InformaticaUmanistica

Basi di Dati

SQL-92

Concetti Avanzati

UNIVERSITÀ DI PISA

Concetti Avanzati

- **Raggruppamenti**
  - Clausole GROUP BY e HAVING
  - Forma Generale della SELECT
- **Nidificazione**
  - Uso nel DML e DDL
  - Nidificazione, Viste e Potere Espressivo
- **Esecuzione di una Query SQL**

Interrogazioni con Raggruppamenti

Nucleo della SELECT

- SELECT, FROM, [WHERE]

- **Clausola aggiuntiva**
  - [ORDER BY]

- **Ulteriori clausole aggiuntive**
  - [GROUP BY]
  - [HAVING]

Clausole GROUP BY e HAVING

- **GROUP BY**
  - operatore di “raggruppamento”
- **Sintassi**
  - GROUP BY <attributi di raggruppamento>
- **Semantica**
  - raggruppamento della tabella
  - divisione in gruppi delle ennuple
  - raggruppamento sulla base dei valori comuni per gli attributi di raggruppamento

Clausole GROUP BY e HAVING

Esempio: raggruppamento della tabella studenti per ciclo (GROUP BY ciclo)

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**

gruppo A
ciclo='laurea tr.'

gruppo B
ciclo='laurea sp.'

Clausole GROUP BY e HAVING

Esempio: raggruppamento della tabella studenti per ciclo e anno
(GROUP BY ciclo, anno)

Studenti

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

gruppo A
laurea tr., 1

gruppo B
laurea tr., 2

gruppo C
laurea tr., 3

gruppo D
laurea sp., 1

Clausole GROUP BY e HAVING

Esempio: raggruppamento della tabella studenti per matricola (GROUP BY matr)

Studenti

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

una ennupla per ogni gruppo

Clausole GROUP BY e HAVING

- **Caratteristiche dei gruppi**
  - collezioni di ennuple
  - valori comuni per gli attributi di raggruppam.

- **Operazioni interessanti sui gruppi**
  - funzioni aggregative
  - analisi della distribuzione di valori tra i gruppi
    es: numero di studenti per ciclo o per anno
  - OLAP (“On Line Analytical Processing”)

Clausole GROUP BY e HAVING

- **Interrogazioni con raggruppamento**
  - attributi di raggruppamento (nella GROUP BY)
  - proiezioni su attributi di raggruppamento e funzioni aggregative applicate al gruppo (nella SELECT)
  - condizioni sui gruppi (che coinvolgono funzioni aggregative) (nella HAVING)

Clausole GROUP BY e HAVING

- **Esempio: numero di studenti per ciclo**

```
SELECT ciclo, count(*)
FROM Studenti
GROUP BY ciclo;
```

ciclo	count(*)
laurea tr.	4
laurea sp.	2

Semantica:

- viene valutata la clausola FROM
- viene effettuato il raggruppam. secondo la GROUP BY
- viene valutata la clausola SELECT per ciascun gruppo (ogni gruppo contribuisce ad UNA sola ennupla del ris.)

Clausole GROUP BY e HAVING

- **Esempio: numero di studenti per ciclo (continua)**

```
SELECT count(*)
FROM Studenti
GROUP BY ciclo;
```

count(*)
4
2

Clausole GROUP BY e HAVING

- **Vincoli sintattici sulla SELECT**
  - se c'è una GROUP BY, solo gli attributi di raggruppamento possono comparire nella SELECT

```
SELECT ciclo, count(*)
FROM Studenti
GROUP BY ciclo;
```

```
SELECT count(*)
FROM Studenti
GROUP BY ciclo;
```

~~SELECT anno, count(*)~~
~~FROM Studenti~~
~~GROUP BY ciclo;~~

Clausole GROUP BY e HAVING

- **Esempio: distribuzione per anno degli studenti della laurea triennale**

```
SELECT anno, count(*) as numstud
FROM Studenti
WHERE ciclo='laurea tr.'
GROUP BY anno;
```

Clausole GROUP BY e HAVING

I passo: WHERE ciclo='laurea tr.'

matr	cognome	nome	ciclo	anno	relatore
111	**Rossi**	**Mario**	**laurea tr.**	**1**	**null**
333	**Rossi**	**Maria**	**laurea tr.**	**1**	**null**
222	**Neri**	**Paolo**	**laurea tr.**	**2**	**null**
444	**Pinco**	**Palla**	**laurea tr.**	**3**	**FT**

