalità:
  - Data Definition Language
  - Data Manipulation Language

Definizione dei dati in SQL

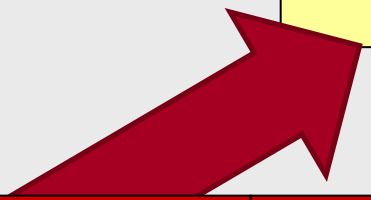
- Istruzione CREATE TABLE:
  - definisce uno schema di relazione e ne crea un'istanza vuota
  - specifica attributi, domini e vincoli

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)
 )
```

Dipartimento

NomeDip	…
Math	…



Impiegato

Matricola	Nome	Cognome	Dipart	Stipendio
123	Max	de Leoni	Math	123456
…	…	…	…	…

Domini

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

Domini elementari

- Stringhe di lunghezza X:
  - Fissa: char(X)
  - Approssimati: varchar(X)
- Tipi Numerici: integer, smallint, float, …
- Tipi Numerici esatti con X cifre intere (e Y decimali): numeric(X,Y)
- Data, ora, data+ora: date, time, timestamp
- Boolean

Definizione di domini

- Istruzione CREATE DOMAIN
- Definisce un dominio (semplice), con eventuali vincoli e valori di default

CREATE DOMAIN, esempio

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

Vincoli intrarelazionali

- NOT NULL
- UNIQUE definisce chiavi
- PRIMARY KEY: chiave primaria
- UNIQUE + NOT NULL: chiave (non primaria)
- CHECK: Vincoli generici

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)
 )
```

UNIQUE e PRIMARY KEY

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

```
Matricola CHAR(6) PRIMARY KEY
```

oppure

```
Matricola CHAR(6),
…
PRIMARY KEY (Matricola)
```

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)
 )
```

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 sono la stessa cosa:

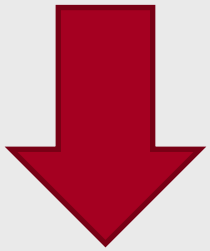
- Caso sopra: (Cognome,Nome) è chiave
  = Impossibile avere due tuple con lo stesso cognome e lo stesso nome
- Caso sotto: Cognome è chiave + Nome è chiave
  = Impossibile avere due tuple con lo stesso cognome
  = Impossibile avere due tuple con lo stesso nome

Vincoli interrelazionali

- 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

Infrazioni

Codice	Data	Vigile	Stato	Numero
34321	1/2/15	3987	I	CC953MS
53524	4/3/15	3295	I	FV077XM
64521	5/4/16	3295	F	AB234ZK
73321	5/2/18	9345	F	AB234ZK



Auto

Stato	Numero	Cognome	Nome
I	CC953MS	Rossi	Mario
I	FV077XM	Rossi	Mario
F	AB234ZK	Neri	Luca

Vigili

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

```
CREATE TABLE Infrazioni(
    Codice CHAR(5) PRIMARY KEY,
    Data DATE NOT NULL,
    Vigile INTEGER NOT NULL REFERENCES Vigili(Matricola),
    Stato VARCHAR(2),
    Numero VARCHAR(8) ,
    FOREIGN KEY(Stato, Numero) REFERENCES Auto(Stato, Numero),
    CHECK(Data > '01/01/2020')
)
```

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(Stato, Numero)
                  REFERENCES Auto(Stato, Numero)
 )
```

Politiche di reazione

- Specificata immediatamente dopo il vincolo di integrità consente di associare politiche diverse ai diversi eventi (delete, update) secondo la seguente sintassi:

on < delete | update >

< cascade | set null | set default | no action >

DELETE : Politiche di reazione

- **cascade**: si propagano le cancellazioni.
- **set null**: all'attributo referente viene assegnato il valore nullo al posto del valore cancellato nella tabella
- **set default**: all'attributo referente viene assegnato il valore di default al posto del valore cancellato nella tabella esterna
- **no action**: la cancellazione non viene consentita

UPDATE : Politiche di reazione

- **cascade:** il nuovo valore viene propagato nell'altra tabella.
- **set null:** all'attributo referente viene assegnato il valore nullo al posto del valore modificato nella tabella.
- **set default:** all'attributo referente viene assegnato il valore di default al posto del valore modificato nella tabella esterna.
- **no action:** l'azione di modifica non viene consentita.

