Basi di Dati

SQL-92

Concetti Fondamentali

Concetti Fondamentali

- Introduzione
- Creazione ed eliminazione di bd
- Creazione ed eliminazione di tabelle
- Inserimenti di ennuple
- Interrogazioni
  - clausola SELECT
  - clausola FROM
  - clausola WHERE
  - clausola ORDER BY
  - metodo di scrittura
- Cancellazioni
- Aggiornamenti

Introduzione

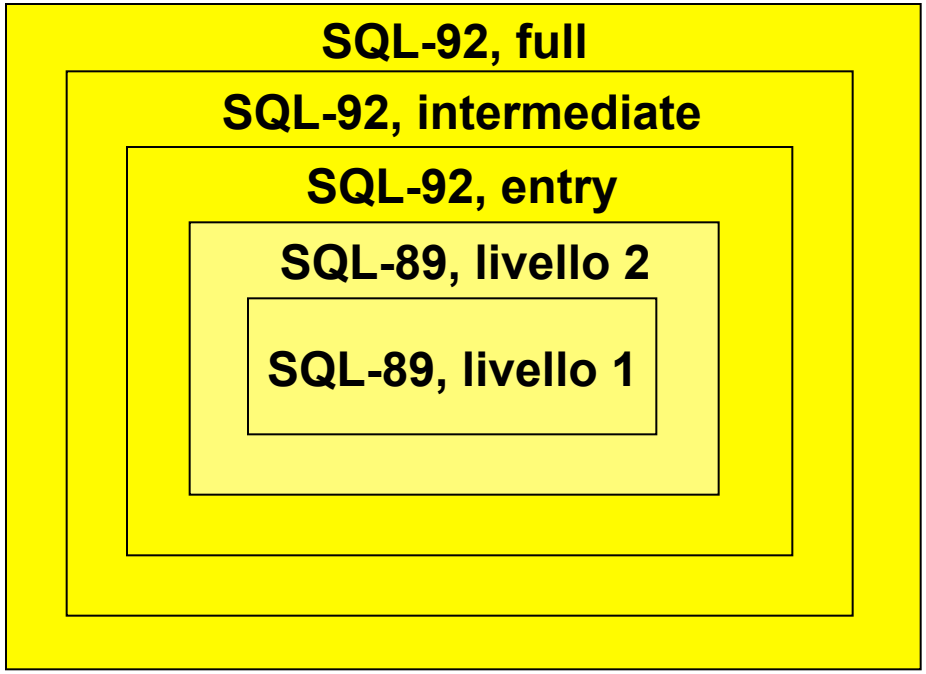
- **SQL ("Structured Query Language")**
  - linguaggio per l'interazione con il DBMS
  - tutte le operazioni vengono specificate in SQL
- **DDL ("Data Definition Language")**
  - creazione degli oggetti dello schema
- **DCL ("Data Control Language")**
  - controllo degli utenti e delle autorizzazioni
- **DML ("Data Manipulation Language")**
  - manipolazione dell'istanza della base di dati (interrogazioni e aggiornamenti)

Storia dello Standard

- Prime implementazioni
  - IBM System/R 1979 (SEQUEL)
- Primi prodotti commerciali
  - IBM SQL/DS, Oracle 1981
- SQL-86
  - prima versione dello standard, basata sul dialetto IBM

Storia dello Standard

- SQL-89 (SQL-1)
  - vincoli di integrità
  - livello1 e livello2
- SQL-92 (SQL-2)
  - entry
  - intermediate
  - full

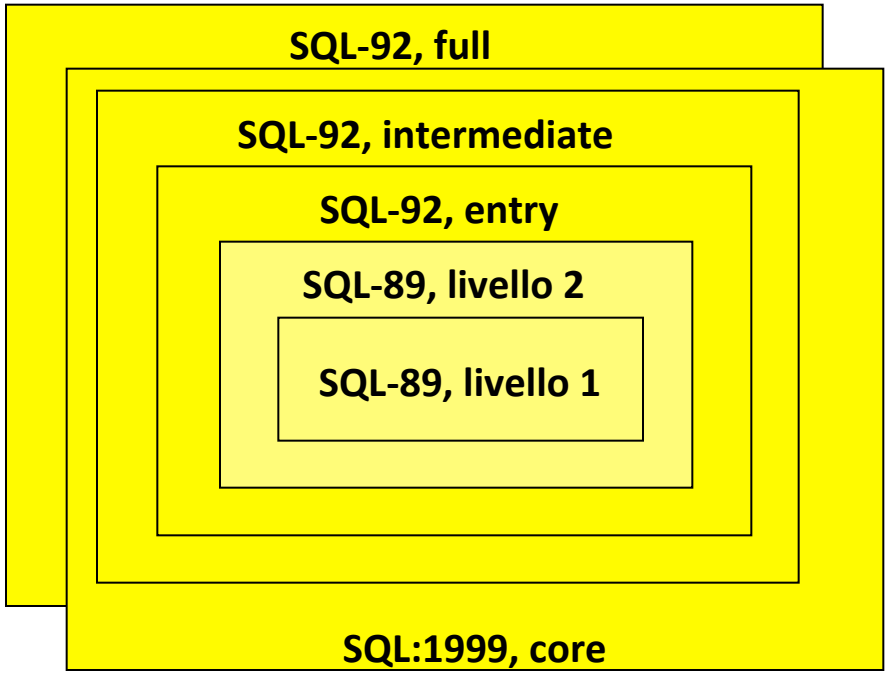


Storia dello Standard

- Standard collegati
- SQL/CLI
  - "Call Level Interface" (ODBC), 1995
- SQL/PSM
  - "Persistent Storage Modules", 1997
- SQL/OLB
  - "Object Language Bindings", 1998

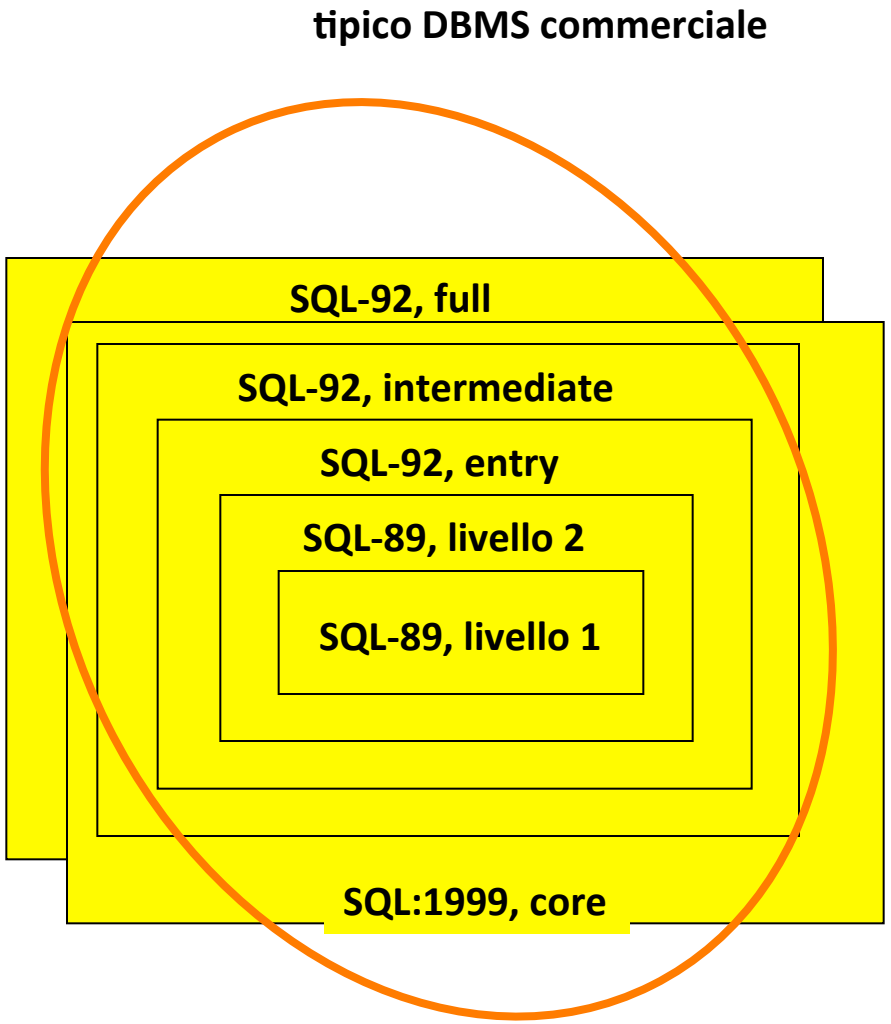
Storia dello Standard

- SQL:1999 (SQL-3)
  - estensioni “object-relational”
  - core: tutto SQL-92 entry, (quasi) tutto SQL-92 intermediate, parte di SQL-92 full
  - packages
- Attualmente:
  - lavori su SQL:200x



