

Basi di Dati

SQL-92

Concetti Fondamentali



UNIVERSITÀ DI PISA

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)

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

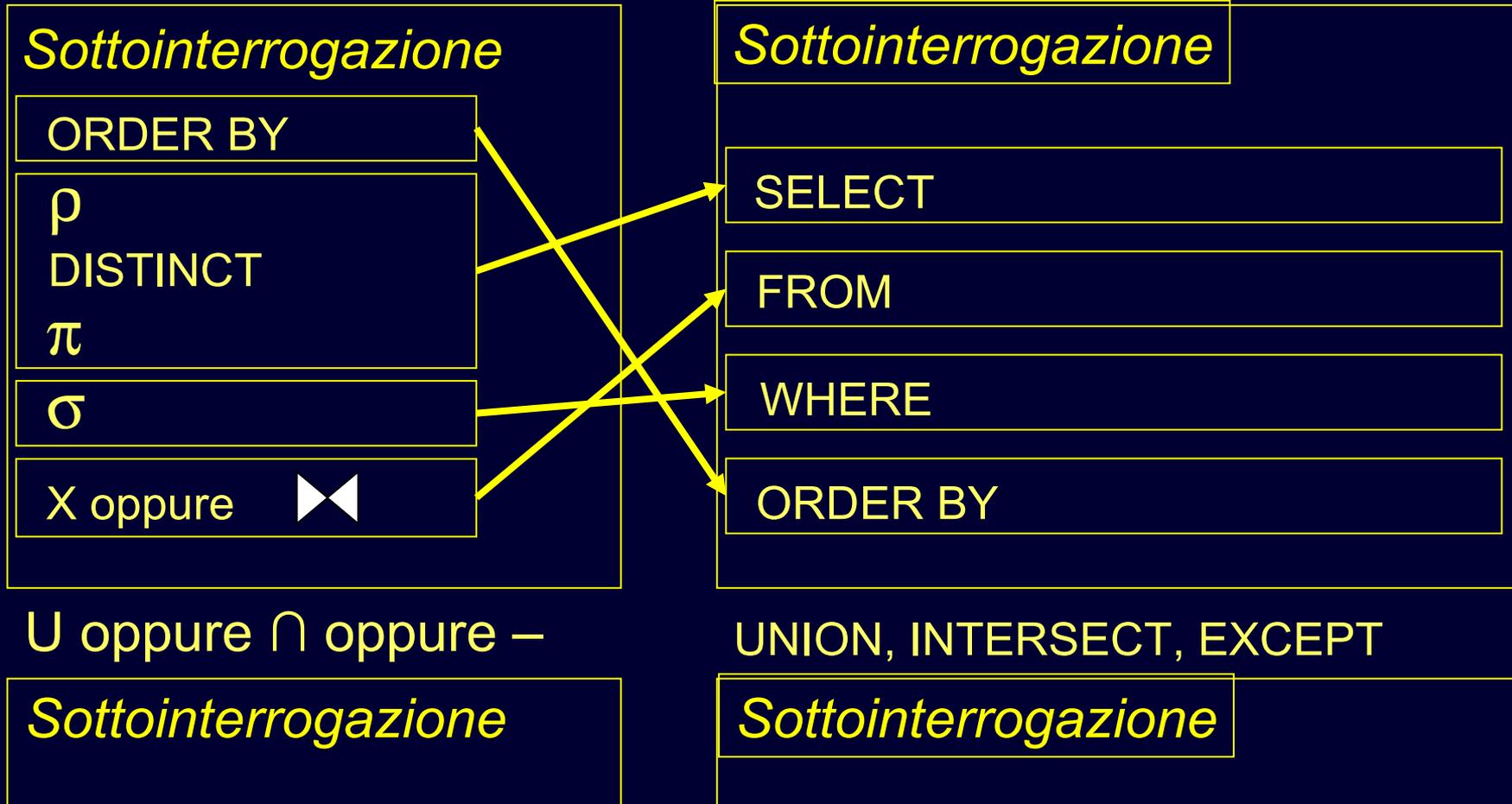
◆ Sottointerrogazioni (Continua)

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

Interrogazioni

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

Interrogazioni



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

```
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)
);
```

```
TABLE Corsi (
    cod char(3) PRIMARY KEY,
    titolo varchar(20) NOT NULL,
    ciclo char(20),
    docente char(4)
    REFERENCES Professori(cod)
);
```

```
TABLE Tutorato (
    studente integer
    REFERENCES Studenti(matr),
    tutor integer
    REFERENCES Studenti(matr),
    UNIQUE (studente,tutor));
```

```
TABLE Esami (
    studente integer
    REFERENCES Studenti(matr)
    corso char(3)
    REFERENCES Corsi(cod),
    voto integer,
    lode bool,
    CHECK (voto>=18 and voto<=30),
    CHECK (not lode or voto=30),
    UNIQUE (studente, corso));
```

```
TABLE Numeri (
    professore char(4)
    REFERENCES Professori(cod),
    numero char(9),
    UNIQUE (professore,numero));
```

Esempi - Selezione

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

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

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

Esempi – Proiezione e DISTINCT

- ◆ “Cognomi e nomi degli studenti, senza duplicati”

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

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

Esempi - Ordinamento

- ◆ “Cognomi e nomi degli studenti, in ordine alfabetico”

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

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

Esempi – Prodotto cartesiano

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

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

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

Esempi - Join

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

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

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

~~WHERE~~

~~ORDER BY~~

Esempi – Join multipli

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

Risultato = π matricola, cognome (σ titolo='Inform. t.' (Students \bowtie matr=studente Esami \bowtie cod=corso Corsi))

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

Esempi – Operatori insiemistici (a)

◆ “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 – Operatori insiemistici (b)

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

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

Esempi – Operatori insiemistici (b)

- ◆ “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 cognome, nome
FROM Studenti JOIN Professori ON relatore=cod
WHERE ciclo='laurea tr.';
```

Esempi – Un caso complesso

- ◆ **“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)

Risultato = $\pi_{\text{cognome, nome}}$ (

$\sigma_{\text{Esami.corso='Pr1' AND EsamiTutor.corso='Pr1' AND Esami.voto > EsamiTutor.voto}}$ (

Studenti  matr=studente Esami

 $\text{matr=Tutorato.studente}$ Tutorato

 $\text{Tutorato.tutor=EsamiTutor.studente}$ (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;
```