Infrazioni

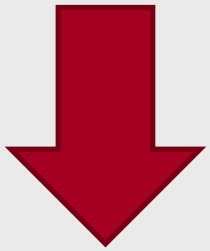
Codice	Data	Vigile	Stato	Numero
34321	1/2/15	3987	I	CC953MS
53524	4/3/15	3295	I	FV077XM
64521	5/4/16	3295	F	AB234ZK
73321	5/2/18	9345	F	AB234ZK

Auto

Stato	Numero	Cognome	Nome
I	CC953MS	Rossi	Mario
I	FV077XM	Rossi	Mario
F	AB234ZK	Neri	Luca

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

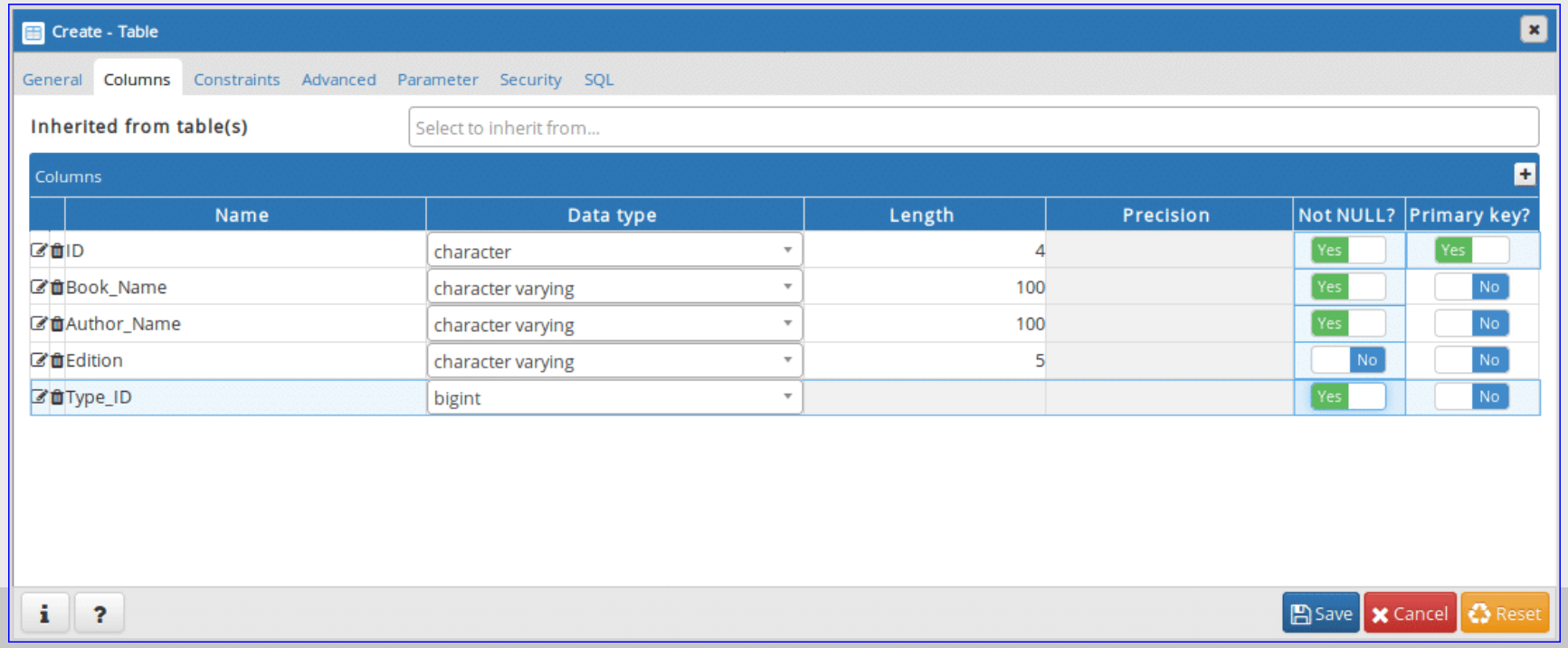
Vigili



```
CREATE TABLE Infrazioni(
   Codice CHAR(5) PRIMARY KEY,
   Data DATE NOT NULL,
   Vigile INTEGER NOT NULL REFERENCES Vigili(Matricola)
          on update cascade
          on delete no action,
   …
)
```

