InformaticaUmanistica

# Basi di Dati

## *SQL-92*

*Concetti Fondamentali*

UNIVERSITÀ DI PISA

# 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

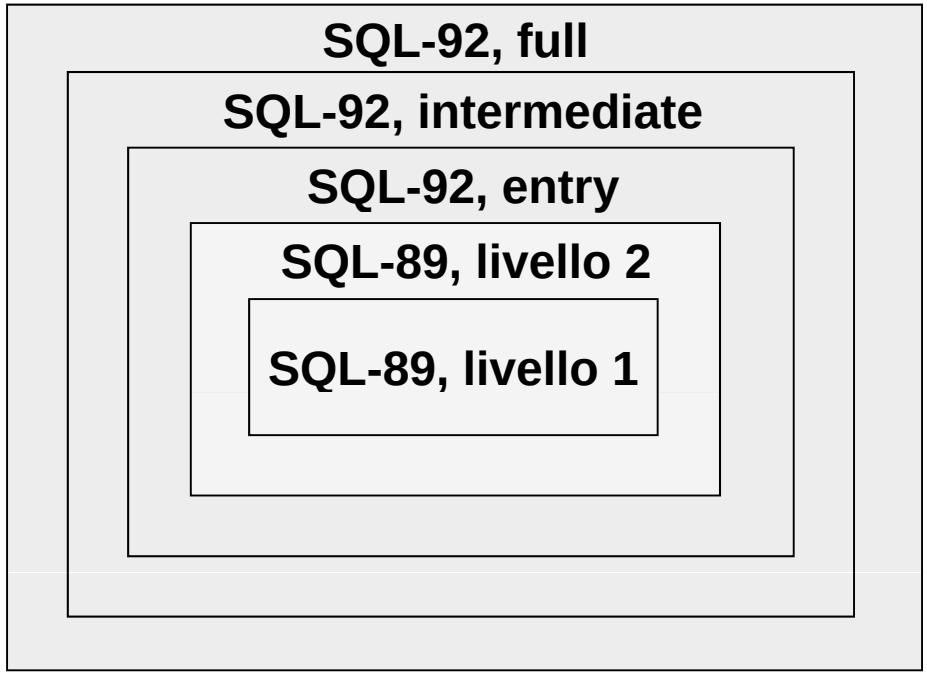
- **SQL ("Structured Query Language")**
  - linguaggio per l'interazione con il DBMS
  - tutte le operazioni vengono specificate in SQL
- **DDL ("Data Definition Language")**
  - creazione degli oggetti dello schema
- **DCL ("Data Control Language")**
  - controllo degli utenti e delle autorizzazioni
- **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

