Informatica Grafica

Corso di Laurea in Ingegneria Edile – Architettura

Basi di dati – parte 2

Michele Lombardi
su materiale originario di Paolo Torroni

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
Università degli Studi di Bologna

Anno Accademico 2010/2011

Parte I

Riepilogo

Riepilogo

Problema di partenza: **gestire collezioni di dati**

- efficienza anche su collezioni di grandi dimensioni
- consentire accesso condiviso
- protezione delle informazioni
- affidabilità a fronte di guasti

Soluzione: **Data Base Management System**

- sistema software per gestire collezioni di dati
- **data base** (base di dati): una collezioni di dati, può essere gestita da un DBMS

Riepilogo

Come organizzare i dati? **Modello relazionale dei dati**:

- è un modello matematico (astratto)
- permette di descrivere i **componenti** di dati
- la loro **struttura**
- e la loro **organizzazione**

Riepilogo

Come organizzare i dati? **Modello relazionale dei dati**:

- è un modello matematico (astratto)
- permette di descrivere i **componenti** di dati $\boxed{\rightarrow \textbf{relazioni}}$
- la loro **struttura** $\boxed{\rightarrow \textbf{schema} \text{ di una relazione}}$
- e la loro **organizzazione** $\boxed{\rightarrow \text{associazioni basate sui valori}}$

Riepilogo

Come organizzare i dati? **Modello relazionale dei dati**:

- è un modello matematico (astratto)
- permette di descrivere i **componenti** di dati $\boxed{\rightarrow \textbf{relazioni}}$
- la loro **struttura** $\boxed{\rightarrow \textbf{schema} \text{ di una relazione}}$
- e la loro **organizzazione** $\boxed{\rightarrow \text{associazioni basate sui valori}}$

Concetti fondamentali:

- **attributi:** componenti di una relazione, hanno **nome** e **dominio**
- **dominio:** tipo di dati di un attributo
- **schema di una relazione:** insieme di attributi
- **ennupla:** insieme di valori (uno per attributo)
- **istanza di una relazione:** insieme di ennuple

Riepilogo

Un **parallelo con le tabelle**:

- relazione $\leftrightarrow$ tabella
- schema $\leftrightarrow$ intestazione
- attributo $\leftrightarrow$ intestazione di una colonna
- ennupla $\leftrightarrow$ riga
- istanza $\leftrightarrow$ insieme delle righe

Riepilogo

Un **DBMS relazionale** è un software che gestisce collezioni di dati organizzate secondo il modello relazionale.

- **componente base: tabella**
- **tipi di dato elementari** con cui rappresentare i domini
- **DDL** per specificare la struttura di una tabella
- **vincoli di integrità** (di dominio, di ennupla, di chiave, di intertità referenziale)

Riepilogo

Un **DBMS relazionale** è un software che gestisce collezioni di dati organizzate secondo il modello relazionale.

- **componente base: tabella**
- **tipi di dato elementari** con cui rappresentare i domini
- **DDL** per specificare la struttura di una tabella
- **vincoli di integrità** (di dominio, di ennupla, di chiave, di intertità referenziale)

Specifica di una **base di dati relazionale:**

- quali **tabelle**
- quali **colonne** per ogni tabella
- quali **tipi di dato** per ogni colonna
- quali **vincoli per ogni tabella**
- quali **vincoli tra diverse tabelle**

Riepilogo

Schema della base di dati di studenti (semplificata):

Studenti	
Matricola	`CHAR(4)`
Nome	`VARCHAR(50)` `NON NULLO`
Cognome	`VARCHAR(50)` `NON NULLO`
`Chiave pr.: Matricola`	

Esami	
Studente	`CHAR(4)` Chiave est., rif.: Studenti(Matricola)
Corso	`VARCHAR(50)`
Voto	`INTEGER` $\geq 18, \leq 30$ `NON NULLO`
`Chiave pr.: Studente, Corso`	

Riepilogo

Schema della base di dati di studenti (semplificata):

Studenti	
Matricola	`CHAR(4)`
Nome	`VARCHAR(50)` `NON NULLO`
Cognome	`VARCHAR(50)` `NON NULLO`
`Chiave pr.: Matricola`	

Esami	
Studente	`CHAR(4)` Chiave est., rif.: Studenti(Matricola)
Corso	`VARCHAR(50)`
Voto	`INTEGER` $\geq 18, \leq 30$ `NON NULLO`
`Chiave pr.: Studente, Corso`	