DDL, in pratica

In molti sistemi si utilizzano strumenti grafici per definire lo schema della base di dati, tradotti internamente in SQL



SQL, operazioni sui dati

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

Operazioni di aggiornamento

- operazioni su 0+ tuple di una relazione:
  - inserimento: INSERT
  - eliminazione: DELETE
  - modifica: UPDATE

- sulla base di una condizione che può coinvolgere anche altre relazioni (ricorda l'effetto «cascade»)

Inserimento

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

oppure

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

Inserimento: Esempi

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

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

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

INSERT INTO Persone ( Nome, Età )
SELECT Padre, 25
FROM Paternita
WHERE Padre NOT IN (SELECT Nome
                    FROM Persone)
```

25 è una costante uguale per tutte le tuple aggiunte

Paternità

Padre	Figlio
…	…

Persone

Nome	Età	Reddito
…	…	…

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

Eliminazione di tuple

DELETE FROM Tabella
[WHERE Condizione]

Eliminazione: Esempi

```
DELETE FROM Persone
WHERE Eta < 35
```

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

```
DELETE FROM Paternita
```

Paternità

Padre	Figlio
…	…

Persone

Nome	Età	Reddito
…	…	…

Eliminazione, commenti

- elimina le tuple che soddisfano la condizione
- può causare eliminazioni da altre relazioni in caso di «on delete cascade»
- ricordare: se la WHERE viene omessa, si intende WHERE true!!!

Modifica di tuple

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

Modifica: Esempi

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

UPDATE Persone
  SET Reddito = Reddito * 1.1
  WHERE Eta < 30
```

Paternità

Padre	Figlio
…	…

Persone

Nome	Età	Reddito
…	…	…

Istruzione SELECT (versione base)

SELECT ListaAttributi
FROM ListaTabelle
[WHERE Condizione]

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

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

Selezione e proiezione

Nome e reddito delle persone con meno di trenta anni

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

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

Persone

Nome	Età	Reddito
...	...	...

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
```

Selezione, senza proiezione

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

$$\sigma_{Eta<30}(Persone)$$

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

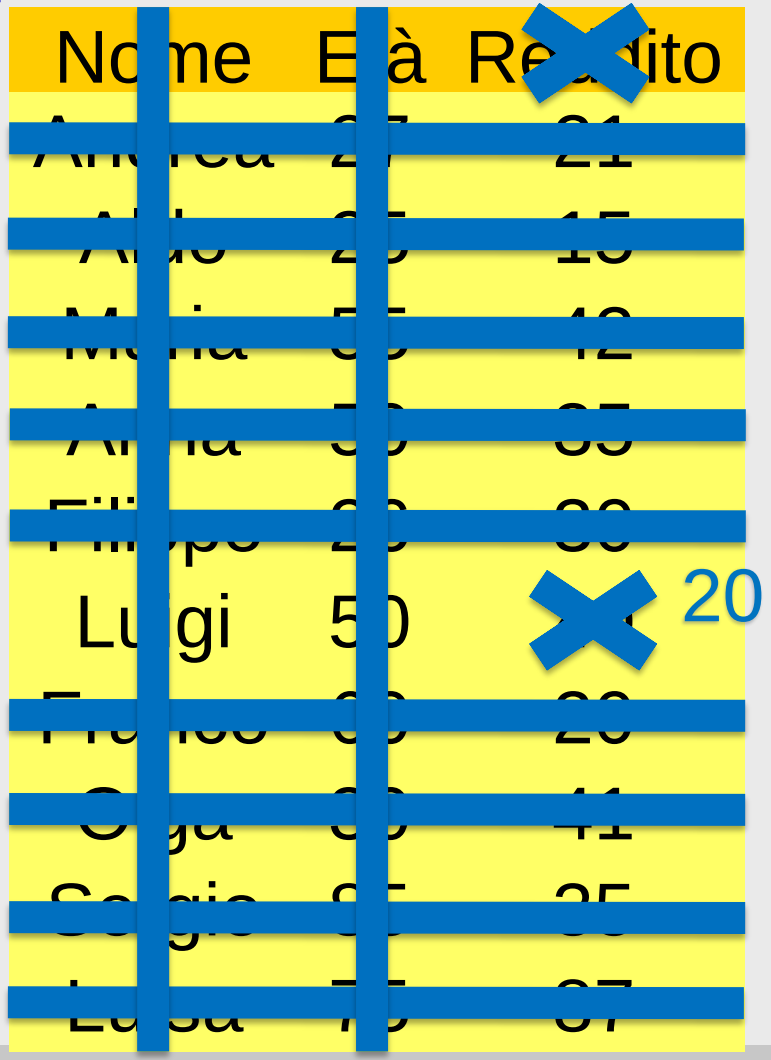
Persone

Nome	Età	Reddito
...	...	...

Espressioni nella target list

Persone

RedditoSemestrale




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

$\rho_{RedditoSemestrale \leftarrow Reddito}$ ($\pi_{Reddito}(\sigma_{Nome="Luigi"}(Persone))/2$)

Condizione complessa