Storia dello Standard

- Attualmente
  - SQL-92 intermediate, con funzionalità di full
  - nessuna implementazione competa di SQL-92 full
  - parte di SQL:1999
- Ci concentriamo su
  - SQL-92, intermediate



Creazione ed Eliminazione di BD

- **Istruzioni del DDL**
  - Permettono di definire la base di dati
- **Sintassi**
  - CREATE DATABASE <nome>;
  - DROP DATABASE <nome>;

- **Esempio**
  - **CREATE DATABASE universita;**
  - **DROP DATABASE universita;**

Creazione ed Eliminazione di BD

- **Semantica**
  - CREATE DATABASE
    - Crea una nuova base di dati vuota
    - L'utente deve essere autorizzato
    - L'utente divebta il proprietario della BD
  - DROP DATABASE
    - Elimina una BD esistente anche non vuota!
    - L'utente deve essere autorizzato

Creazione ed Eliminazione di tabelle

- **Istruzioni del DDL**
  - CREATE TABLE
    - definisce uno schema di relazione e crea un'istanza vuota
    - specifica attributi, domini e vincoli
  - DROP TABLE

- **Sintassi**
  - `CREATE TABLE <nome> (<schema>);`
  - `DROP TABLE <nome>;`

Esempio: Tabella Professori

```
CREATE TABLE Professori (
  codice CHAR(4) PRIMARY KEY,
  nome VARCHAR(20) NOT NULL,
  cognome VARCHAR(20) NOT NULL,
  qualifica CHAR(15),
  facolta CHAR(10)
  );

DROP TABLE Professori;
```

Esempio: Tabella Esami

```
CREATE TABLE Esami(
  studente integer
      REFERENCES Studenti(matricola)
      ON DELETE cascade
      ON UPDATE cascade,
  corso CHAR(3) REFERENCES Corsi(cod),
  voto integer,
  lode bool,
  CHECK (voto>=18 and voto<=30),
  CHECK (not lode or voto=30),
  PRIMARY KEY (studente, corso)
  );
```

Creazione ed Eliminazione di tabelle

- **<schema>**
  - una o più definizioni di attributo
  - zero o più definizioni di vincoli di vincoli di tabella
- **Definizione di attributo**
  - `<nome attributo> <tipo> [<vincolo di colonna>]`
  - Vincoli su singolo attributo
- **Definizione di attributo**
  - Normalmente vincoli relativi a più attributi

Definizione di attributo

<nome attributo>

- Identificatore

<tipo>

- **Tipi numerici esatti senza parte frazionaria:**
  - INTEGER,
  - SMALLINT
- **Tipi numerici esatti in base decimale**
  - DECIMAL(lung,dec)
  - NUMERIC
- **Tipo numerici approssimati mediante una rappresentazione in virgola mobile:**
  - REAL
  - FLOAT
  - DOUBLE PRECISION

Definizione di attributo

- **Rappresentare singoli caratteri oppure stringhe**:
  - CHAR(n)
  - VARCHAR(n)
  - LONG VARCHAR, TEXT
- **Rappresentare istanti di tempo:**
  - DATE
  - TIME
  - TIMESTAMP
- **Rappresentare attributi, detti FLAG, che specificano se l'oggetto rappresentato da una tupla possiede o meno quella proprietà.**
  - BINARY(n), BIT(n)
  - VARBINARY(n), VARBIT(n)
  - LONG VARBINARY, BLOB
  - BOOLEAN

Vincoli di Colonna: intrarelazionali

- PRIMARY KEY
  - chiave primaria
  - una sola
  - implica NOT NULL
- UNIQUE: definisce chiavi
- NOT NULL
- REFERENCES <chiave esterna> [ON update CASCADE] [ON delete CASCADE]
  - vincoli di integrità referenziale
- CHECK (<espressione>)

Vincoli di Tabella se su più attributi …

- PRIMARY KEY(<lista attributi>)
- UNIQUE(<lista attributi>)
- FOREIGN KEY (<lista attributi>) REFERENCES <chiave esterna>[ON update CASCADE] [ON delete CASCADE]
  - vincoli di integrità referenziale
- CHECK (<espressione>)

Esempio: Chiave Primaria

```
CREATE TABLE Professori (
   codice CHAR(4) PRIMARY KEY,
   nome VARCHAR(20) NOT NULL,
   cognome VARCHAR(20) NOT NULL,
   qualifica CHAR(15),
   facolta CHAR(10)
   );

CREATE TABLE Esami(
   studente integer
        REFERENCES Studenti(matricola)
        ON DELETE cascade
        ON UPDATE cascade,
   corso CHAR(3),
   voto integer,
   lode bool,
   CHECK (voto>=18 and voto<=30),
   CHECK (not lode or voto=30),
   PRIMARY KEY (studente, corso),
   FOREIGN KEY (corso) REFERENCES Corsi(cod)
   );
```

Esempio: UNIQUE e NOT NULL

```
CREATE TABLE Professori (
   codice CHAR(4) PRIMARY KEY,
   nome VARCHAR(20) NOT NULL,
   cognome VARCHAR(20) NOT NULL,
   qualifica CHAR(15),
   facolta CHAR(10),
   UNIQUE (Cognome,Nome)
   );

CREATE TABLE Esami(
   studente integer
        REFERENCES Studenti(matricola)
        ON DELETE cascade
        ON UPDATE cascade,
   corso CHAR(3) REFERENCES Corsi(cod),
   voto integer,
   lode bool,
   CHECK (voto>=18 and voto<=30),
   CHECK (not lode or voto=30),
   PRIMARY KEY (studente, corso)
   );
```

Questo non è la stessa cosa!

```
Nome    VARCHAR(20) NOT NULL UNIQUE,
Cognome VARCHAR(20) NOT NULL UNIQUE,
```

Esempio: Chiavi esterne

```
CREATE TABLE Professori (
   codice CHAR(4) PRIMARY KEY,
   nome VARCHAR(20) NOT NULL,
   cognome VARCHAR(20) NOT NULL,
   qualifica CHAR(15),
   facolta CHAR(10),
   UNIQUE (Cognome,Nome)
   );

CREATE TABLE Esami(
   studente integer
        REFERENCES Studenti(matricola)
        ON DELETE cascade
        ON UPDATE cascade,
   corso CHAR(3) REFERENCES Corsi(cod),
   voto integer,
   lode bool,
   CHECK (voto>=18 and voto<=30),
   CHECK (not lode or voto=30),
   PRIMARY KEY (studente, corso)
   );
```

ATTENZIONE: L'attributo della tabella esterna a cui si fa riferimento deve essere soggetto a vincolo UNIQUE (o PRIMARY KEY).

```
CREATE TABLE Professori (
   codice char(4) PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   qualifica char(15),
   facolta char(10) );

CREATE TABLE Studenti (
   matricola integer PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   ciclo char(20),
   anno integer,
   relatore char(4)
     REFERENCES Professori(codice)
);
CREATE TABLE Corsi (
   codice char(3) PRIMARY KEY,
   titolo varchar(20) NOT NULL,
   ciclo char(20),
   docente char(4)
     REFERENCES Professori(codice)
);
```

```
CREATE TABLE Tutorato (
   studente integer
     REFERENCES Studenti(matricola),
   tutor integer
     REFERENCES Studenti(matricola),
     PRIMARY KEY (studente,tutor));
CREATE TABLE Esami (
    studente integer
      REFERENCES Studenti(matricola)
      ON DELETE cascade
      ON UPDATE cascade,
    corso char(3)
      REFERENCES Corsi(codice),
    voto integer,
    lode bool,
    CHECK (voto>=18 and voto<=30),
    CHECK (not lode or voto=30),
    PRIMARY KEY (studente, corso));
CREATE TABLE Numeri (
   professore char(4)
     REFERENCES Professori(codice),
   numero char(9),
  PRIMARY KEY (professore,numero));
```