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

Riepilogo

Un **vincolo di integrità relazionale:**

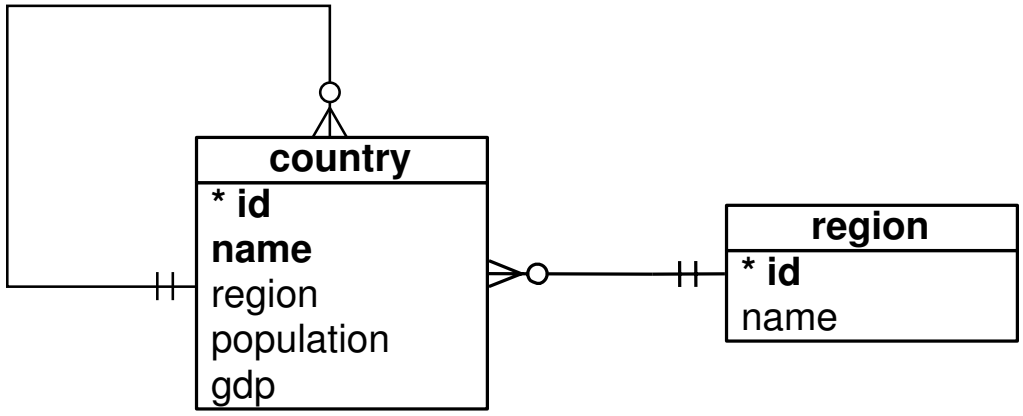
- interessa una tabella interna ed una esterna
- un attributo "bersaglio" ed uno "di riferimento"
- costringe l'attributo bersaglio nella tabella interna ad assumere **uno dei valori dell'attributo di riferimento nella tabella esterna**

Riepilogo

1. **Progetto logico:**
   - **diagramma Entity-Relationship** (E-R): evita di pensare subito in termini di tabelle/relazioni
   - concetti chiave:
     - **entità:** un oggetto su cui vogliamo mantenere informazioni
     - **relazione:** una associazione tra entità
   - risultato del progetto logico = **schema logico**
2. **Progetto fisico:** traduzione da schema logico a schema fisico
   - ogni **entità** corrisponde a:
     - una tabella + relativi tipi di dato
     - i vincoli di dominio, di ennupla, di chiave
   - ogni **relazione** corrisponde a:
     - un attributo o una nuova tabella (colonne = chiavi primarie delle entità collegate)
     - vincoli di integrità referenziale

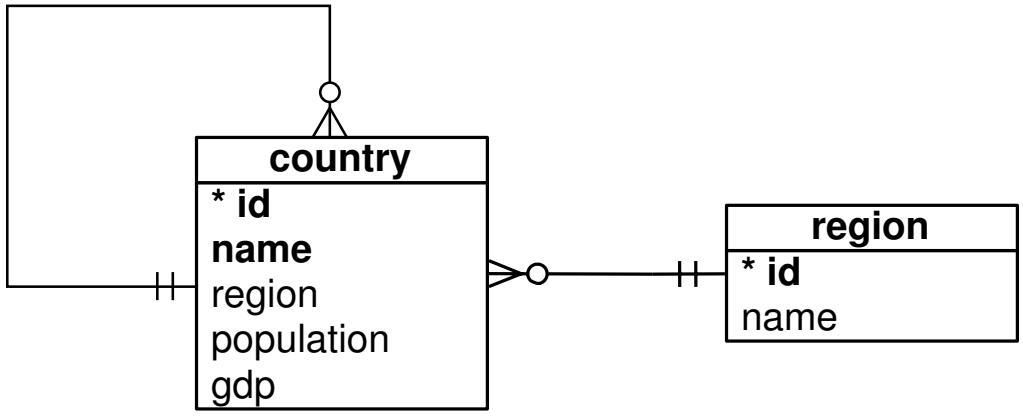
Riepilogo

Progetto logico del DB “Country profiles”:



Riepilogo

Progetto logico del DB “Country profiles”:

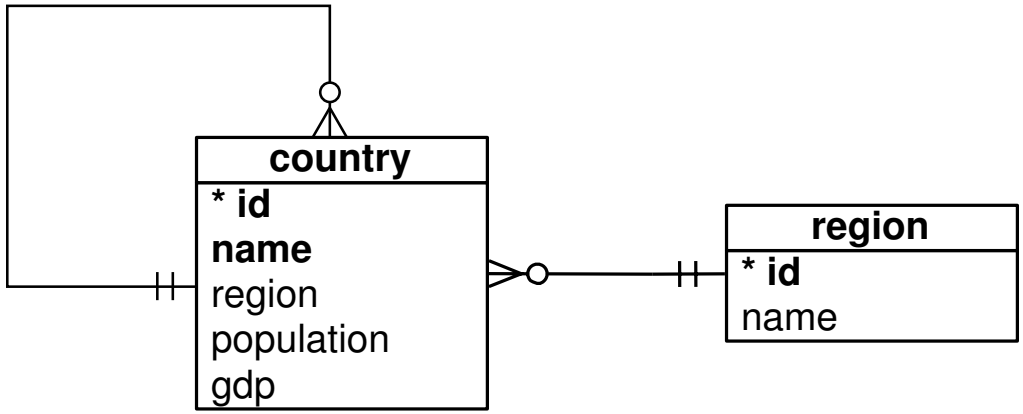


Progetto fisico del DB “Country profiles”:

countries	
id	INTEGER
name	VARCHAR(50) NON NULLO UNICO
region	INTEGER Chiave ext., rif. region(id)
population	INTEGER
gdp	FLOAT
Chiave pr.: id	

Riepilogo

Progetto logico del DB “Country profiles”:



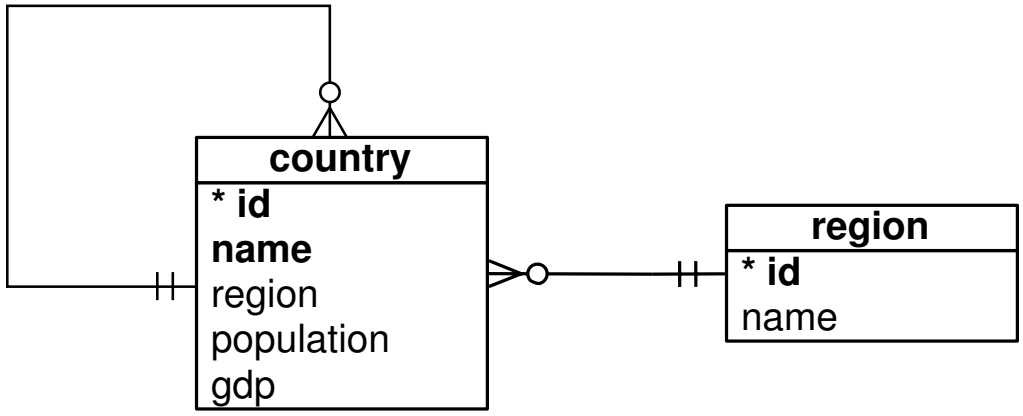
Progetto fisico del DB “Country profiles”:

countries	
id	INTEGER
name	VARCHAR(50) NON NULLO UNICO
region	INTEGER Chiave ext., rif. region(id)
population	INTEGER
gdp	FLOAT
Chiave pr.: id	

regions	
id	INTEGER
name	VARCHAR(50) NON NULLO UNICO
Chiave pr.: id	

Riepilogo

Progetto logico del DB “Country profiles”:



Progetto fisico del DB “Country profiles”:

countries	
id	INTEGER
name	VARCHAR(50) NON NULLO UNICO
region	INTEGER Chiave ext., rif. region(id)
population	INTEGER
gdp	FLOAT
Chiave pr.: id	

regions	
id	INTEGER
name	VARCHAR(50) NON NULLO UNICO
Chiave pr.: id	

borders	
target	INTEGER
neighbor	INTEGER
Chiave pr.: target, neighbor	

Parte II

Outline & Introduzione

Interrogazione e manipolazione di basi di dati

- **Introduzione**
- **Estrarre informazioni da relazioni: algebra relazionale**
  - operatori insiemistici, ridenominazione, proiezione, selezione, join
- **Estrarre informazioni da una Base di Dati relazionale: linguaggio SQL**
  - manipolazione di dati con SQL (accenni)
  - Query SQL
  - funzioni aggregate e raggruppamenti

Parte III

Data Manipulation Language

Interrogazione e manipolazione di basi di dati

Finora abbiamo visto

- come **progettare** una base di dati
- come **realizzare** una base di dati su un DBMS relazionale