II passo: GROUP BY anno

matr	cognome	nome	ciclo	anno	relatore
111	**Rossi**	**Mario**	**laurea tr.**	**1**	**null**
333	**Rossi**	**Maria**	**laurea tr.**	**1**	**null**
222	**Neri**	**Paolo**	**laurea tr.**	**2**	**null**
444	**Pinco**	**Palla**	**laurea tr.**	**3**	**FT**

risultato finale:
SELECT anno, count(*) as numstud

anno	numstud
1	**2**
2	**1**
3	**1**

Clausole GROUP BY e HAVING

- **Esempio: distribuzioni delle medie, solo per i corsi con più di 2 esami**
  - non è possibile usare la WHERE
  - HAVING: condizioni aggregate su gruppi

```
SELECT corso, avg(voto) as votomedio
FROM Esami
GROUP BY corso
HAVING count(voto)>2;
```

Clausole GROUP BY e HAVING

Esami

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

risultato finale

corso	votomedio
PR1	26
ASD	26

I passo: raggruppamento secondo la GROUP BY

II passo: selezione dei gruppi secondo la HAVING

III passo: proiezione e rid. secondo la SELECT

Forma Generale della SELECT

- **Forma generale della SELECT**
  - SELECT [DISTINCT] <risultato>
  - FROM <join o prodotti cartesiani>
  - [WHERE <condizioni>]
  - [GROUP BY <attributi di raggruppamento>]
  - [HAVING <condizioni sui gruppi>]
  - [ORDER BY <attributi di ordinamento>]

Forma Generale della SELECT

- **Se la GROUP BY manca**
  - tutte le ennuple ottenute dopo la WHERE vengono considerate un unico gruppo
  - in questo caso le funzioni aggregative producono un unico valore e non sono ammessi attributi ordinari nella SELECT

- **Nota sulla semantica**
  - tutti i valori NULL normalmente vengono raggruppati assieme

Forma Generale della SELECT

- **Una semantica operazionale**
  - viene valutata la clausola FROM
    - ➔ **join o prodotti cartesiani >> unica tabella**
  - viene valutata la clausola WHERE
    - ➔ **selezione delle ennuple della tabella**
  - viene valutata l'eventuale GROUP BY
    - ➔ **raggruppamento delle ennuple della tabella**
  - viene valutata l'eventuale HAVING
    - ➔ **selezione dei gruppi della tabella >>**

Forma Generale della SELECT

- **Una semantica operazionale (continua)**
  - viene valutata la clausola SELECT
    - ➔ **proiezioni, espressioni e funzioni aggregative**
    - ➔ **ridenominazioni**
    - ➔ **eventuale eliminazione di duplicati**
  - viene valutata la clausola ORDER BY
    - ➔ **ordinamenti finali**

Forma Generale della SELECT

- **Esempio: medie in ordine decrescente degli studenti della laurea specialistica che hanno sostenuto almeno due esami**

```
SELECT matr, cognome, nome, avg(voto)
FROM Studenti JOIN Esami ON matr=studente
WHERE ciclo='laurea sp.'
GROUP BY matr, cognome, nome
HAVING count(*)>=2
ORDER BY avg(voto) DESC;
```

Forma Generale della SELECT

Studenti

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

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

Forma Generale della SELECT

Passo 1: FROM Studenti JOIN Esami ON matr=studente

matr	cognome	nome	ciclo	relat	studente	corso	voto	lode
111	**Rossi**	**Mario**	**laurea tr.**	**null**	**111**	**PR1**	**27**	**false**
111	**Rossi**	**Mario**	**laurea tr.**	**null**	**111**	**INFT**	**24**	**false**
222	**Neri**	**Paolo**	**laurea tr.**	**null**	**222**	**ASD**	**30**	**true**
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**PR1**	**21**	**false**
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**ASD**	**20**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**ASD**	**28**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**PR1**	**30**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**INFT**	**30**	**true**

Forma Generale della SELECT

Passo II: WHERE ciclo='laurea sp.'

matr	cognome	nome	ciclo	relat	studente	corso	voto	lode
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**PR1**	**21**	**false**
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**ASD**	**20**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**ASD**	**28**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**PR1**	**30**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**INFT**	**30**	**true**

Passo III: GROUP BY matr, cognome, nome

matr	cognome	nome	ciclo	relat	studente	corso	voto	lode
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**PR1**	**21**	**false**
77777	**Bruno**	**Pasquale**	**laurea sp.**	**FT**	**77777**	**ASD**	**20**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**ASD**	**28**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**PR1**	**30**	**false**
88888	**Pinco**	**Pietro**	**laurea sp.**	**VC**	**88888**	**INFT**	**30**	**true**