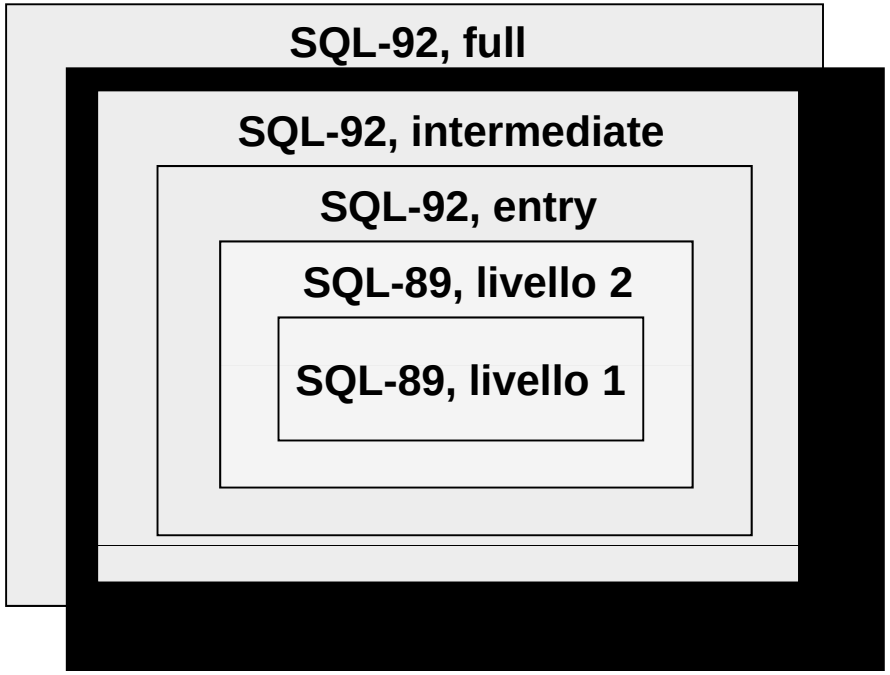


# Storia dello Standard

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

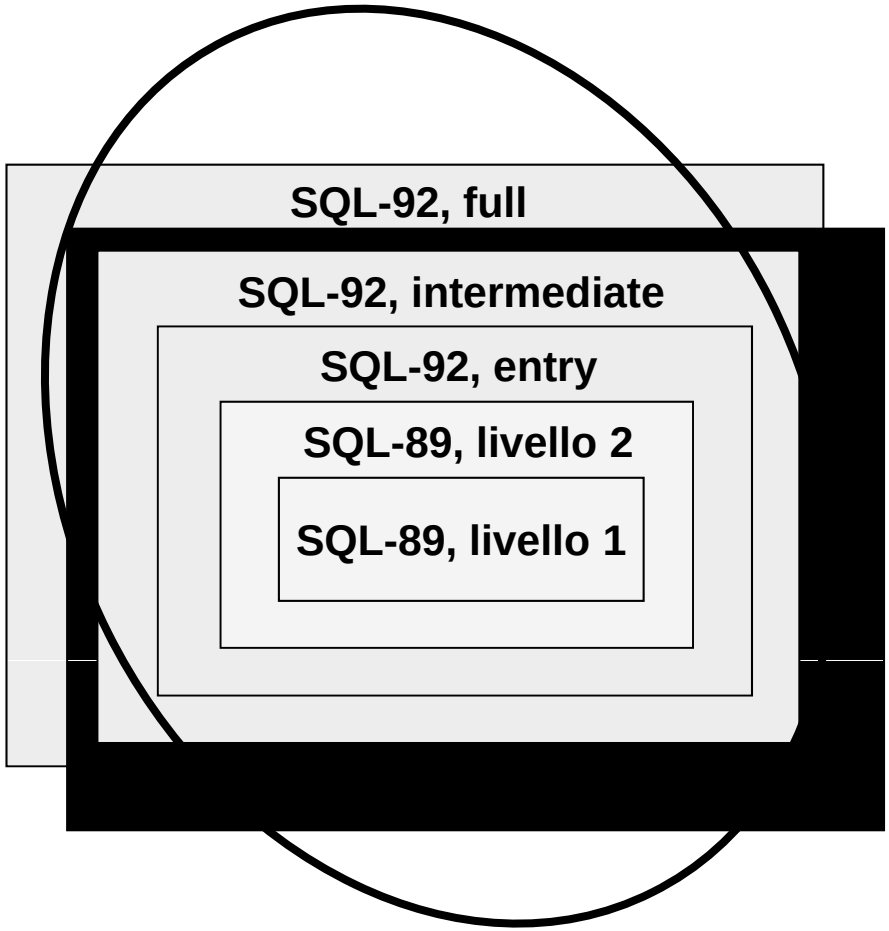
# Storia dello Standard

- **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**

- **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 diventa il proprietario della bd**
  - DROP DATABASE
    - ➔ **elimina una base di dati esistente (anche non vuota)**
    - ➔ **l'utente deve essere autorizzato**

# Creazione ed Eliminazione di Tabelle

- **Istruzioni del DDL**
  - CREATE TABLE
  - DROP TABLE

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

# Creazione ed Eliminazione di Tabelle

- **Esempio: la tabella Professori**

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



DROP TABLE Professori;
```

# Creazione ed Eliminazione di Tabelle

- **Esempio: la tabella Esami**