Inserimento

- **Istruzione del DML**
  - INSERT
- **Sintassi**
  - `INSERT INTO <tabella> (<attributi>) VALUES (<valori>);`
- **Semantica**
  - Inserimento della ennupla nella tabella
  - corrispondenza ordinata tra valori e attributi (notazione posizionale)

Esempi di Inserimenti

```
INSERT INTO Professori (cod, cognome, nome,
qualifica, facolta) VALUES ('FT', 'Totti',
'Francesco', 'ordinario', 'Ingegneria');
```

```
INSERT INTO Studenti (matr, cognome, nome, ciclo,
anno, relatore) VALUES (111, 'Rossi', 'Mario',
'laurea tr.', 3, null);
```

```
INSERT INTO Corsi (codice, titolo, ciclo, docente)
VALUES ('PR1', 'Programmazione1', 'laurea tr.',
'FT');
```

Interrogazioni

- **Istruzione del DML**
  - SELECT
  - una o più sottointerrogazioni correlate da operatori insiemistici
- **Filosofia**
  - parzialmente dichiarativa
  - si specificano gli operatori da applicare, non l'ordine in cui devono essere applicati
  - l'ottimizzatore sceglie la strategia ottima
- **Sottointerrogazione**:
  - selezioni
  - proiezioni (con funzioni aggregative)
  - eliminazione di duplicati (DISTINCT)
  - Ridenominazioni
  - ordinamenti finali (ORDER BY)

Interrogazioni

- **Forma standard dell'algebra**
  - una o più sottointerrogazioni
  - correlate da operatori insiemistici
- **Sottointerrogazioni**
  - **strategia a**: prodotti cartesiani tra le tabelle (con eventuali alias)
  - **strategia b**: join tra le tabelle (con eventuali alias)

>>

Interrogazioni

Sottointerrogazione

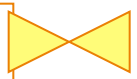
ORDER BY

ρ

DISTINCT

π

σ

X oppure 

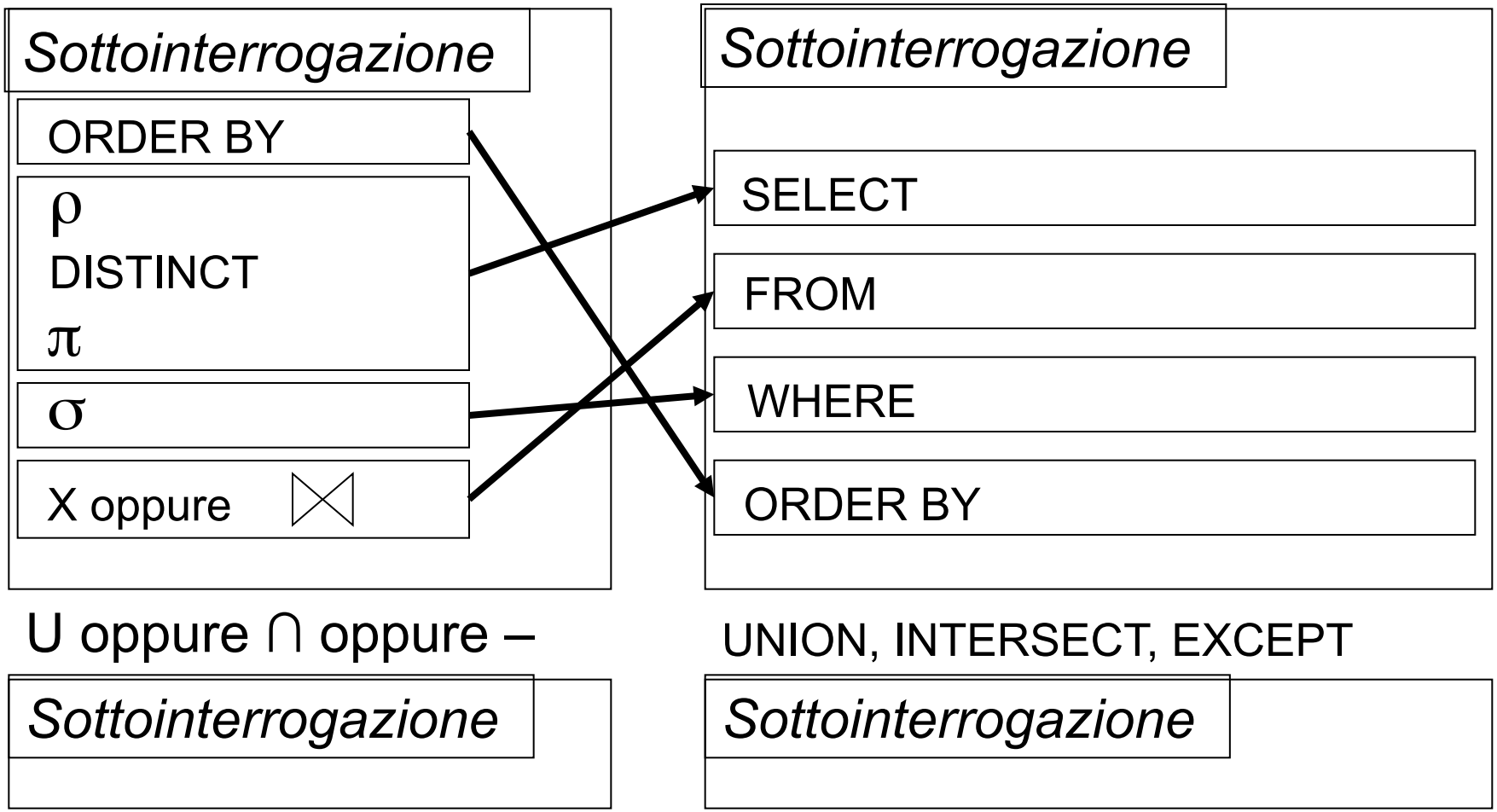
U oppure ∩ oppure –

Sottointerrogazione

Interrogazioni

- **Interrogazioni SQL**
  - una o più sottointerrogazioni correlate da operatori insiemistici
- **Nucleo della SELECT**
  - SELECT: proiezioni, ridenominazioni, distinct
  - FROM: prodotti cartesiani o join, alias
  - [WHERE]: selezioni
- **Clausole aggiuntive**
  - [ORDER BY]: ordinamenti

Interrogazioni



Interrogazione

```
SELECT Lista Attributi
FROM Lista Tabelle
[WHERE Condizione]
```

Intuitivamente:

– Seleziona tra le righe delle tabelle elencate nel `FROM` quelle che soddisfano le condizioni espresse nel `WHERE` e estrae solo gli attributi indicati nel `SELECT`

Clausola `FROM`

- **Funzione**
  - Permette di indicare la tabella o l'insieme delle tabelle su cui eseguire l'interrogazione.