Due osservazioni:

1. una base di dati **vuota** non serve a nessuno

$\Rightarrow$ occorre un metodo per **inserire, modificare e cancellare dati**

2. perché una base di dati sia utile occorre poterne estrarre per **estrarne informazioni**

$\Rightarrow$ occorre un metodo per **interrogare la base di dati**

Data Manipulation Language

In un DBMS lo strumento per accedere ai dati è un **Data Manipulation Language (DML)**:

- il DML consente la **modifica** del contenuto delle tabelle
- il DML consente di formulare **interrogazioni** (o **query**)

I DBMS relazioni tipicamente usano lo **Structured Query Language (SQL)**:

- è un DML basato sul modello relazionale dei dati
- è un linguaggio testuale
- tipicamente, gli utenti possono modificare ed accedere ai dati attraverso **maschere**, che spesso non fanno altro che tradurre i click dell'utente in comandi SQL ed eseguirli

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

1. **Inserimento** di una ennupla (codice solo a titolo di esempio)

```
INSERT INTO Esami
VALUES ('0124','Informatica',27)
```

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23
0142	**Informatica**	**27**

1. **Inserimento** di una ennupla (codice solo a titolo di esempio)

```
INSERT INTO Esami
VALUES ('0124','Informatica',27)
```

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23
~~0142~~	~~Informatica~~	~~27~~

1. **Inserimento** di una ennupla: `insert`
2. **Eliminazione** di una ennupla (codice solo a titolo di esempio)

```
DELETE FROM Esami
WHERE Studente='0124' AND Corso='Informatica'
```

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

1. **Inserimento** di una ennupla: `insert`
2. **Eliminazione** di una ennupla (codice solo a titolo di esempio)

```
DELETE FROM Esami
WHERE Studente='0124' AND Corso='Informatica'
```

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

1. **Inserimento** di una ennupla: `insert`
2. **Eliminazione** di una ennupla: `delete`
3. **Modifica** di una ennupla (codice solo a titolo di esempio)

```
UPDATE Esami
SET Voto=26
WHERE Studente='0124' AND Corso='Informatica'
```

Manipolazione dei dati con SQL

- **Inserimento**, **eliminazione** e **aggiornamento** di una ennupla
- Operatori `INSERT`, `DELETE`, `UPDATE`
- vediamo solo **velocemente** alcuni **esempi**:

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	**26**
8392	Geometria	23

1. **Inserimento** di una ennupla: `insert`
2. **Eliminazione** di una ennupla: `delete`
3. **Modifica** di una ennupla (codice solo a titolo di esempio)

```
UPDATE Esami
SET Voto=26
WHERE Studente='0124' AND Corso='Informatica'
```

Parte IV

Algebra relazionale

Interrogazione di una base di dati relazionale

- "quali studenti hanno preso 28 in Informatica?"
- "quali studenti hanno preso almeno un voto superiore a 26 in un esame qualsiasi?"
- "quali esami ha sostenuto Mario Rossi?"

Definizione (Query)

Si chiama **query** *una richiesta precisa di recuperare informazioni ad una base di dati*

In una base di dati relazionale, una query è una **funzione che da un insieme di relazioni ottiene una nuova relazione**.

La seconda definizione è formale (matematica) $\Rightarrow$ **comprensibile da un computer**

Query

Algebra relazionale: insieme di operatori che manipolano relazioni

- algebra tradizionale: uno o più numeri $\mapsto$ un numero
- algebra relazionale: **uno o più relazioni** $\mapsto$ **una relazione**
- Operatori principali:
  - **operatori insiemistici**: $\cup, \setminus, \cap$
  - **ridenominazione**
  - **proiezione**
  - **selezione**
  - **join**

Operatori insiemistici

- Solo su tabelle con gli stessi attributi
- Operatori **binari** di unione, differenza, intersezione

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

Operatori insiemistici

- Solo su tabelle con gli stessi attributi
- Operatori **binari** di unione, differenza, intersezione

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

- **Unione**

Tabella: Stud_Ing $\cup$ Stud_Sci

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
8392	Giulia	Bianchi
0142	Giulia	Bianchi
8279	Fabrizia	Bianchi
0989	Stefano	Neri

Operatori insiemistici

- Solo su tabelle con gli stessi attributi
- Operatori **binari** di unione, differenza, intersezione