```
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));
```

# Creazione ed Eliminazione di Tabelle

- **<schema>**
  - una o più definizioni di attributo
  - zero o più definizioni di vincoli di tabella

- **Definizione di attributo**
  - <nomeattributo> <tipo> [<vincoli di colonna>]
  - vincoli sul singolo attributo

- **Definizioni di vincoli di tabella**
  - normalmente vincoli relativi a più attributi

# Creazione ed Eliminazione di Tabelle

- **<nomeattributo>**
  - identificatore
- **<tipo>**
  - INT, INTEGER
  - REAL, FLOAT
  - DECIMAL(lung,dec)
  - DOUBLE PRECISION
  - CHAR(n), CHARACTER(n)
  - VARCHAR(n)
  - LONG VARCHAR, TEXT
  - BOOLEAN, BOOL
  - DATE
  - TIME
  - TIMESTAMP
  - BINARY(n), BIT(n)
  - VARBINARY(n), VARBIT(n)
  - LONG VARBINARY, BLOB

# Creazione ed Eliminazione di Tabelle

- **Vincoli di colonna**
  - PRIMARY KEY
  - UNIQUE
  - NOT NULL
  - REFERENCES <chiave esterna>
    [ON update CASCADE]
    [ON delete CASCADE]
  - CHECK (<espressione>)

# Creazione ed Eliminazione di Tabelle

- **Vincoli di tabella (su più attributi)**
  - PRIMARY KEY (<lista attributi>)
  - UNIQUE (<lista attributi>)
  - FOREIGN KEY (<lista attributi>)
    REFERENCES <chiave esterna>
    [ON update CASCADE]
    [ON delete CASCADE]
  - CHECK (<espressione>)

# Creazione ed Eliminazione di Tabelle

- **Semantica**
  - CREATE TABLE
    - ➔ **crea una nuova tabella vuota secondo lo schema specificato**
    - ➔ **l'utente deve essere autorizzato**
    - ➔ **l'utente diventa il proprietario della tabella**
    - ➔ **attenzione ai vincoli di riferimento**
  - DROP TABLE
    - ➔ **elimina una tabella esistente**
    - ➔ **l'utente deve essere autorizzato**

```
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));
```

# Inserimenti

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

# Inserimenti

- **Esempi:**

```
INSERT INTO Professori VALUES
    (‘FT’, ‘Totti’, ‘Francesco’, ‘ordinario’,
     ‘Ingegneria’);


INSERT INTO Studenti VALUES
    (111, ‘Rossi’, ‘Mario’, ‘laurea tr.’, 3, null);