Situazione Semplice: una tabella

- `FROM Studenti`

oppure

- `FROM Studenti AS S`

Proiezione: Clausola SELECT

Funzione

- **estrarre alcune delle colonne di una tabella (PROIEZIONE)**

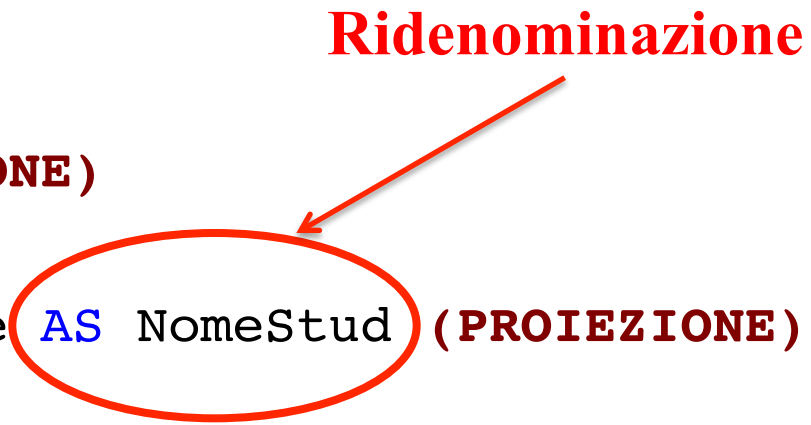
- **SELECT [DISTINCT] <attributi> | ***
  - <attributi>
    - lista di nomi di attributo
    - usare AS per le ridenominazione

Esempio

```
SELECT * (SENZA PROIEZIONE)
```

Oppure

```
SELECT S.Cognome, S.nome AS NomeStud (PROIEZIONE)
```



Proiezione: Clausola `SELECT`

- **Schema del risultato**
  - attributi dello schema originale su cui si effettua la proiezione
- **Istanza del risultato**
  - restrizione ("proiezione") delle ennuple originali agli attributi specificati
- **ATTENZIONE**
  - se nel risultato non sopravvivono chiavi dello schema originale possono esserci duplicati

Esempio di Proiezione

- Estrarre il cognome degli studenti

```
SELECT Cognome
FROM Studenti
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Cognome
Rossi
Neri
Verdi
Rossi
Bruni

Selezione: Clausola `WHERE`

- **Funzione**
  - **Seleziona solo le ennuple che soddisfano la condizione**
- **`WHERE <condizione>`**
- **<condizione>**
  - condizioni di selezione, connettivi booleani

Esempio

```
WHERE S.Cognome='Rossi' AND S.Anno>1
```

Selezione: Clausola `WHERE`

- **Condizioni di selezione**
  - condizioni sui valori degli attributi
  - operatori di confronto =, >, <, >=, <=, <>
  - espressioni con operatori e funzioni
  - connettivi booleani AND, OR, NOT
- **Operatori speciali**
  - IS NULL, IS NOT NULL
  - LIKE

Esempio di Selezione

- Estrarre le informazioni degli studenti che si chiamano Rossi

```
SELECT *
FROM Studenti AS S
WHERE S.Cognome='Rossi'
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
587614	Rossi	Luca	laurea sp.	2	FT
276545	Rossi	Maria	laurea tr.	1	null

Esempio di Selezione

- Estrarre il cognome degli studenti

```
SELECT *
FROM Studenti AS S
WHERE S.Cognome='Rossi' AND S.Anno>1
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
587614	Rossi	Luca	laurea sp.	2	FT

Condizione complessa

```
SELECT *
FROM Studenti
WHERE ciclo = 'laurea tr'
  AND(anno = 1 OR anno = 3)
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
200768	Verdi	Fabio	laurea tr.	3	FT

Condizione "LIKE"

– Le persone che hanno un nome che inizia per 'M' e ha una 'r' come terza lettera

```
SELECT *
FROM Studenti
WHERE nome LIKE 'M_r%'
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
937653	Bruni	Mario	laurea sp.	1	CV

Operatore LIKE

- Operatore LIKE
  - corrispondenza “parziale” tra valori testuali
- “Pattern”
  - sequenza di caratteri e simboli speciali
  - %: una sequenza di 0 o più caratteri
  - _ : un carattere qualsiasi
  - corrisponde ad un insieme di stringhe

Operatore LIKE

- Esempi:
  - ‘B%i’: {‘Bianchi’, ‘Belli’, ‘Brutti’, ‘Bi’, …}
  - ‘p_ _ _a’: {‘palla’, ‘pasta’, ‘pista’, …}
  - ‘A_t%’: {‘Antonio’, ‘Artrite’, …}
- Condizioni
  - <attributo di tipo testo> LIKE <pattern>
  - vera se il valore dell’attributo appartiene all’insieme di stringhe corrispondenti

Gestione Valori Null

– Studenti che non hanno un relatore

```
SELECT *
FROM Studenti
WHERE relatore is NULL
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null

Selezione e Valori Nulli

- Attenzione ai valori nulli
  - **le condizioni sono sempre false in presenza di valori nulli**
  - es: facolta<>‘Ingegneria’ solo le ennuple per cui la facoltà è non nulla e diversa da ing.
  - condizioni speciali: IS NULL, IS NOT NULL
  - es: facolta<>‘Ingegneria’ OR facolta IS NULL tutte le ennuple in cui il valore non è ing.

Esempio: “Professori che Non Sono di Ingegneria”

Professori

cod	cognome	nome	qualifica	facolta
FT	Totti	Francesco	ordinario	Ingegneria
CV	Vieri	Christian	associato	Scienze
ADP	Del Piero	Alessandro	supplente	null

ProfessoriAltreFacolta = $\sigma_{\text{facolta <> 'Ingegneria'}}$ (Professori)

cod	cognome	nome	qualifica	facolta
CV	Vieri	Christian	associato	Scienze

ProfessoriNoIngegneria= $\sigma_{\text{facolta <> 'Ingegneria' OR facolta IS NULL}}$ (Professori)

cod	cognome	nome	qualifica	facolta
CV	Vieri	Christian	associato	Scienze
ADP	Del Piero	Alessandro	supplente	null

```
SELECT *
FROM Professori
WHERE facolta <> 'Ingegneria'
```

```
SELECT *
FROM Professori
WHERE facolta <> 'Ingegneria'
OR facolta IS NULL
```

Proiezione e Duplicati

- La proiezione può generare duplicati
  - nel caso nel risultato non sopravvivano chiavi
- Filosofia dell'SQL (e quindi dell'algebra)
  - l'utente può scegliere se eliminare o meno i duplicati
- Operatore DISTINCT
  - sintassi: DISTINCT (R)
  - semantica: elimina da R i duplicati

DISTINCT

- Il risultato di una selezione può contenere delle ennuple duplicate
- **Esempio**
  - Estrarre i tipi di "ciclo" degli studenti

```
SELECT S.ciclo
FROM Studenti AS S
```

π_{ciclo} (Studenti)

Ciclo
laurea tr.
laurea tr.
laurea tr.
laurea sp.
laurea sp.

```
SELECT DISTINCT S.ciclo
FROM Studenti AS S
```

DISTINCT(π_{ciclo} (Studenti))

Ciclo
laurea tr.
laurea sp.

Esempio: “Cognomi e Anni di Corso degli Studenti”

Studenti

matr	cognome	nome	ciclo	anno	relatore
111	Rossi	Mario	laurea tr.	1	null
222	Neri	Paolo	laurea tr.	2	null
333	Rossi	Maria	laurea tr.	1	null
444	Pinco	Palla	laurea tr.	3	FT
77777	Bruno	Pasquale	laurea sp.	1	FT
88888	Pinco	Pietro	laurea sp.	1	CV

$\pi_{\text{cognome, anno}}$(Studenti)

cognome	anno
Rossi	1
Neri	2
Rossi	1
Pinco	3
Bruno	1
Pinco	1

DISTINCT($\pi_{\text{cognome, anno}}$(Studenti))

cognome	anno
Rossi	1
Neri	2
Pinco	3
Bruno	1
Pinco	1

Ordinamento

- E' possibile gestire anche l'ordinamento
- Operatore dell'algebra ORDER BY
  - sintassi: ORDER BY $_{attributi}$ (R)
  - *attributi*: lista di attributi di R
  - semantica: riordina le ennuple di R utilizzando i valori degli attributi specificati come chiavi di ordinamento (dal primo in avanti) in ordine crescente

ESEMPIO: “Cognomi e Nomi degli Studenti in Ordine”