```
SELECT *
FROM Persone
WHERE Reddito > 25
    and (Eta < 30 or Eta > 60)
```

$\sigma_{(Eta < 30 \text{ or } Eta > 60) \text{ and } Reddito > 25}$ (Persone)

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%'
```

Simbolo	Significato
_	Qualsiasi carattere
%	Qualsiasi sequenza anche vuota

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

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

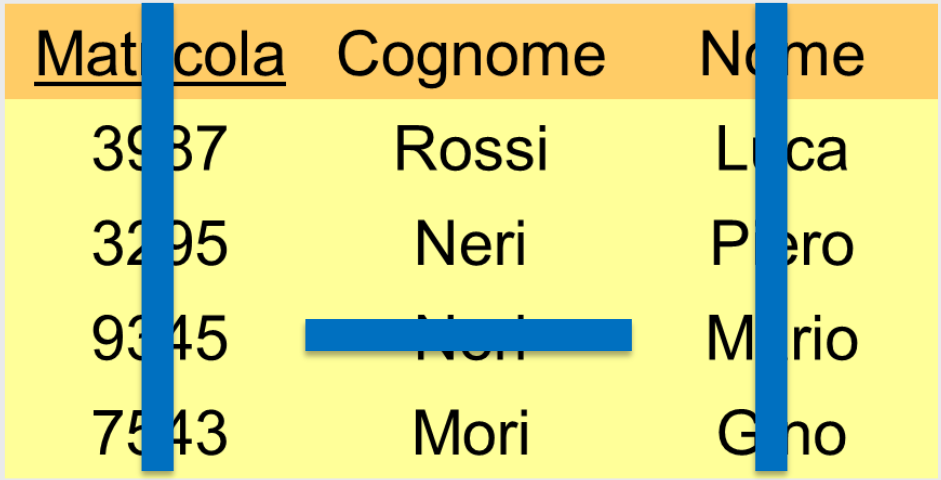
- Gli impiegati la cui età è o potrebbe essere maggiore di 40

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

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

Proiezione: Differenze tra SQL e Algebra Relazionale

Cognome dei vigili



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

Vigili

π_{Cognome} (Vigili)

```
SELECT Cognome
FROM Vigili
```

Cognome
Rossi
Neri
Neri
Mori

```
SELECT DISTINCT Cognome
FROM Vigili
```

Cognome
Rossi
Neri
Mori

SELECT non rimuove i duplicati.
Occorre Aggiungere DISTINCT

Selezione, proiezione e join / 1

Istruzioni SELECT più relazioni nella FROM si realizzano join (e prodotti cartesiani)