INSERT INTO Corsi VALUES
    (‘PR1’, ‘Programmazione 1’, ‘laurea tr.’, ‘FT’);
```

# Interrogazioni

- **Istruzione del DML**
  - SELECT
  - sintassi per specificare operatori dell'algebra
- **Filosofia**
  - parzialmente dichiarativa
  - si specificano gli operatori da applicare, non l'ordine in cui devono essere applicati
  - l'ottimizzatore sceglie la strategia ottima

# Interrogazioni

- **Forma standard dell'algebra**
  - una o più sottointerrogazioni
  - correlate da operatori insiemistici

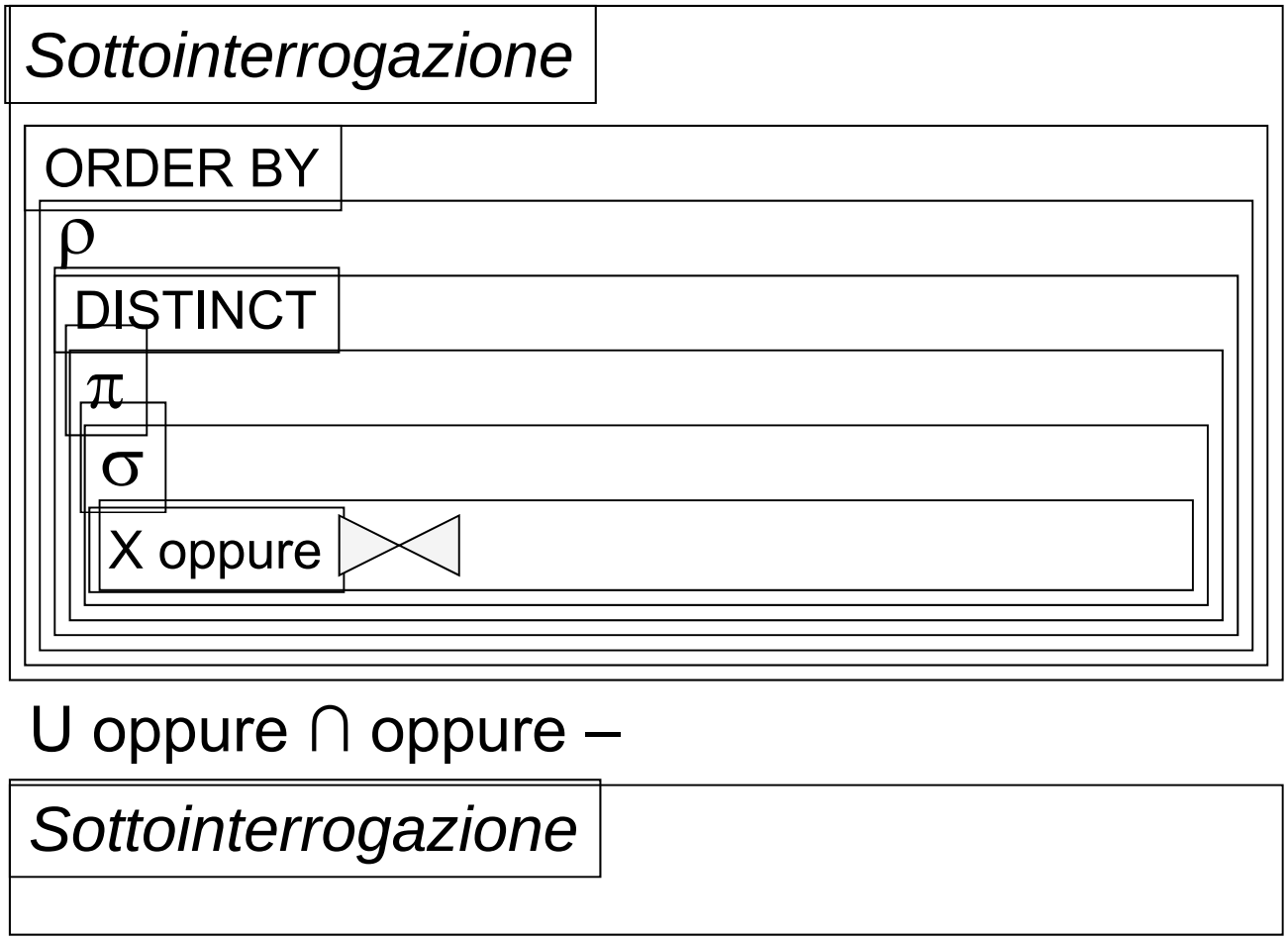
- **Sottointerrogazioni**
  - strategia a: prodotti cartesiani tra le tabelle (con eventuali alias)
  - strategia b: join tra le tabelle (con eventuali alias)

>>

# Interrogazioni

- **Sottointerrogazioni (Continua)**
  - selezioni
  - proiezioni (con funzioni aggregative)
  - eliminazione di duplicati (DISTINCT)
  - ridenominazioni
  - ordinamenti finali (ORDER BY)

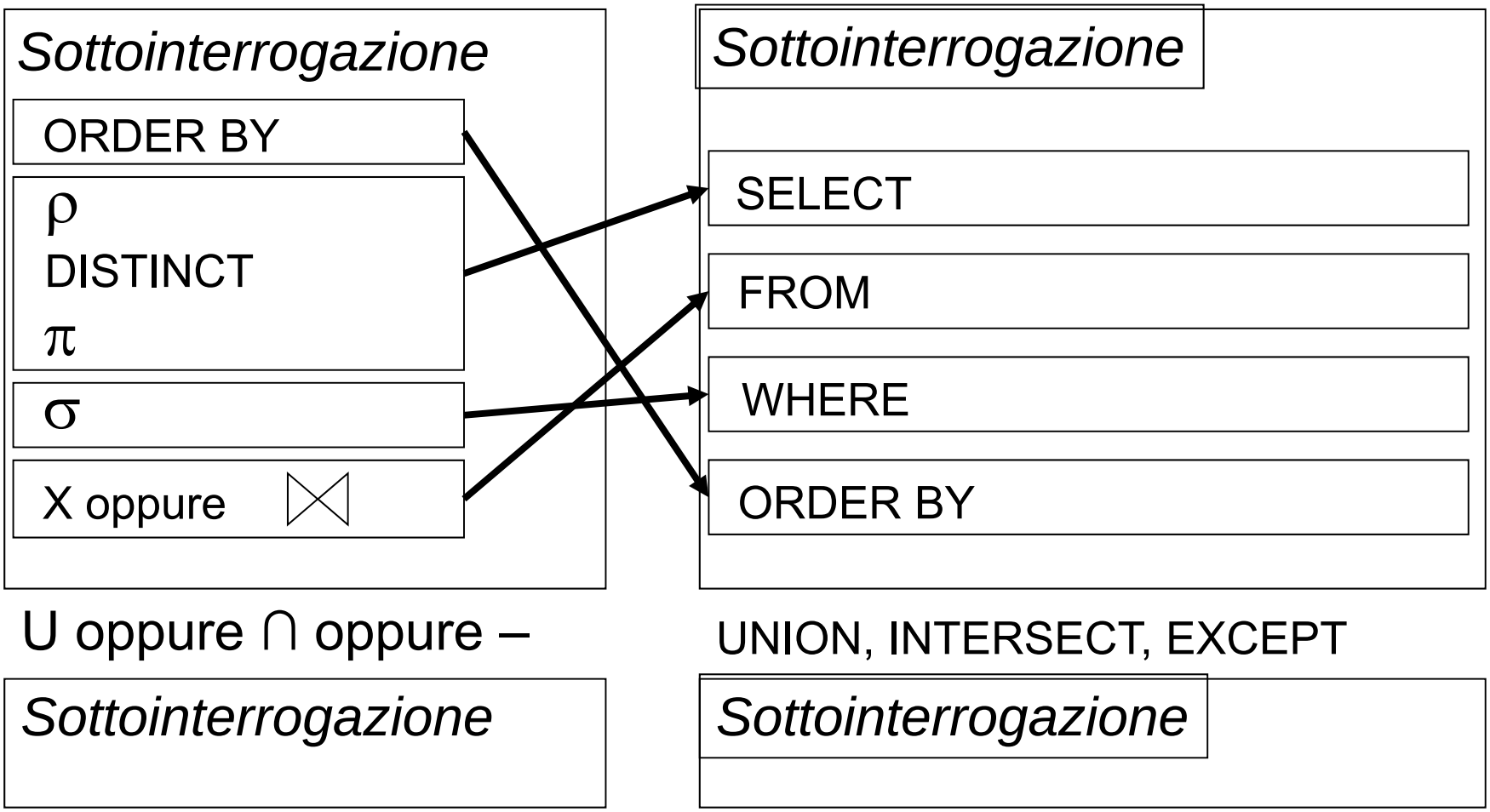
# Interrogazioni



# Interrogazioni

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

# Interrogazioni



# Clausola FROM

- **Strategia a**
  - FROM R, S, T AS V …
- **Strategia b**
  - FROM S JOIN R ON S.A=R.B JOIN T AS V..
- **Strategia mista**
  - FROM S JOIN R ON S.A=R.B, T AS V…
- **Semantica**
  - applicazione degli operatori corrispondenti

# Clausola WHERE

- **WHERE <condizione>**
- **<condizione>**
  - condizioni di selezione, connettivi booleani
- **Esempio**

  **WHERE R.A=1 AND S.B>V.C OR V.D IS NOT NULL**

- **Semantica**
  - selezione corrispondente

# Clausola SELECT

- **SELECT [DISTINCT] <attributi> | ***
- **<attributi>**
  - lista di nomi di attributo, AS per le ridenom.
- **Esempio**

  **SELECT R.A, R.B AS C, V.D  oppure  SELECT ***
- **Semantica**
  - proiezioni, distinct e ridenominazioni corrispondenti

# Clausola ORDER BY

- **ORDER BY <attributi>**
- **<attributi>**
  - lista di attributi
  - <attributo> {ASC | DESC}
- **Esempio**

  **ORDER BY R.A, R.B DESC**
- **Semantica**
  - ordinamenti corrispondenti

# Esempi

- **“Studenti della laurea triennale di anni successivi al primo”**

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