Studenti

matr	cognome	nome	ciclo	anno	relatore
111	Rossi	Mario	laurea tr.	1	null
222	Neri	Paolo	laurea tr.	2	null
333	Rossi	Maria	laurea tr.	1	null
444	Pinco	Palla	laurea tr.	3	FT
77777	Bruno	Pasquale	laurea sp.	1	FT
88888	Pinco	Pietro	laurea sp.	1	CV

$\text{ORDER BY}_{\text{cognome}}(\pi_{\text{cognome, nome}}(\text{Studenti}))$

cognome	nome
Bruno	Pasquale
Neri	Paolo
Pinco	Pietro
Pinco	Palla
Rossi	Mario
Rossi	Maria

$\text{ORDER BY}_{\text{cognome, nome}}(\pi_{\text{cognome, nome}}(\text{Studenti}))$

cognome	nome
Bruno	Pasquale
Neri	Paolo
Pinco	Palla
Pinco	Pietro
Rossi	Maria
Rossi	Mario

Clausola ORDER BY

- **Funzione**
  - Ordina le tuple in modo discendente o ascendente
- **ORDER BY <attributi>**
  - **<attributi>**
    - lista di attributi
    - <attributo> {ASC | DESC}

Esempio

```
ORDER BY S.Cognome ASC, S.Nome DESC
ORDER BY S.Cognome DESC, S.Nome DESC
ORDER BY S.Cognome, S.Nome
```

(default ASC)

- **Semantica**
  - Ordinamento delle tuple

Esempio di ORDER BY

- Estrarre le informazioni degli studenti che si chiamano Rossi

```
SELECT *
FROM Studenti AS S
WHERE S.Cognome='Rossi'
ORDER BY S.Cognome, S.Nome DESC
```

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
587614	Rossi	Luca	laurea sp.	2	FT

Clausola `FROM: JOIN`

- E' nella natura del modello relazionale frammentare i dati tra le tabelle
- molto spesso è necessario **correlare dati provenienti da tabelle diverse**
- è possibile utilizzare il **prodotto cartesiano**
- Sul prodotto cartesiano si applicano le condizioni del WHERE indicando quelle che esprimono il legame tra le tabelle per esprime il JOIN

Esempio di `JOIN`

STUDENTI

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

PROFESSOR

Codice	Cognome	Nome	Qualifica	Facolta
FT	Monreale	Anna	Ricercatore	Informatica
CV	Tesconi	Maurizio	Ricercatore	Ingegneria

Matricola	Cognome	Nome	Ciclo	Anno	Relatore	Codice	Cognome	Nome	Qualifica	Facolta
276545	Rossi	Maria	laurea tr.	1	null	FT	Monreale	Anna	Ricercatore	Informatica
276545	Rossi	Maria	laurea tr.	1	null	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
485745	Neri	Anna	laurea tr.	2	null	FT	Monreale	Anna	Ricercatore	Informatica
485745	Neri	Anna	laurea tr.	2	null	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
200768	**Verdi**	**Fabio**	**laurea tr.**	**3**	**FT**	**FT**	**Monreale**	**Anna**	**Ricercatore**	**Informatica**
200768	Verdi	Fabio	laurea tr.	3	FT	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
587614	**Rossi**	**Luca**	**laurea sp.**	**2**	**FT**	**FT**	**Monreale**	**Anna**	**Ricercatore**	**Informatica**
587614	Rossi	Luca	laurea sp.	2	FT	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
937653	Bruni	Mario	laurea sp.	1	CV	FT	Monreale	Anna	Ricercatore	Informatica
937653	**Bruni**	**Mario**	**laurea sp.**	**1**	**CV**	**CV**	**Tesconi**	**Maurizio**	**Ricercatore**	**Ingegneria**

Esempio di JOIN

STUDENTI

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

PROFESSOR

Codice	Cognome	Nome	Qualifica	Facolta
FT	Monreale	Anna	Ricercatore	Informatica
CV	Tesconi	Maurizio	Ricercatore	Ingegneria

Matricola	Cognome	Nome	Ciclo	Anno	Relatore	Codice	Cognome	Nome	Qualifica	Facolta
276545	Rossi	Maria	laurea tr.	1	null	FT	Monreale	Anna	Ricercatore	Informatica
276545	Rossi	Maria	laurea tr.	1	null	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
485745	Neri	Anna	laurea tr.	2	null	FT	Monreale	Anna	Ricercatore	Informatica
485745	Neri	Anna	laurea tr.	2	null	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
200768	**Verdi**	**Fabio**	**laurea tr.**	**3**	**FT**	**FT**	**Monreale**	**Anna**	**Ricercatore**	**Informatica**
200768	Verdi	Fabio	laurea tr.	3	FT	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
587614	**Rossi**	**Luca**	**laurea sp.**	**2**	**FT**	**FT**	**Monreale**	**Anna**	**Ricercatore**	**Informatica**
587614	Rossi	Luca	laurea sp.	2	FT	CV	Tesconi	Maurizio	Ricercatore	Ingegneria
937653	Bruni	Mario	laurea sp.	1	CV	FT	Monreale	Anna	Ricercatore	Informatica
937653	**Bruni**	**Mario**	**laurea sp.**	**1**	**CV**	**CV**	**Tesconi**	**Maurizio**	**Ricercatore**	**Ingegneria**

Esempio di JOIN

STUDENTI

Matricola	Cognome	Nome	Ciclo	Anno	Relatore
276545	Rossi	Maria	laurea tr.	1	null
485745	Neri	Anna	laurea tr.	2	null
200768	Verdi	Fabio	laurea tr.	3	FT
587614	Rossi	Luca	laurea sp.	2	FT
937653	Bruni	Mario	laurea sp.	1	CV

PROFESSOR

Codice	Cognome	Nome	Qualifica	Facolta
FT	Monreale	Anna	Ricercatore	Informatica
CV	Tesconi	Maurizio	Ricercatore	Ingegneria

Matricola	Cognome	Nome	Ciclo	Anno	Relatore	Codice	Cognome	Nome	Qualifica	Facolta
200768	Verdi	Fabio	laurea tr.	3	FT	FT	Monreale	Anna	Ricercatore	Informatica
587614	Rossi	Luca	laurea sp.	2	FT	FT	Monreale	Anna	Ricercatore	Informatica
937653	Bruni	Mario	laurea sp.	1	CV	CV	Tesconi	Maurizio	Ricercatore	Ingegneria

Clausola FROM

- **Strategia a**

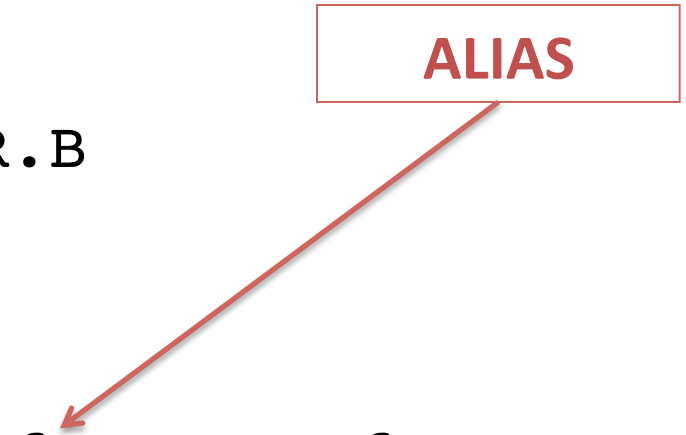
```
FROM R,S
WHERE S.A=R.B
```

- **Strategia b**

```
FROM S JOIN R ON S.A=R.B
```

- **Strategia c**

```
FROM S AS p JOIN R AS f ON p.A=f.B
```



Alias

- **Necessario quanto la stessa tabella può essere coinvolta più di una volta nello stesso join**

- Esempio
  - "Cognomi e nomi degli studenti che all'esame di Programmazione hanno riportato un voto superiore a quello dei loro tutor"

- ATTENZIONE
  - si tratta di una interrogazione molto complessa

Alias

- In questo caso
  - devo confrontare il voto dello studente nell'esame di programmazione con quello del tutor
  - entrambe le informazioni vengono dalla tabella Studenti (che deve necessariamente essere usata due volte)
  - problema con i nomi: come distinguo l'attributo che corrisponde al voto dello studente da quello del tutor?