Forma Generale della SELECT

Passo IV: HAVING count(*) >= 2

matr	cognome	nome	ciclo	relat	studente	corso	voto	lode
77777	Bruno	Pasquale	laurea sp.	FT	77777	PR1	21	false
77777	Bruno	Pasquale	laurea sp.	FT	77777	ASD	20	false
88888	Pinco	Pietro	laurea sp.	VC	88888	ASD	28	false
88888	Pinco	Pietro	laurea sp.	VC	88888	PR1	30	false
88888	Pinco	Pietro	laurea sp.	VC	88888	INFT	30	true

Passo V: SELECT matr, cognome, nome, avg(voto)

matr	cognome	nome	avg(voto)
77777	Bruno	Pasquale	20,5
88888	Pinco	Pietro	29,66

Passo VI: ORDER BY avg(voto) DESC

matr	cognome	nome	avg(voto)
88888	Pinco	Pietro	29,66
77777	Bruno	Pasquale	20,5

Interrogazioni Nidificate

- **SELECT Nidificate**
  - la clausola WHERE di una SELECT contiene un'altra SELECT

- **Due possibili utilizzi**
  - condizioni basate su valori semplici (SELECT che restituiscono un singolo valore)
  - condizioni basate su collezioni (SELECT ordinarie che restituiscono insiemi di ennup.)

Interrogazioni Nidificate

- **Condizioni su valori semplici**
  - confrontano il valore di un attributo con il risultato di una SELECT “scalare”
  - operatori: >, <, =, >=, <=, <>, LIKE, IS NULL

- **SELECT “scalare”**
  - SELECT che restituisce un’unica ennupla con un un unico attributo
  - tipicamente: funzione aggregativa

Interrogazioni Nidificate

- **Esempio: lo studente con la matricola più alta**

```
SELECT matr, cognome, nome
FROM Studenti
WHERE matr = (SELECT max(matr)
              FROM Studenti);
```

max(matr)
88888

per ogni ennupla di Studenti, il valore della matricola viene confrontato con il numero 88888

Interrogazioni Nidificate

- **Condizioni su valori non scalari (collezioni)**
  - confrontano il valore di un attributo con il risultato di una SELECT generica (collezione di ennuple)
  - operatori: ordinari combinati con ANY, ALL

- **ANY**
  - “un elemento qualsiasi della collezione”;
    es: = ANY, oppure IN

- **ALL**
  - “tutti gli elementi della collezione”; es: > ALL

Interrogazioni Nidificate

- **Esempio: lo studente con la matricola più alta (senza funzioni aggregative)**

```
SELECT matr, cognome, nome
FROM Studenti
WHERE matr >= ALL (SELECT matr
                   FROM Studenti);
```

per ogni ennupla di Studenti, il valore della matricola viene confrontato con tutte le matricole

matr
111
222
333
444
77777
88888

Interrogazioni Nidificate

- **Sintatticamente**
  - no ORDER BY nelle SELECT nidificate
- **Semantica**
  - ogni volta che è necessario verificare la condizione, viene calcolato il risultato della SELECT interna
  - il processo si può ripetere a più livelli
  - in pratica: memorizzazione in una tabella temporanea

Interrogazioni Nidificate

- **Nota: Le interrogazioni nidificate possono sostituire i join**
- **Esempio: voti riportati in corsi della laurea triennale**

```
SELECT voto
FROM Esami
WHERE corso = ANY (SELECT cod
                   FROM Corsi
                   WHERE ciclo='laurea tr.');
```

cod
PR1
ASD

stessa semantica del join

Interrogazioni Nidificate

- **Nota: Le interrogazioni nidificate possono sostituire intersezione e differenza**

- **Esempio: cognome e nome dei professori ordinari che non hanno tesisti**

```
SELECT cognome, nome
FROM Professori
WHERE qualifica='ordinario' AND
      cod <> ALL (SELECT DISTINCT relatore
                  FROM Studenti );
```

relatore
FT
VC

Interrogazioni Nidificate

- **Metodologicamente**
  - i join si realizzano applicando i join
  - le op. insiemistiche si realizzano applicando gli op. insiemistici

- **Quando può servire la nidificazione**
  - nei sistemi in cui non c'è intersezione o diff.
    es: Access e MySQL
  - condizioni nella WHERE su aggregati
    es: lo studente con la media più alta