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

- Unione
- **Differenza**

Tabella: Stud_Ing $\setminus$ Stud_Sci

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
8392	Giulia	Bianchi

Operatori insiemistici

- Solo su tabelle con gli stessi attributi
- Operatori **binari** di unione, differenza, intersezione

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

- Unione
- Differenza
- **Intersezione**

Tabella: Stud_Ing $\cap$ Stud_Sci

Matricola	**Nome**	**Cognome**
0142	Giulia	Bianchi

Operatore di ridenominazione

- Operatore unario
- Modifica l'**intestazione** di una tabella

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: $\text{Ridenomina}_{(Matricola \rightarrow ID)}(\text{Stud\_Ing})$

ID	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Operatore di proiezione

- Operatore unario di decomposizione verticale
- Seleziona un **sottoinsieme degli attributi**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: $\text{Proietta}_{(Nome, Cognome)}(\text{Stud\_Ing})$

Nome	**Cognome**
Mario	Rossi
Giulia	Bianchi

Operatore di selezione

- Operatore unario di decomposizione orizzontale
- Seleziona un **sottoinsieme delle tuple** in base a una **condizione**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: $\text{Seleziona}_{(Nome=Giulia)}(\text{Stud\_Ing})$

Matricola	**Nome**	**Cognome**
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Operatore di θ-join

- Operatore **binario**
- Serve a **correlare** i dati presenti **in più tabelle**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

Operatore di θ-join

- Esempio:
  - Join tra Studenti ed Esami

Tabella: Studenti Join (Esami)

Matricola	Nome	Cognome	Studente	Corso	Voto
1234	Mario	Rossi	8392	Informatica	28
1234	Mario	Rossi	1234	Informatica	30
1234	Mario	Rossi	0142	Analisi 1	24
1234	Mario	Rossi	8392	Geometria	23
0142	Giulia	Bianchi	8392	Informatica	28
0142	Giulia	Bianchi	1243	Informatica	30
0142	Giulia	Bianchi	0142	Analisi 1	24
0142	Giulia	Bianchi	8392	Geometria	23
8392	Giulia	Bianchi	8392	Informatica	28
8392	Giulia	Bianchi	1234	Informatica	30
8392	Giulia	Bianchi	0142	Analisi 1	24
8392	Giulia	Bianchi	8392	Geometria	23

Operatore di θ-join

- Operatore **binario**
- Serve a **correlare** i dati presenti **in più tabelle**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
1234	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

- Esempio:
  - Join tra Studenti ed Esami
  - Condizione: Studenti.Matricola = Esami.Studente

Tabella: Studenti $\text{Join}_{(Studenti.Matricola=Esami.Studente)}$(Esami)

Matricola	**Nome**	**Cognome**	**Studente**	**Corso**	**Voto**
1234	Mario	Rossi	1234	Informatica	30
0142	Giulia	Bianchi	0142	Analisi 1	24
8392	Giulia	Bianchi	8392	Informatica	28
8392	Giulia	Bianchi	8392	Geometria	23

Esempio di uso degli operatori per eseguire query

Q1. *quali studenti (matricole) hanno preso 28 in Informatica?*

Esempio di uso degli operatori per eseguire query

Q1. *quali studenti (matricole) hanno preso 28 in Informatica?*

- **Proietta**$_{(Studente)}$(
  Seleziona$_{(Voto=28\ e\ Corso=Informatica)}$(Esami)
)

Esempio di uso degli operatori per eseguire query

Q1. *quali studenti (matricole) hanno preso 28 in Informatica?*

- **Proietta**$_{(Studente)}$(
  Seleziona$_{(Voto=28\ e\ Corso=Informatica)}$(Esami)
)

Q2. *quali esami ha sostenuto Mario Rossi?*

Esempio di uso degli operatori per eseguire query

Q1. *quali studenti (matricole) hanno preso 28 in Informatica?*

- **Proietta**$_{(Studente)}$(
  Seleziona$_{(Voto=28\ e\ Corso=Informatica)}$(Esami)
)

Q2. *quali esami ha sostenuto Mario Rossi?*

- **Proietta**$_{(Corso)}$(
  Esami **Join**$_{(Studenti.Matricola=Esami.Studente)}$ (
  Seleziona$_{(Nome=Mario\ e\ Cognome=Rossi)}$(Studenti)
)
)