Alias

- Operatore di Alias per una Tabella
  - crea una copia di una tabella esistente
  - con un nome diverso (e quindi risolve il problema del nome degli attributi)
- Sintassi
  - R AS T
- Semantica
  - l'istanza di T è identica all'istanza di R
  - nello schema di T, l'attributo R.A assume il nome T.A

Alias

- **“Cognomi e nomi degli studenti che all’esame di Programmazione hanno riportato un voto superiore a quello dei loro tutor”**
  - Studenti, per i dati degli studenti
  - Esami, per i dati sugli esami degli studenti
  - Tutorato, per le relazioni tra studenti e tutor
  - Esami di nuovo, per i dati sugli esami sostenuti dai tutor; è necessario un alias (Esami as EsamiTutor)

“Studenti, Voti e Tutor”

- Strategia
  - I Passo: tabella StudentiVoti, join tra Studenti ed Esami
  - II Passo: tabella StudentiVotiTutor, join tra StudentiVoti e Tutorato
  - III Passo: tabella StudentiVotiTutorVoti, join tra StudentiVotiTutor e Esami AS EsamiTutor

“Studenti, Voti e Tutor”

- I Passo: join tra Studenti ed Esami

StudentiVoti= Studenti $\bowtie_{\text{matr=studente}}$ Esami

```
TABLE StudentiVoti (
   Studenti.matr integer,
   Studenti.cognome varchar(20),
   Studenti.nome varchar(20),
   Studenti.ciclo char(20),
   Studenti.anno integer,
   Studenti.relatore char(4),
   Esami.studente integer
   Esami.corso char(3)
   Esami.voto integer,
   Esami.lode bool);
```

“Studenti, Voti e Tutor”

- Il Passo: join con Tutorato

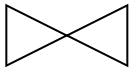
StudentiVotiTutor= StudentiVoti $\bowtie_{matr=Tutorato.studente}$ Tutorato

```
TABLE StudentiVotiTutor (
   Studenti.matr integer,
   Studenti.cognome varchar(20),
   Studenti.nome varchar(20),
   Studenti.ciclo char(20),
   Studenti.anno integer,
   Studenti.relatore char(4),
   Esami.studente integer
   Esami.corso char(3)
   Esami.voto integer,
   Esami.lode bool,
   Tutorato.studente integer,
   Tutorato.tutor integer);
```

“Studenti, Voti e Tutor”

- III Passo: join con Esami AS EsamiTutor

StudentiVotiTutorEsamitutor = StudentiVotiTutor

 Tutorato.tutor=EsamiTutor.studente (Esami AS EsamiTutor)

```
TABLE StudentiVotiTutorEsamitutor (
   Studenti.matr integer,
   Studenti.cognome varchar(20),
   Studenti.nome varchar(20),
   Studenti.ciclo char(20),
   Studenti.anno integer,
   Studenti.relatore char(4),
   Esami.studente integer
   Esami.corso char(3)
   Esami.voto integer,
   Esami.lode bool,
   Tutorato.studente integer,
   Tutorato.tutor integer,
   EsamiTutor.studente integer
   EsamiTutor.corso char(3)
   EsamiTutor.voto integer,
   EsamiTutor.lode bool);
```

“Studenti, Voti e Tutor”

StudentiVotiTutorEsamitutor

Stud.matr	Studenti. cognome	.	Esami. studente	Esami. corso	Esami. voto	.	Tutorato. studente	Tutorato. tutor	ET. studente	ET. Corso	ET. voto	.
111	Rossi	.	111	PR1	27	.	111	77777	77777	PR1	21	.
111	Rossi	.	111	INFT	24	.	111	77777	77777	PR1	21	.
222	Neri	.	222	ASD	30	.	222	77777	77777	PR1	21	.
111	Rossi	.	111	PR1	27	.	111	77777	77777	ASD	20	.
111	Rossi	.	111	INFT	24	.	111	77777	77777	ASD	20	.
222	Neri	.	222	ASD	30	.	222	77777	77777	ASD	20	.

tutte le possibili coppie fatte di
un esame di uno studente ed un esame del suo tutor

“Studenti, Voti e Tutor”

- Selezioni e proiezioni finali

Risultato = π cognome, nome (

σ Esami.corso=‘Pr1’ AND EsamiTutor.corso=‘Pr1’ AND Esami.voto > EsamiTutor.voto (

StudentiVotiTutorEsamitutor)

```
TABLE Risultato (
   Studenti.cognome varchar(20),
   Studenti.nome varchar(20));
```

“Studenti, Voti e Tutor”

- Riassumendo

$$\text{Risultato} = \pi_{\text{cognome, nome}} \Big(\sigma_{\text{Esami.corso='Pr1' AND EsamiTutor.corso='Pr1' AND Esami.voto > EsamiTutor.voto}} \Big(\text{Studenti} \bowtie_{\text{matr=studente}} \text{Esami} \bowtie_{\text{matr=Tutorato.studente}} \text{Tutorato} \bowtie_{\text{Tutorato.tutor=EsamiTutor.studente}} (\text{Esami AS EsamiTutor}) \Big) \Big)$$

Soluzione SQL

```
SELECT Cognome, nome
FROM STUDENTI JOIN ESAMI ON matr=studente
     JOIN Tutorato ON Tutorato.studente=matr
     JOIN ESAMI AS EsamiTutor ON
     Tutorato.tutor=EsamiTutor.studente
WHERE Esami.corso='Pr1' AND
      EsamiTutor.corso='Pr1' AND
      Esami.voto > EsamiTutor.voto
```

Operatori insiemistici

- Operatori binari

- `Union`: R U S
- `Intersect`: R ∩ S
- `EXCEPT`: R – S

Operatori Insiemistici

- Si applicano solo in alcuni casi
  - le tabelle R ed S devono avere lo stesso numero di attributi
- **Associazione posizionale**
  - gli attributi devono avere ordinatamente lo stesso tipo
- **Schema del risultato**
  - eredita i nomi degli attributi dalla prima tabella
- **Attenzione**
  dal risultato degli operatori insiemistici vengono eliminati eventuali duplicati

Unione

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati ∪ Specialisti

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45
9297	Neri	33

```
SELECT *
FROM Laureati

UNION

SELECT *
FROM Specialisti
```

Intersezione

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati ∩ Specialisti

Matricola	Nome	Età
7432	Neri	54
9824	Verdi	45

```
SELECT *
FROM Laureati

INTERSECT

SELECT *
FROM Specialisti
```

Intersezione

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	trenta
7432	Neri	venti
9824	Verdi	quaranta

```
SELECT *
FROM Laureati

INTERSECT

SELECT *
FROM Specialisti
```

```
SELECT matricola, nome
FROM Laureati

INTERSECT

SELECT matricola, nome
FROM Specialisti
```

Intersezione IN

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati ∩ Specialisti

Matricola	Nome	Età
7432	Neri	54
9824	Verdi	45

```
SELECT *
FROM Laureati AS L
WHERE L.Matricola IN (

SELECT S.Matricola
FROM Specialisti AS S
)
```

Differenza

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati – Specialisti

Matricola	Nome	Età
7274	Rossi	42

```
SELECT *
FROM Laureati

EXCEPT

SELECT *
FROM Specialisti
```

Differenza NOT IN

Laureati

Matricola	Nome	Età
7274	Rossi	42
7432	Neri	54
9824	Verdi	45

Specialisti

Matricola	Nome	Età
9297	Neri	33
7432	Neri	54
9824	Verdi	45

Laureati – Specialisti

Matricola	Nome	Età
7274	Rossi	42

```
SELECT *
FROM Laureati AS L
WHERE L.matricola NOT IN(

SELECT S.matricola
FROM Specialisti
)
```

Esempi

- "Studenti della laurea triennale di anni successivi al primo"

Risultato = $\sigma_{\text{ciclo='laurea tr.' AND anno>1}}$ (Studenti)

```
SELECT *
FROM Studenti
WHERE ciclo='laurea tr.' AND anno>1;
```