Interrogazioni Nidificate

- **Aspetti avanzati (cenni)**
  - è possibile fare riferimento ad ennuple della SELECT esterna nella SELECT interna
  - regole di visibilità
  - operatore EXISTS: verifica se una SELECT nidificata restituisce un risultato vuoto
  - sostanzialmente servono per fare join
  - non utilizzeremo questa forma

Utilizzo nel DML e nel DDL

- **Utilizzo nel DML**
  - nella DELETE, nella UPDATE e nella INSERT, clausola WHERE completa

- **Utilizzo nel DDL**
  - vincoli di ennupla
  - CHECK (<condizione>)
  - <condizione>: sintassi e semantica identica alla condizione della clausola WHERE

Utilizzo nel DML e nel DDL

- **Esempio: è possibile sostenere esami solo per i corsi per cui c'è un docente**

```
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));
```

Vincolo di ennupla aggiuntivo

```
CHECK (corso = ANY
      (SELECT cod
       FROM Corsi
       WHERE docente IS NOT NULL))
```

Nidificazione, Viste, Potere Espressivo

- **Terzo utilizzo delle viste**
  - esprimere interrogazioni altrimenti inesprimibili

- **Esempio: Studenti con la media più alta**
  - per calcolare la media di ciascuno studente serve un raggruppamento
  - condizione nidificata sui gruppi
  - non è possibile nidificare la HAVING (nidificazione solo nella WHERE)

Nidificazione, Viste, Potere Espressivo

- **Soluzione con le viste**

```
CREATE VIEW StudentiConMedia
AS SELECT matr, cognome, nome,
          avg(voto) as media
FROM Esami JOIN Studenti
       on studente=matr
GROUP BY matr, cognome, nome;
```

StudentiConMedia

matr	cognome	nome	media
111	Rossi	Mario	20,7
222	Neri	Paolo	24,5
333	Rossi	Maria	25,8
444	Pinco	Palla	19,6
77777	Bruno	Pasquale	26
88888	Pinco	Pietro	26

```
SELECT matr, cognome, nome
FROM StudentiConMedia
WHERE media =(SELECT max(media)
              FROM StudentiConMedia);
```

Nidificazione, Viste, Potere Espressivo

- **Un ulteriore esempio**
  - numero medio di docenti appartenenti alle facoltà

~~~~
SELECT avg(count(cod))
FROM Professori
GROUP BY facolta;
~~~~

```
CREATE VIEW Facolta AS
  SELECT facolta, count(*)
         as numdocenti
  FROM Professori
  GROUP BY facolta;
```

```
SELECT avg(numdocenti)
FROM Facolta;
```

Esecuzione di una Query SQL

- **Processo di valutazione di una query**
  - la query viene inviata al DBMS interattivamente o da un'applicazione
  - il DBMS effettua l'analisi sintattica del codice SQL
  - il DBMS effettua le verifiche sulle autorizzazioni di accesso
  - il DBMS esegue il processo di ottimizzazione della query

Ottimizzazione delle Interrogazioni

- **Processo di ottimizzazione**
  - scelta di una strategia efficiente per la valutazione della query

- **Piano di esecuzione di una query**
  - scelta dell'ordine di applicazione degli operatori algebrici necessari
  - strategia di calcolo del risultato di ciascun operatore algebrico attraverso le strutture di accesso disponibili

Un Esempio

- **Studiamo la seguente interrogazione: *"Nomi e cognomi dei tesisti di Christian Vieri iscritti alla laurea specialistica"***