Esempio:

Supponiamo due relazioni R1(A1,A2) e R2(A1,A3)

La query $\pi_{R1.A1,A3}(\sigma_{R1.A1>R2.A1}(R1 \bowtie R2))$ è

```
SELECT R1.A1, A3
FROM   R1, R2
WHERE  R1.A1 > R2.A1
```

Selezione, proiezione e join / 2

Forse necessarie ridenominazioni

Esempio:

Supponiamo una relazione R1(A1,A2)

R2=R1

$\rho_{B1 \leftarrow R1.A1, B2 \leftarrow R1.A2}(\pi_{R1.A1,R1.A2}(\sigma_{R1.A1>R2.A1}(R1 \bowtie R2)))$

diventa:

```
SELECT R1.A1 AS B1, R1.A2 AS B2
FROM  R1, R1 AS R2
WHERE  R1.A1 > R2.A1
```

Osservazione: Specifica delle interrogazioni

- DBMS "ottimizzano" le interrogazioni
  → non necessario preoccuparsi dell'efficienza

- Importante preoccuparsi della chiarezza

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

Esempio: I padri di persone che guadagnano più di 20

$$\pi_{Padre}(\sigma_{Reddito>20 \wedge Figlio = Nome}(paternita \bowtie persone))$$

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

```
SELECT DISTINCT Padre
FROM Persone, Paternita
WHERE Figlio = Nome and Reddito > 20
```

Selezionerebbe quindi:
(Franco, Aldo, Aldo, 25, 15)

Esempio: Il nome delle persone che guardagnano più dei rispettivi padri

$P1 = Persone$

$P2 = Persone$

$\pi_{Figlio}($

$\sigma_{Figlio = P1.Nome \wedge Padre = P2.Nome \wedge P2.reddito < P1.reddito}($

$(paternita \bowtie P1) \bowtie P2$

$))$

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

```
SELECT DISTINCT Paternita.Figlio
FROM Persone AS P1, Persone AS P2, Paternita
WHERE Figlio = P1.Nome AND Padre = P2.Nome
                AND P2.Reddito < P1.Reddito
```

Join esplicito / 1

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

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
```

Paternità

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

Join esplicito / 2

Persone che guardagnano più dei rispettivi padri

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40

```
SELECT DISTINCT Paternita.Figlio
   FROM Persone P1, Persone P2, Paternita
   WHERE Figlio = P1.Nome AND Padre = P2.Nome
                AND P2.Reddito > P1.Reddito

   SELECT Paternita.Figlio
   FROM (Persone P2 join Paternita on P2.Nome = Padre)
                join Persone P1 on Figlio = P1.Nome
   WHERE P2.Reddito > P1.Reddito
```

Maternità

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

Join esterno: "outer join"

Paternità

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

Padre e, se nota, madre di ogni persona

```
SELECT Padre, Paternita.Figlio, Madre
FROM Paternita left join Maternita
        on Paternita.Figlio = Maternita.Figlio
```

```
SELECT Padre, Paternita.Figlio, Madre
FROM Paternita left outer join Maternita
        on Paternita.Figlio = Maternita.Figlio
```

Risultato: Differenza tra “left join” e “inner join”

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

```
SELECT Padre, Paternita.Figlio, Madre
FROM Paternita left join Maternita
on Paternita.Figlio =
Maternita.Figlio
```

Padre	Pat.Figlio	Madre
Sergio	Franco	*NULL*
Luigi	Olga	Anna
Luigi	Filippo	Anna
Franco	Andrea	Maria
Franco	Aldo	Maria

Se «Join» senza «Left Join» (conosciuto anche come «Inner Join»), la prima riga sarebbe esclusa dal risultato

Risultato: Differenza tra “left join” e “full join”

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