Esempi

- "Cognomi e nomi degli studenti"

$$\text{ElencoNomi} = \text{DISTINCT}\,(\pi_{\text{cognome, nome}}(\text{Studenti}))$$

```
SELECT DISTINCT cognome, nome
FROM Studenti;
```

Esempi

- "Cognomi e nomi degli studenti, in ordine alfabetico"

$$\text{ElencoNomi} = \text{ORDER BY}_{\text{cognome, nome}}(\text{DISTINCT}(\pi_{\text{cognome, nome}}(\text{Studenti})))$$

```
SELECT DISTINCT cognome, nome
FROM Studenti
ORDER BY cognome, nome;
```

Esempi

- “Cognomi, nomi e numeri di telefono dei professori” (strategia a)

$$\text{ProfessoriENumeri} = \pi_{\text{cognome, nome, numero}} (\sigma_{\text{codice=professore}} (\text{Professori X Numeri}))$$

```
SELECT cognome, nome, numero
FROM Professori, Numeri
WHERE codice=professore;
```

Esempi

- "Cognomi, nomi e numeri di telefono dei professori" (strategia b)

$$\text{ProfessoriENumeri} = \pi_{\text{cognome, nome, numero}} (\text{Professori} \bowtie_{\text{codice=professore}} \text{Numeri})$$

```
SELECT cognome, nome, numero
FROM Professori JOIN Numeri
     ON codice=professore;
```

Esempi

- "Matricola e cognome degli studenti che hanno sostenuto l'esame di informatica teorica"

$$\text{Risultato} = \pi_{\text{matricola, cognome}} (\sigma_{\text{titolo='Inform. t.'}} (\text{Studenti} \bowtie_{\text{matr=studente}} \text{Esami} \bowtie_{\text{cod=corso}} \text{Corsi}))$$

```
SELECT matricola, cognome
FROM Studenti JOIN Esami ON matricola=studente
              JOIN Corsi ON codice=corso
WHERE titolo='Inform. t.';
```

Esempi

- “Cognomi e nomi degli studenti che all’esame di Programmazione hanno riportato un voto superiore a quello dei loro tutor”

- Tabelle coinvolte
  - Studenti, Esami
  - Tutorato, Esami AS EsamiTutor

Esempi

- "Cognome e nome delle persone"

$$\text{Risultato} = \rho_{\text{cognome AS cognomePersona, nome AS nomePersona}} (\pi_{\text{cognome, nome}} (\text{Professori})) \cup \pi_{\text{cognome, nome}} (\text{Studenti})$$

```
SELECT cognome AS cognomePersona, nome AS nomePersona
FROM Professori
UNION
SELECT cognome, nome
FROM Studenti;
```

Esempi

- "Studenti e tutor" (continua)

$$\text{Risultato} = \pi_{\text{cognome, nome}} (\sigma_{\text{Esami.corso='Pr1' AND EsamiTutor.corso='Pr1' AND Esami.voto > EsamiTutor.voto}} (\text{Studenti} \bowtie_{\text{matr=studente}} \text{Esami} \bowtie_{\text{matr=Tutorato.studente}} \text{Tutorato} \bowtie_{\text{Tutorato.tutor=EsamiTutor.studente}} (\text{Esami AS EsamiTutor})))$$

Esempi

- “Studenti e tutor” (continua)

```
SELECT cognome, nome
FROM Studenti JOIN Esami ON matr=studente
     JOIN Tutorato ON matr=Tutorato.studente
     JOIN Esami AS EsamiTutor
          ON Tutorato.tutor = EsamiTutor.studente
WHERE Esami.corso='Pr1' AND EsamiTutor.corso='Pr1'
          AND Esami.voto > EsamiTutor.voto;
```

Esempi

- “Cognome e nome dei professori ordinari che non supervisionano tesi triennali”

$$\text{Risultato} = \rho_{\text{cognome AS cognomeProf, nome AS nomeProf}} (\pi_{\text{cognome, nome}} (\sigma_{\text{qualifica = 'Ordinario'}} (\text{Professori})) - \pi_{\text{cognome, nome}} (\sigma_{\text{ciclo = 'laurea tr.'}} (\text{Studenti} \bowtie_{\text{relatore = cod}} \text{Professori})))$$

Esempi

- “Cognome e nome dei professori ordinari che non supervisionano tesi triennali”

```
SELECT cognome AS cognomeProf, nome AS nomeProf
FROM Professori
WHERE qualifica='ordinario'

EXCEPT

SELECT Relatore.cognome, Relatore.nome
FROM Studenti JOIN Professori AS Relatore
      ON relatore=cod
WHERE ciclo='laurea tr.';
```

Esempi

- “Cognome e nome dei professori ordinari che non supervisionano tesi triennali”

```
SELECT cognome AS cognomeProf, nome AS nomeProf
FROM Professori
WHERE qualifica='ordinario' AND cod NOT IN (

SELECT relatore, Relatore.cognome, Relatore.nome
FROM Studenti JOIN Professori AS Relatore
      ON relatore=cod
WHERE ciclo='laurea tr.');
```

Forma Standard Completa

- Per ogni sottointerrogazione
  - eventuali selezioni
  - eventuali proiezioni, con eventuali funzioni aggregative
  - eventuale eliminazione dei duplicati
  - eventuali ridenominazioni
  - eventuali riordinamenti

Interrogazioni: Metodo di Scrittura

- Scrivere l'interrogazione in algebra relazionale utilizzando la forma standard
- Tradurre gli operatori nella sintassi di SQL
- Stabilire se sono necessari operatori insiemistici
  - dividere in sottointerrogazioni
- Per ogni sottointerrogazione
  - decidere da quali tabelle prelevare i dati
- Decidere eventuali alias
- Se le tabelle sono più di una,
  - strategia a) prodotti cartesiani oppure
  - strategia b) join con le condizioni opportune

Interrogazioni: Metodo di Scrittura (2)

- Scrivere le eventuali selezioni
  - strategia a) incluse le condizioni di Join
- Scrivere le eventuali proiezioni
  - e le eventuali funzioni aggregative
- Scrivere le eventuali eliminazione di duplicati
- Scrivere le eventuali ridenominazioni finali
- Scrivere gli eventuali operatori di ordinamento
- Rimettere le sottointerrogazioni insieme

Metodo di Scrittura Completo

- Suggerimento
  - dividere la scrittura dell'interrogazione in passi, producendo ad ogni passo un risultato intermedio
  - ragionare sulla struttura del risultato intermedio (schema, ovvero attributi, e istanza, ovvero numero e natura delle ennuple)

Cancellazioni Ennuple

- **Istruzione del DML**
  - DELETE
- **Sintassi**
  - `DELETE FROM <tabella> [<clausola WHERE>];`
  - `<clausola WHERE>:` identica alla clausola `WHERE` di una interrogazione
- **Semantica**
  - Elimina dalla tabella tutte le ennuple (che soddisfano la condizione)

Esempi Cancellazioni

```
DELETE FROM Studenti
   WHERE matr=111;

DELETE FROM Studenti
   WHERE ciclo='laurea tr.' AND docente='FT';
```

Aggiornamenti Ennuple

- **Istruzione del DML**
  - UPDATE
- **Sintassi**
  - `UPDATE <tabella> SET <attributo>=<espressione> [<clausola WHERE>]`
- **Semantica**
  - aggiorna il valore dell'attributo di tutte le ennuple (che soddisfano la condizione)

Esempi Aggiornamenti

```
UPDATE Studenti SET anno=anno+1;

UPDATE Studenti SET matr=11111
   WHERE matr=111;

UPDATE Studenti SET docente='VC'
   WHERE ciclo='laurea tr.' AND docente='FT';
```

Modifiche Schema

- **Istruzione del DML**
  - ALTER
- **Sintassi**
  - ALTER TABLE <tabella> <

    alter COLUMN NomeAttributo <SET default NuovoDefault | drop default > |

    add CONSTRAINT defVincolo |

    drop CONSTRAINT NomeVincolo |

    add COLUMN DefAttributo |

    drop COLUMN NomeAttributo >
- **Semantica**
  - Modifica delle varie componenti dello schema