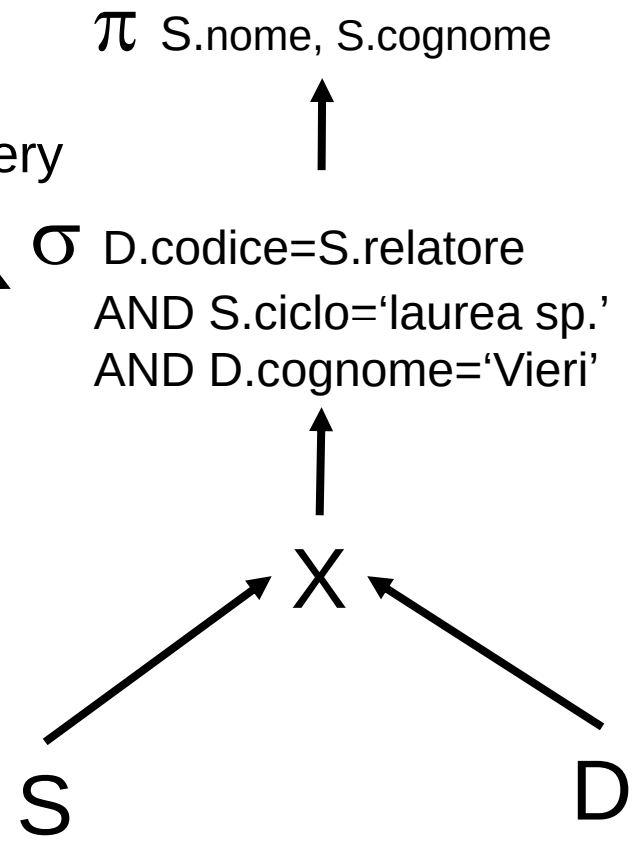
```
SELECT Studente.nome, Studente.cognome
FROM Docente, Studente
WHERE Docente.codice=Studente.relatore AND
      Studente.ciclo = 'laurea sp.' AND
      Docente.cognome = 'Vieri';
```

Un Esempio

- **Forma standard**

```
SELECT S.nome,
       S.cognome
FROM Docente AS D,
     Studente AS S
WHERE D.codice=S.relatore AND
      S.ciclo = 'laurea sp.'
      AND D.cognome = 'Vieri';
```