```
SELECT * FROM Paternita full outer join
    Maternita
    on Paternita.Figlio = Maternita.Figlio
```

Padre	Pat.Figlio	Mat.Figlio	Madre
Sergio	Franco	*NULL*	*NULL*
Luigi	Olga	Olga	Anna
Luigi	Filippo	Filippo	Anna
Franco	Andrea	Andrea	Maria
Franco	Aldo	Aldo	Maria
NULL	*NULL*	Maria	Luisa
NULL	*NULL*	Luigi	Luisa

Righe aggiunte perché «full join»

Ordinamento del risultato

Nome e reddito delle persone con meno di 31 anni in ordine alfabetico

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 31
ORDER BY Nome
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40

```
SELECT Nome, Reddito
FROM Persone
WHERE ETA < 31
```

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

```
SELECT Nome, Reddito
FROM Persone
WHERE ETA < 31
ORDER BY Nome
```

Nome	Reddito
Aldo	15
Andrea	21
Filippo	30

Operatori aggregati

Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di tuple:

- Conteggio (COUNT)
- Minimo (MIN)
- Massimo (MAX)
- Media (AVG)
- Somma (SUM)

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'
```

Paternità

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

Paternità

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

NumFigliDiFranco
2

COUNT DISTINCT

SELECT count(*) FROM persone

4

SELECT count(distinct reddito) FROM persone

2

Persone

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

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'
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30

Paternità

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

COUNT e valori nulli

SELECT count(*) FROM persone — 4

SELECT count(reddito) FROM persone — 3

SELECT count(distinct reddito) FROM persone — 2

Persone

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

Operatori aggregati e valori nulli

- Tutti gli operatori aggregati, ignorano i null.
- Esempio: *Il reddito medio delle persone*

Persone

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

```
SELECT avg(reddito) as Reddito_Medio
FROM persone
```

Reddito_Medio
34

Un Esempio: Attenzione!!!

- Interrogazione scorretta perché non chiaro di chi sarebbe il nome:

```
SELECT nome, max(reddito)
FROM persone
```

- Non restituisce il nome della persona con il max reddito (potrebbero anche essere molti)

Persone

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

Operatori aggregati e raggruppamenti

- Operatore GROUP BY permette di fare gruppi per min, max, count, ecc..
- Esempio: *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

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

Condizioni sui gruppi con HAVING

I padri con figli i cui reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
SELECT padre, avg(reddito)
FROM persone, paternita
WHERE  nome=figlio
GROUP by padre
HAVING avg(reddito) > 25
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30

Paternità

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

Condizioni sui gruppi con HAVING

I padri con figli i cui reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
SELECT padre, avg(reddito)
FROM persone, paternita
WHERE  nome=figlio
GROUP by padre
HAVING avg(reddito) > 25
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30

$\pi_{padre,reddito}$ (Paternità $\bowtie_{nome=figlio}$ Persone)

Padre	Reddito
Sergio	35
Luigi	42
Luigi	30
Franco	21
Franco	15

Condizioni sui gruppi con HAVING

I padri con figli i cui reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
SELECT padre, avg(reddito)
FROM persone, paternita
WHERE  nome=figlio
GROUP by padre
HAVING avg(reddito) > 25
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30

$\pi_{padre,reddito}$ (Paternità $\bowtie_{nome=figlio}$ Persone)

Padre	Reddito
Sergio	35
Luigi	42
Luigi	30

Condizioni sui gruppi con HAVING

I padri con figli i cui reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
SELECT padre, avg(reddito)
   as media
FROM persone, paternita
WHERE  nome=figlio
GROUP by padre
HAVING avg(reddito) > 25
```

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30

Padre	media
Sergio	35
Luigi	36

«Group by» e valori nulli

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

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!

UNION: Un esempio

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

```
SELECT A,  B
FROM R
union all
SELECT A , B
FROM S
```

Notazione posizionale!