```
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

- **“Voto medio riportato negli esami”**

Risultato = $\pi_{AVG(voto)}$ (Esami)

```
SELECT AVG(voto)
FROM Esami;
```

# Esempi

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

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

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

# Esempi

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

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

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

# Esempi

- **“Matricola e cognome degli studenti che hanno sostenuto l’esame di informatica teorica” (strategia b)**

Risultato = $\pi$ matricola, cognome ( $\sigma$ titolo=‘Inform. t.’ (
Studenti $\bowtie_{matr=studente}$ Esami $\bowtie_{cod=corso}$ Corsi ))

```
SELECT matricola, cognome
FROM Studenti JOIN Esami ON matr=studente
              JOIN Corsi ON cod=corso
WHERE titolo=‘Inform. t.’;
```

# Esempi

- **“Cognome e nome delle persone”**

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

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

# Esempi

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

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

# 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

- **“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

- **“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;
```

# Interrogazioni: Metodo di Scrittura

- **Scrivere l'interrogazione in algebra relazionale utilizzando la forma standard**
- **Tradurre gli operatori nella sintassi di SQL**
- **Bisogna scegliere tra**
  - strategia a: prodotti cartesiani
  - strategia b: join
  - in generale è più efficiente la seconda

# Cancellazioni

- **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)

# Cancellazioni

- **Esempi:**