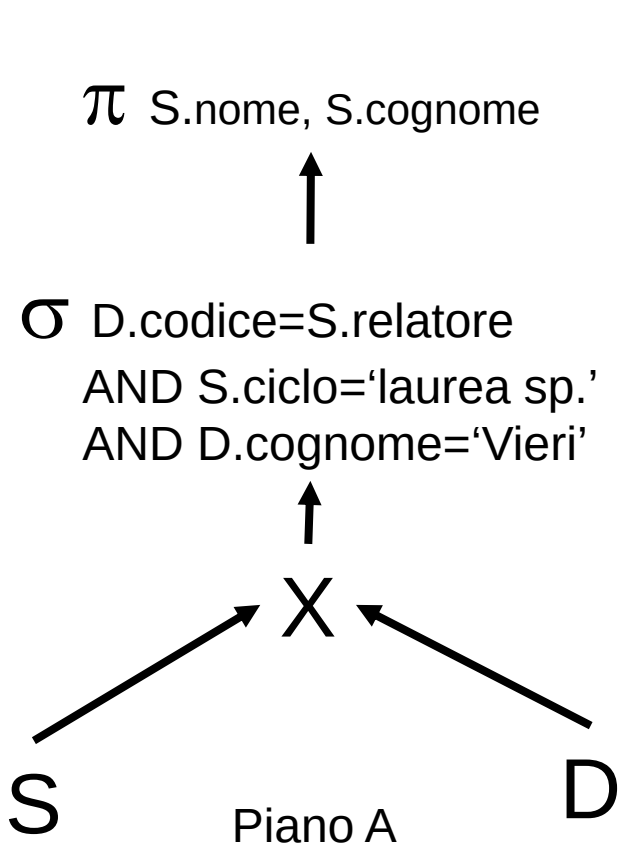
Albero degli operatori della query

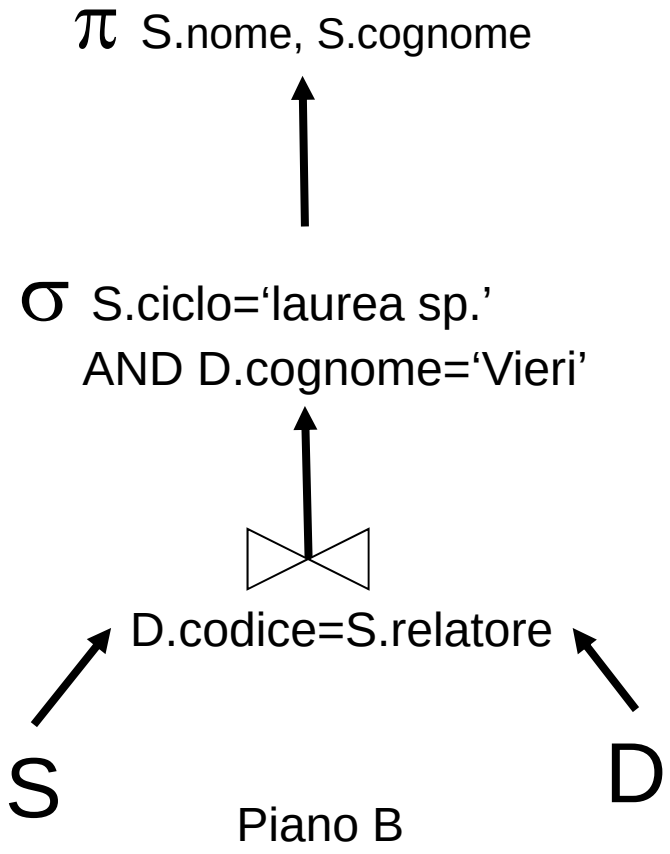


π S.nome, S.cognome (σ D.codice=S.relatore AND S.ciclo='laurea sp.' AND D.cognome='Vieri' (S X D))