Esempio

- ALTER TABLE Studenti ADD COLUMN dataNascita DATE;
- ALTER TABLE Studenti ADD COLUMN luogoNascita VARCHAR(20);
- ALTER TABLE Studenti ADD COLUMN reddito DECIMAL(8,2);
- ALTER TABLE Studenti DROP COLUMN dataDiNascita;

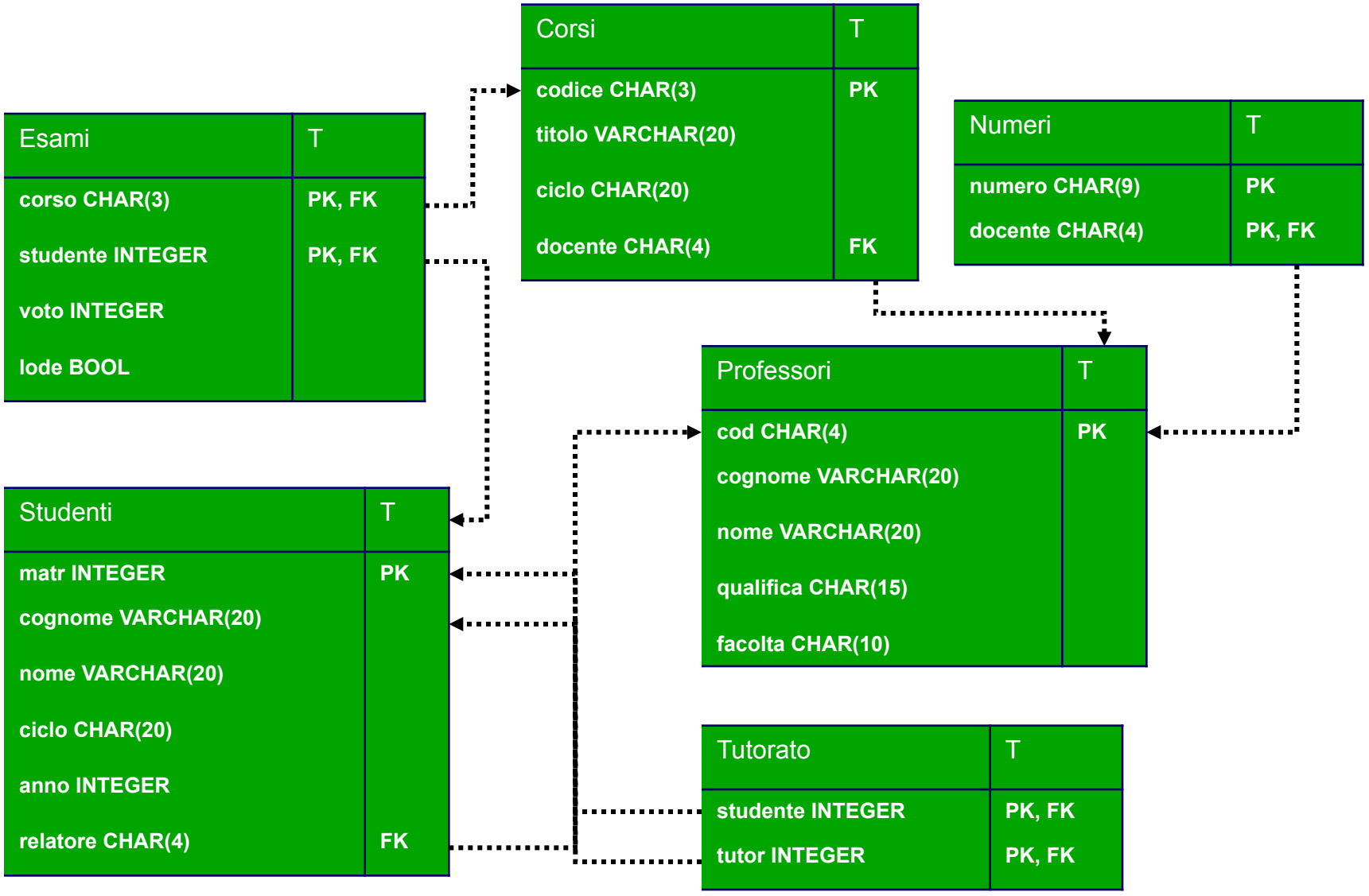
Concetti Fondamentali

- Introduzione
- Creazione ed eliminazione di bd
- Creazione ed eliminazione di tabelle
- Inserimenti di ennuple
- Interrogazioni
  - clausola SELECT
  - clausola FROM
  - clausola WHERE
  - clausola ORDER BY
  - metodo di scrittura
- Cancellazioni
- Aggiornamenti

```
CREATE TABLE Professori (
   cod char(4) PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   qualifica char(15),
   facolta char(10) );

CREATE TABLE Studenti (
   matr integer PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   ciclo char(20),
   anno integer,
   relatore char(4)
     REFERENCES Professori(cod)
);
CREATE TABLE Corsi (
   cod char(3) PRIMARY KEY,
   titolo varchar(20) NOT NULL,
   ciclo char(20),
   docente char(4)
     REFERENCES Professori(cod)
);
```

```
CREATE TABLE Tutorato (
   studente integer
     REFERENCES Studenti(matr),
   tutor integer
     REFERENCES Studenti(matr),
     PRIMARY KEY (studente,tutor));
CREATE TABLE Esami (
    studente integer
      REFERENCES Studenti(matr)
      ON DELETE cascade
      ON UPDATE cascade,
    corso char(3)
      REFERENCES Corsi(cod),
    voto integer,
    lode bool,
    CHECK (voto>=18 and voto<=30),
    CHECK (not lode or voto=30),
    PRIMARY KEY (studente, corso));
CREATE TABLE Numeri (
   professore char(4)
     REFERENCES Professori(cod),
   numero char(9),
  PRIMARY KEY (professore,numero));
```


Corsi
T
codice CHAR(3)
PK
titolo VARCHAR(20)
ciclo CHAR(20)
docente CHAR(4)
FK
Esami
T
corso CHAR(3)
PK, FK
studente INTEGER
PK, FK
voto INTEGER
lode BOOL
Numeri
T
numero CHAR(9)
PK
docente CHAR(4)
PK, FK
Professori
T
cod CHAR(4)
PK
cognome VARCHAR(20)
nome VARCHAR(20)
qualifica CHAR(15)
facolta CHAR(10)
Studenti
T
matr INTEGER
PK
cognome VARCHAR(20)
nome VARCHAR(20)
ciclo CHAR(20)
anno INTEGER
relatore CHAR(4)
FK
Tutorato
T
studente INTEGER
PK, FK
tutor INTEGER
PK, FK

Professori

cod	cognome	nome	qualifica	facolta
FT	Totti	Francesco	ordinario	Ingegneria
CV	Vieri	Christian	associato	Scienze
ADP	Del Piero	Alessandro	supplente	null

Studenti

matr	cognome	nome	ciclo	anno	relatore
111	Rossi	Mario	laurea tr.	1	null
222	Neri	Paolo	laurea tr.	2	null
333	Rossi	Maria	laurea tr.	1	null
444	Pinco	Palla	laurea tr.	3	FT
77777	Bruno	Pasquale	laurea sp.	1	FT
88888	Pinco	Pietro	laurea sp.	1	CV

Corsi

cod	titolo	ciclo	docente
PR1	Programmazione I	laurea tr.	FT
ASD	Algoritmi e Str. Dati	laurea tr.	CV
INFT	Informatica Teorica	laurea sp.	ADP

Tutorato

studente	tutor
111	77777
222	77777
333	88888
444	88888

Numeri

professore	numero
FT	0971205145
FT	347123456
VC	0971205227
ADP	0971205363
ADP	338123456

Esami

studente	corso	voto	lode
111	PR1	27	false
222	ASD	30	true
111	INFT	24	false
77777	PR1	21	false
77777	ASD	20	false
88888	ASD	28	false
88888	PR1	30	false
88888	INFT	30	true