```
SELECT padre, figlio
FROM paternita
union
SELECT madre, figlio
FROM maternita
```

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

- Quali nomi per gli attributi del risultato? Tipicalmente, quelli del primo operando

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

```
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

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
```

- Per essere certi del risultato:

```
SELECT padre as genitore, figlio
FROM paternita
union
SELECT madre as genitore, figlio
FROM maternita
```

Differenza e Intersezione

Differenza

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

Intersezione

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

Intersezione: Zucchero Sintattico

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

equivale a

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

Interrogazioni Nidificate

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

Nome e reddito del padre di Franco

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

Interrogazione all'interno di un'altra

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

Interrogazioni Nidificate

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

Le persone con il reddito superiore alla media

```
SELECT *
FROM Persone
WHERE Reddito >=     (SELECT avg(Reddito)
                     FROM Persone)
```

Operatore «In»

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

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)
```

Operatore «Any»

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

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

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome = any (SELECT Padre
                  FROM Paternita, Persone
                  WHERE Figlio = Nome and Reddito > 20)
```

Ovviamente l'alternativa più ovvia è

```
SELECT distinct P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre and Figlio = F.Nome and
F.Reddito > 20
```

Operatore «All»

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

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

```
SELECT Nome, Reddito
FROM Paternita JOIN Persone
ON Padre=Nome
WHERE Nome <> all (SELECT Padre
                   FROM Paternita, Persone
                   WHERE Figlio = Nome and Reddito <= 20)
```

Ovviamente l'alternativa più ovvia è

```
SELECT distinct P.Nome, P.Reddito
FROM Persone P, Paternita, Persone F
WHERE P.Nome = Padre and Figlio = F.Nome and
F.Reddito > 20
```

Operatore «All»

Nome e reddito delle persone che guardagnano di più

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

```
SELECT Nome, Reddito
FROM Persone
WHERE Nome >= all (SELECT Padre
                   FROM Paternita, Persone
                   WHERE Figlio = Nome and Reddito <= 20)
```

Ovviamente l'alternativa più ovvia è

```
SELECT distinct P.Nome, P.Reddito FROM Persone P
WHERE Reddito = (SELECT MAX(REDDITO)
                 FROM PERSONE)
```

Quantificazione esistenziale

- Ulteriore tipo di condizione
  - EXISTS (Sottoespressione)

- L’interrogazione interna viene eseguita una volta per ciascuna tupla dell’interrogazione esterna

Quantificazione esistenziale: Esempio 1

Relazione Persona(CodFisc, Nominativo, Città)

Estrarre le persone omonime

```
SELECT *
FROM Persona AS P
WHERE EXISTS    (SELECT *
                 FROM PERSONA Q
                 WHERE Q.Nominativo=P.Nominativo
                 AND Q.CodFisc=P.CodFisc)
```

Stesso esempio senza EXIST

Relazione Persona(CodFisc, Nominativo, Città)

Estrarre le persone omonime

```
SELECT P.*
FROM Persona AS P, Persona AS Q
WHERE P.Nominativo = Q.Nominativo AND
                     P.CodFisc <> Q.CodFisc
```

Quantificazione esistenziale: Esempio 2

Relazione Persona(CodFisc, Nominativo, Città)

Estrarre le persone senza omonimi

```
SELECT *
FROM Persona AS P
WHERE NOT EXISTS      (SELECT *
                       FROM PERSONA Q
                       WHERE Q.Nominativo=P.Nominativo
                       AND Q.CodFisc=P.CodFisc)
```

Stesso esempio senza EXIST

Relazione Persona(CodFisc, Nominativo, Città)

Estrarre le persone senza omonimi

```
SELECT P.*
FROM Persona AS P
WHERE P.Nominativo NOT IN
                    (SELECT Nominativo
                    FROM Persona AS Q
                    WHERE Q.Nominativo = P.Nominativo
                          AND Q.CF <> P.CF)
```

Quantificazione esistenziale: Esempio 3

Relazione Persona(CodFisc, Nominativo, Città)

Estrarre le città con almeno due persone nel DB

```
SELECT Città
FROM Persona AS P
WHERE EXISTS    (SELECT *
                 FROM PERSONA Q
                 WHERE Q.Città=P.Città
                 AND Q.CodFisc=P.CodFisc)
```

Visibilità in Query Annidate

- Visibilità è solo in query annidate
- La seguente query non è annidata, quindi è 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)
```

Riferimento

Capitolo 4 del libro, escluse:

- Sezione 4.2.8
- Sezione 4.2.9