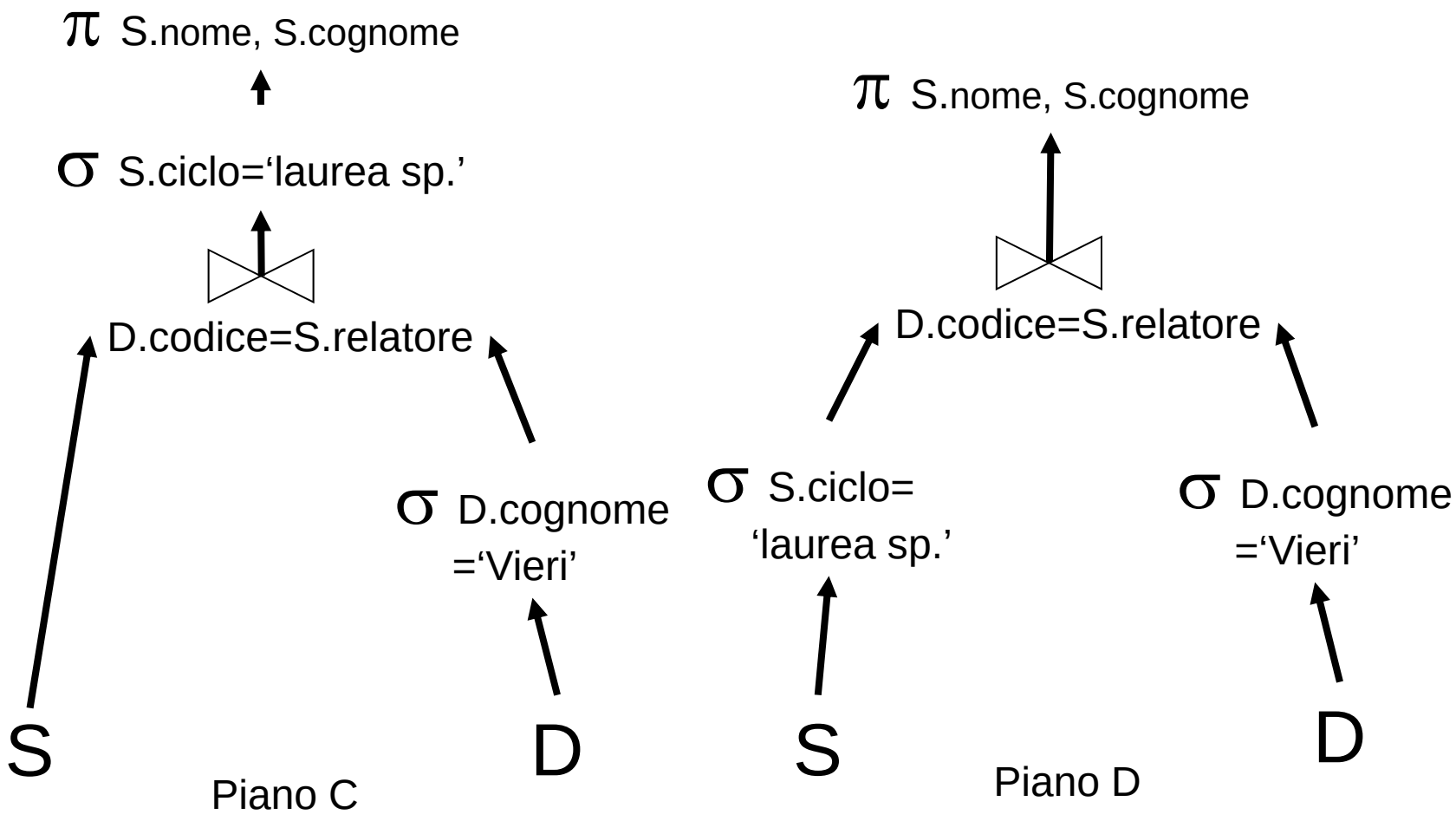
Un Esempio

- **Non è l'unico possibile**





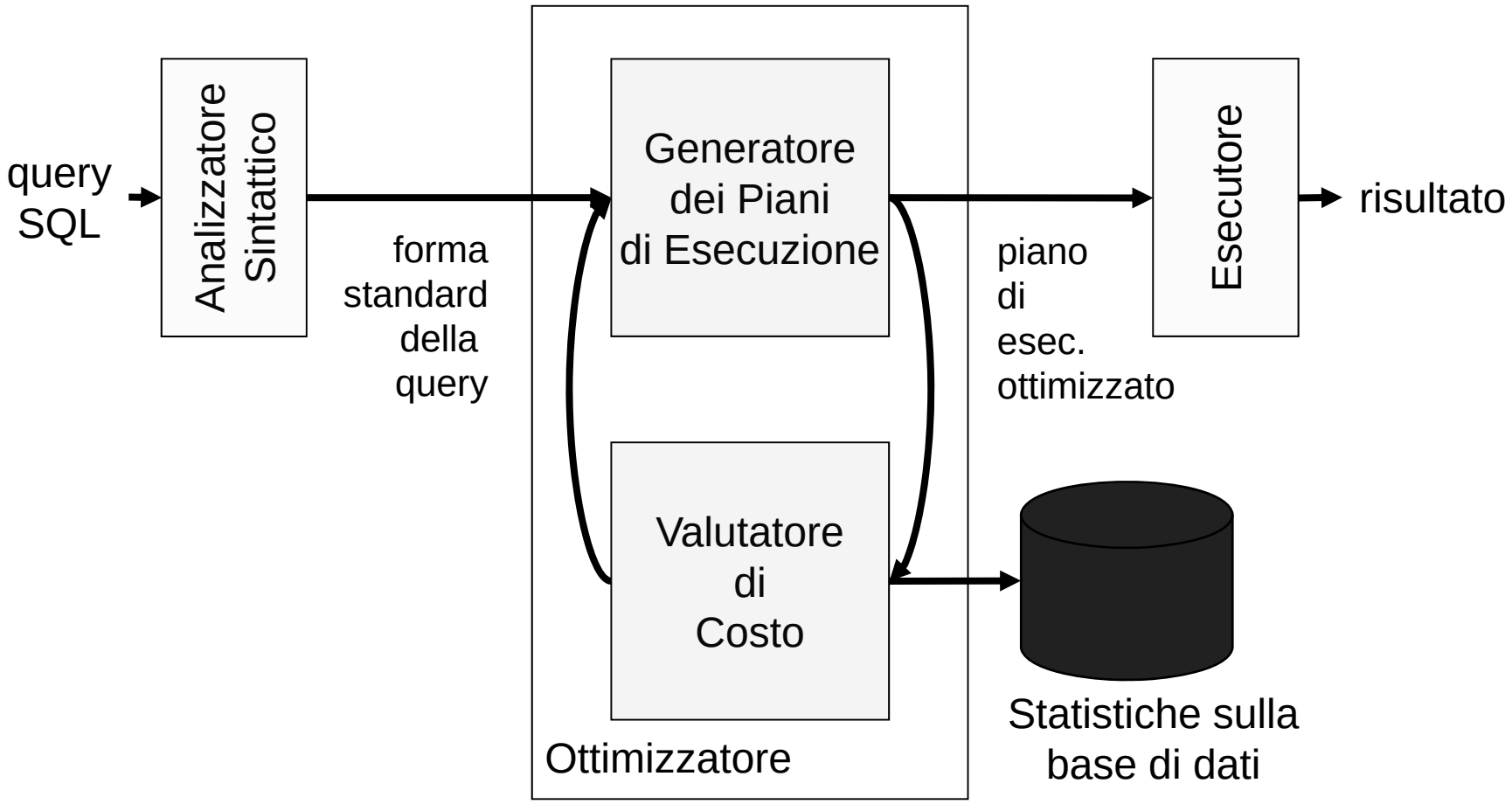
Altri Piani di Esecuzione



Ottimizzazione delle Interrogazioni

- **Per effettuare l'ottimizzazione**
  - vengono valutati molti diversi piani di esecuzione alternativi
  - l'ottimizzatore dispone di statistiche sul contenuto della base di dati (dimensione delle tabelle, dimensione dei record, dimensione degli indici, selettività ecc.)
  - sulla base delle statistiche viene stimato il costo di ciascun piano di esecuzione (numero di accessi ai blocchi su disco)