```
DELETE FROM Numeri;


DELETE FROM Studenti WHERE matr=111;


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

# Aggiornamenti

- **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)

# Aggiornamenti

- **Esempi:**

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

```
UPDATE Studenti SET matr=11111
   WHERE matr=111;
```

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

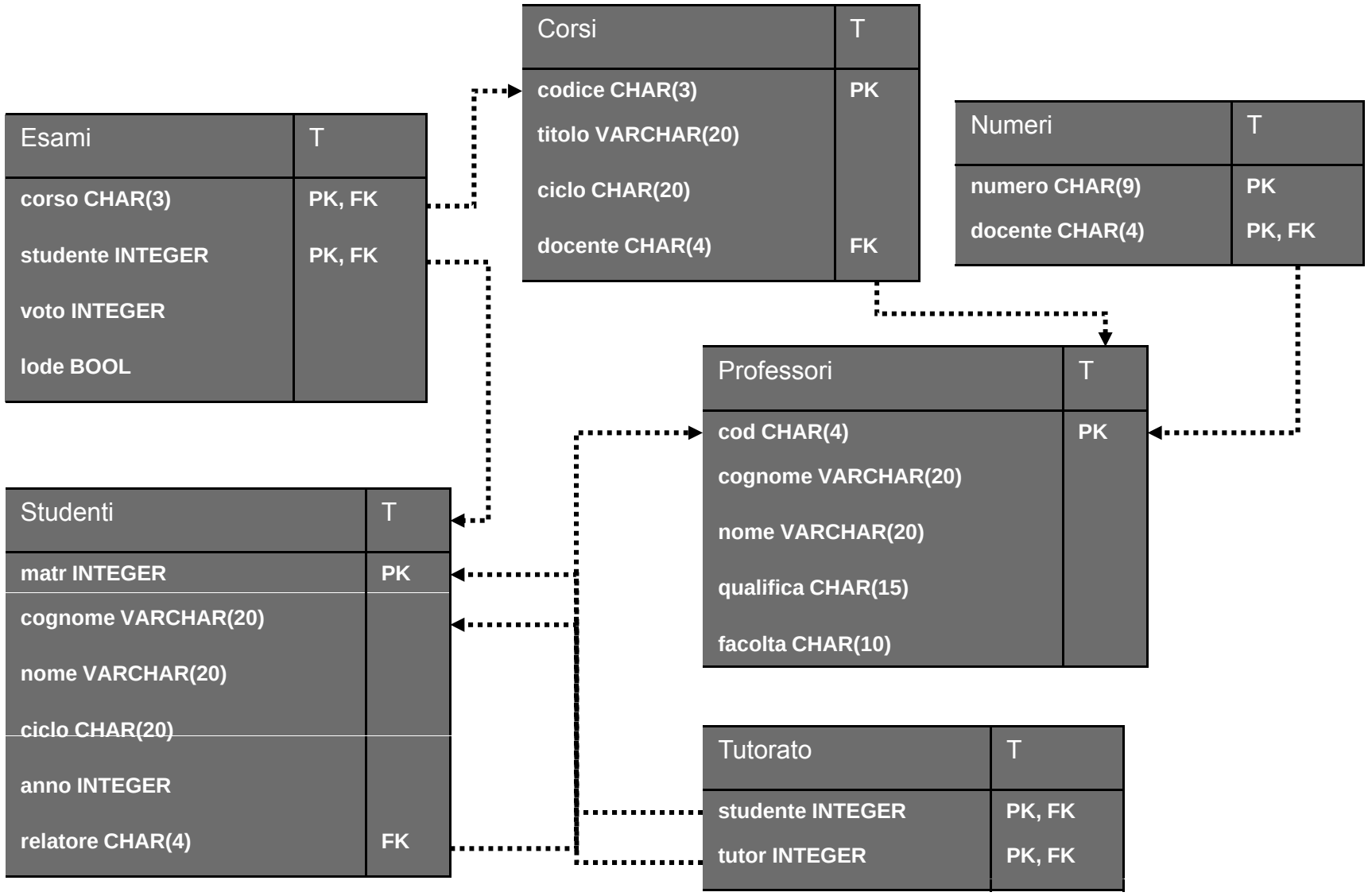
# Concetti Fondamentali

- **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 |