Parte V

SQL per l'Interrogazione di Basi di Dati Relazionali

SQL per l'Interrogazione di Basi di Dati Relazionali

SQL permette anche (e soprattutto) di **interrogare una base di dati relazionale**

- L'insieme dei costrutti per definire una interrogazione è **modellato sull'algebra relazionale**
- In pratica, ritroviamo (con nomi diversi) **gli stessi operatori**

Costrutto principale: SELECT

```
SELECT ...
FROM <tabella>
```

- Input: una tabella
- Output: una "tabella" (non corrisponde ad una tabella effettivamente memorizzata)

Costrutto SELECT: Proiezione

- si può specificare la lista di colonne da includere nella tabella risultato
- corrisponde all'**operatore di proiezione**

```
SELECT < ‘‘*’’ oppure ‘‘lista di colonne’’>
FROM <tabella>
```

Costrutto SELECT: Proiezione

- si può specificare la lista di colonne da includere nella tabella risultato
- corrisponde all'**operatore di proiezione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *

FROM Studenti
```

Costrutto SELECT: Proiezione

- si può specificare la lista di colonne da includere nella tabella risultato
- corrisponde all'**operatore di proiezione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *

FROM Studenti
```

Risultato:

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Costrutto SELECT: Proiezione

- si può specificare la lista di colonne da includere nella tabella risultato
- corrisponde all'**operatore di proiezione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Matricola

FROM Studenti
```

Costrutto SELECT: Proiezione

- si può specificare la lista di colonne da includere nella tabella risultato
- corrisponde all'**operatore di proiezione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Matricola
FROM Studenti
```

Risultato:

Matricola
1234
0142
8392

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

```
SELECT < ‘‘*’’ oppure ‘‘lista di colonne’’>
FROM <tabella>
WHERE <condizione>
```

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome='Giulia'
```

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome='Giulia'
```

Risultato:

Matricola	**Nome**	**Cognome**
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Termine di una condizione:
`<nome colonna> <operatore> <valore>`
I valori stringa vanno tra apici: `'...'`

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome='Giulia' AND
Matricola='8392'
```

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome='Giulia' AND
Matricola='8392'
```

Risultato:

Matricola	**Nome**	**Cognome**
8392	Giulia	Bianchi

I termini si possono **combinare** mediante: `AND`, `OR`, `NOT`

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Matricola < '8000'
```

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Matricola < '8000'
```

Risultato:

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi

Operatori di confronto (anche per stringhe): `=`, `<`, `>`, `<=`, `>=`

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome LIKE '%rio'
```

Costrutto SELECT: Selezione

- si può specificare un condizione sulle righe da includere
- corrisponde all'**operatore di selezion**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT *
FROM Studenti
WHERE Nome LIKE '%rio'
```

Risultato:

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi

Operatore di pattern matching (solo per stringhe): `LIKE`

Operatori di selezione

=	Uguaglianza
<>	Disuguaglianza
>	Maggioranza
<	Minoranza
>=	Maggiore o uguale
<=	Minore o uguale
`LIKE`	*Pattern* (*espressioni regolari*[1])
`AND`	Congiunzione logica
`OR`	Disgiunzione logica
`IS NULL`	Valore non definito

(1) Uso di *wildcards*: %, _, `[charlist]`, `[!charlist]`

```
SELECT * FROM Stud_Ing
WHERE Cognome LIKE '[bsp]%'
```

Operatori di selezione

=	Uguaglianza
<>	Disuguaglianza
>	Maggioranza
<	Minoranza
>=	Maggiore o uguale
<=	Minore o uguale
`LIKE`	*Pattern* (*espressioni regolari*[1])
`AND`	Congiunzione logica
`OR`	Disgiunzione logica
`IS NULL`	Valore non definito

(1) Uso di *wildcards*: %, _, `[charlist]`, `[!charlist]`

```
SELECT * FROM Stud_Ing
WHERE Cognome LIKE '[!bsp]%'
```

Costrutto SELECT: Ridenominazione

- le colonne selezionate possono essere ridenominate
- corrisponde all'**operatore di ridenominazione**

```
SELECT < ‘‘*’’ oppure ‘‘lista di colonne’’> [AS
<nuovo nome>]
FROM <tabella>
WHERE <condizione>
```