Ottimizzazione delle Interrogazioni



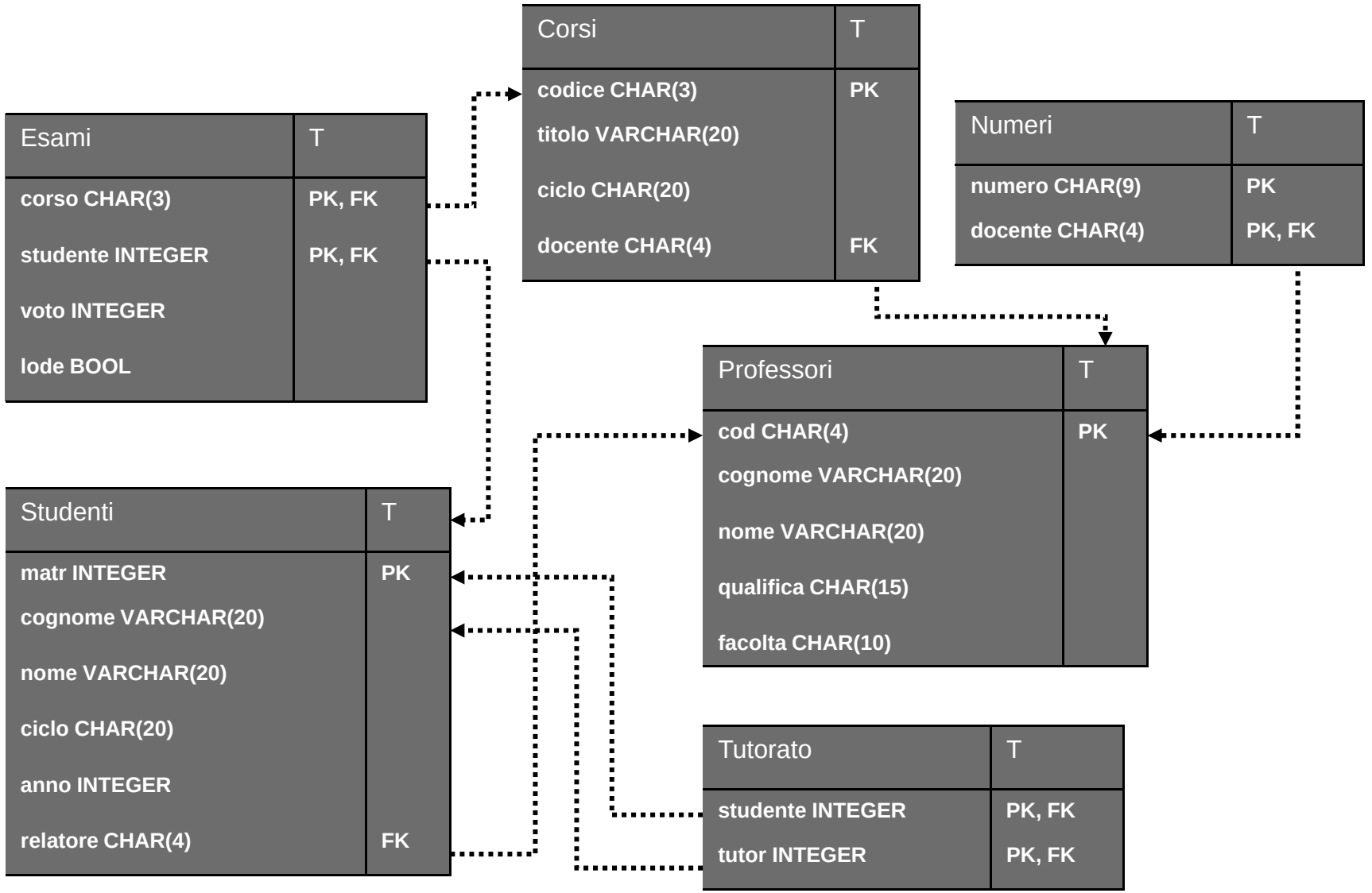
Concetti Avanzati

- **Raggruppamenti**
  - Clausole GROUP BY e HAVING
  - Forma Generale della SELECT
- **Nidificazione**
  - Uso nel DML e DDL
  - Nidificazione, Viste e Potere Espressivo
- **Esecuzione di una Query SQL**

```
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