Costrutto SELECT: Ridenominazione

- le colonne selezionate possono essere ridenominate
- corrisponde all'**operatore di ridenominazione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Nome AS Name, Cognome
AS LastName

FROM Studenti
```

Costrutto SELECT: Ridenominazione

- le colonne selezionate possono essere ridenominate
- corrisponde all'**operatore di ridenominazione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Nome AS Name, Cognome
AS LastName

FROM Studenti
```

Risultato:

Name	**LastName**
Mario	Rossi
Giulia	Bianchi
Giulia	Bianchi

Costrutto SELECT: Ridenominazione

- le colonne selezionate possono essere ridenominate
- corrisponde all'**operatore di ridenominazione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

È particolarmente utile se si usano **funzioni di aggregazione**

Query:

```
SELECT COUNT(Matricola) AS
nstudenti
FROM Studenti
```

Costrutto SELECT: Ridenominazione

- le colonne selezionate possono essere ridenominate
- corrisponde all'**operatore di ridenominazione**

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

È particolarmente utile se si usano **funzioni di aggregazione**

Query:

```
SELECT COUNT(Matricola) AS
nstudenti
FROM Studenti
```

Risultato:

nstudenti
3

Funzione di aggregazione: calcola il valore di una poprietà su tutti i valori di uno o più campi

Funzioni di aggregazione

`COUNT`	Numero di valori non nulli[1]
`SUM`	Somma dei valori
`MIN`	Minimo
`MAX`	Massimo
`AVG`	Valore medio
...	

(1) `COUNT(*)` conta le **righe** selezionate

Costrutto SELECT: Operatore di unione

- si possono aggregare is risultati di più SELECT, purché abbiano la **stessa intestazione**
- corrisponde all'**operatore insiemistico di unione**

```
SELECT < ‘‘*’’ oppure ‘‘lista di colonne’’>
FROM <tabella>
UNION
SELECT < ‘‘*’’ oppure ‘‘lista di colonne’’>
FROM <tabella>
```

Costrutto SELECT: Operatore di unione

- si possono aggregare is risultati di più SELECT, purché abbiano la **stessa intestazione**
- corrisponde all'**operatore insiemistico di unione**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

Query:

```
SELECT Nome, Cognome
FROM Stud_Ing
WHERE Matricola = '1234'
UNION
SELECT Nome, Cognome
FROM Stud_Sci
WHERE Matricola = '0142'
```

Costrutto SELECT: Operatore di unione

- si possono aggregare is risultati di più SELECT, purché abbiano la **stessa intestazione**
- corrisponde all'**operatore insiemistico di unione**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

Query:

```
SELECT Nome, Cognome
FROM Stud_Ing
WHERE Matricola = '1234'
UNION
SELECT Nome, Cognome
FROM Stud_Sci
WHERE Matricola = '0142'
```

Risultato:

Name	**LastName**
Mario	Rossi
Giulia	Bianchi

Costrutto SELECT: Operatore di unione

- si possono aggregare is risultati di più SELECT, purché abbiano la **stessa intestazione**
- corrisponde all'**operatore insiemistico di unione**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

Query:

```
SELECT Nome, Cognome
FROM Stud_Ing
WHERE Matricola = '0142'
UNION
SELECT Nome, Cognome
FROM Stud_Sci
WHERE Matricola = '0142'
```

Costrutto SELECT: Operatore di unione

- si possono aggregare is risultati di più SELECT, purché abbiano la **stessa intestazione**
- corrisponde all'**operatore insiemistico di unione**

Tabella: Stud_Ing

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Stud_Sci

Matricola	**Nome**	**Cognome**
8279	Fabrizia	Bianchi
0989	Stefano	Neri
0142	Giulia	Bianchi

Query:

```
SELECT Nome, Cognome
FROM Stud_Ing
WHERE Matricola = '0142'
UNION
SELECT Nome, Cognome
FROM Stud_Sci
WHERE Matricola = '0142'
```

Risultato:

Name	**LastName**
Giulia	Bianchi

Costrutto SELECT: Operatore di join

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

Esempio:

- Join tra Studenti ed Esami
- Condizione: Studenti.Matricola = Esami.Studente

Costrutto SELECT: Operatore di join

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

Costrutto SELECT: Operatore di join

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

```
SELECT *
FROM Studenti
INNER JOIN Esami
ON Studenti.Matricola=Esami.Studente
```

Costrutto SELECT: Operatore di join

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

Risultato:

Matricola	**Nome**	**Cognome**	**Studente**	**Corso**	**Voto**
0142	Giulia	Bianchi	0142	Analisi 1	24
8392	Giulia	Bianchi	8392	Informatica	28
8392	Giulia	Bianchi	8392	Geometria	23

Tipi di join

`INNER JOIN`	θ-join
`LEFT JOIN`	Include tutte le righe della prima tabella (anche senza corrispettivi nella seconda)
`RIGHT JOIN`	Include tutte le righe della seconda tabella (anche senza corrispettivi nella prima)
`FULL JOIN`	Include tutte le righe di tutte le tabelle

LEFT JOIN

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

```
SELECT Matricola, Nome, Cognome, Corso, Voto
FROM Studenti
LEFT JOIN Esami
ON Studenti.Matricola=Esami.Studente
```

Risultato:

Matricola	**Nome**	**Cognome**	**Corso**	**Voto**
1234	Mario	Rossi		
0142	Giulia	Bianchi	Analisi 1	24
8392	Giulia	Bianchi	Informatica	28
8392	Giulia	Bianchi	Geometria	23

RIGHT JOIN

Tabella: Studenti

Matricola	**Nome**	**Cognome**
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	**Corso**	**Voto**
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

```
SELECT Matricola, Nome, Cognome, Corso, Voto
FROM Studenti
RIGHT JOIN Esami
ON Studenti.Matricola=Esami.Studente
```

Risultato:

Matricola	**Nome**	**Cognome**	**Corso**	**Voto**
8392	Giulia	Bianchi	Informatica	28
			Informatica	30
0142	Giulia	Bianchi	Analisi 1	24
8392	Giulia	Bianchi	Geometria	23

FULL JOIN

Tabella: Studenti

Matricola	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Tabella: Esami

Studente	Corso	Voto
8392	Informatica	28
0989	Informatica	30
0142	Analisi 1	24
8392	Geometria	23

```
SELECT Matricola, Nome, Cognome, Corso, Voto
FROM Studenti
FULL JOIN Esami
ON Studenti.Matricola=Esami.Studente
```

Risultato:

Matricola	Nome	Cognome	Corso	Voto
1234	Mario	Rossi		
0142	Giulia	Bianchi	Analisi 1	24
8392	Giulia	Bianchi	Informatica	28
8392	Giulia	Bianchi	Geometria	23
			Informatica	30

Parte VI

Ordinamento e Raggruppamento

Ordinamento

È possibile **ordinare** il risultato di una query:

- clausola `ORDER BY <lista di attributi> ASC | DESC`

Query:

```
SELECT * FROM Studenti
```

Risultato:

Matricola	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT * FROM Studenti
ORDER BY Matricola ASC
```

Ordinamento

È possibile **ordinare** il risultato di una query:

- clausola `ORDER BY <lista di attributi> ASC | DESC`

Query:

```
SELECT * FROM Studenti
```

Risultato:

Matricola	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT * FROM Studenti
ORDER BY Matricola ASC
```

Risultato:

Matricola	Nome	Cognome
0142	Giulia	Bianchi
1234	Mario	Rossi
8392	Giulia	Bianchi

Raggruppamento

È possibile **raggruppare** il risultato di una query:

- clausola `GROUP BY <lista di attributi>`
- in questo caso si possono selezionare solo gli attributi di raggruppamento
- particolamente utile con **funzioni di aggregazione**
- in questo caso la funzione si applica a ciascun gruppo

Query:

```
SELECT * FROM Studenti
```

Risultato:

Matricola	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Cognome, COUNT(*) AS N
FROM Studenti
GROUP BY Cognome ASC
```

Raggruppamento

È possibile **raggruppare** il risultato di una query:

- clausola `GROUP BY <lista di attributi>`
- in questo caso si possono selezionare solo gli attributi di raggruppamento
- particolamente utile con **funzioni di aggregazione**
- in questo caso la funzione si applica a ciascun gruppo

Query:

```
SELECT * FROM Studenti
```

Risultato:

Matricola	Nome	Cognome
1234	Mario	Rossi
0142	Giulia	Bianchi
8392	Giulia	Bianchi

Query:

```
SELECT Cognome, COUNT(*) AS N
FROM Studenti
GROUP BY Cognome ASC
```

Risultato:

Cognome	N
Bianchi	2
Rossi	1