Basi di Dati
Esempi di prove di verifica con soluzioni

Prima prova di verifica del 6/11/2001

1. Una rivista periodica di fumetti vuole memorizzare informazioni relative a tutte le storie che ha pubblicato nel passato, ed ai relativi personaggi. Di una storia interessa il titolo, che la identifica, ed interessano informazioni relative alle puntate in cui è stata divisa: per ogni puntata interessa il numero di pagine, il numero d'ordine all'interno della storia (prima, seconda ...) ed il numero della rivista su cui è stata pubblicata. I personaggi si dividono in principali e secondari. Per tutti i personaggi interessa il nome, che li identifica. Per i personaggi secondari interessa ricordare le storie in cui sono apparsi, mentre per quelli principali si vogliono memorizzare precisamente le puntate di apparizione. Se due personaggi sono parenti, se ne memorizza la relazione di parentela (ovvero, il fatto che sono parenti ed anche il grado di parentela).

   (a) Si definisca lo schema concettuale della base di dati.

   (b) Si traduca lo schema concettuale in uno schema relazionale grafico

2. Si consideri il seguente schema relazionale:

   Attori(CodiceAtt, Nome, AnnoNascita);
   AttoriFilm(CodiceAtt*, CodiceFilm*)
   Film(CodiceFilm, Titolo, AnnoProduzione, Regista)
   Proiezioni(CodiceFilm*, CodiceSala*, Incasso, DataProiezione)
   Sala(CodiceSala, Posti, Nome, Città)

   Si scrivano in SQL le seguenti interrogazioni.

   (a) Per ogni film in cui appare un attore nato nel 1970 restituire il titolo del film e il nome dell'attore

(b) Restituire, senza duplicati, il CodiceFilm di tutti i film per i quali c'è stata una proiezione con incasso maggiore di un milione in una sala che avesse meno di 100 posti oppure che si trovasse a Pisa

(c) Per ogni film che è stato proiettato a Pisa restituire il titolo del film e il nome della sala, evitando le duplicazioni nella risposta

(d) Per ogni film in cui appaiono due attori diversi nati lo stesso anno trovare il codice del film ed i nomi dei due attori

(e) Trovare il titolo di tutti i film prodotti prima del 1975.

(f) Per ogni film di 'John Waters' restituire il numero totale di proiezioni a Pisa e l'incasso totale (sempre a 'Pisa')

(g) Per ogni regista, restituire il nome e l'incasso totale di tutte le proiezioni dei suoi film

(h) Per ogni città restituire il nome della città ed il numero di sale con più di 100 posti

Soluzione prima prova di verifica del 6/11/2001

1. Il progetto concettuale è in Figura 1 e quello relazionale in Figura 2.

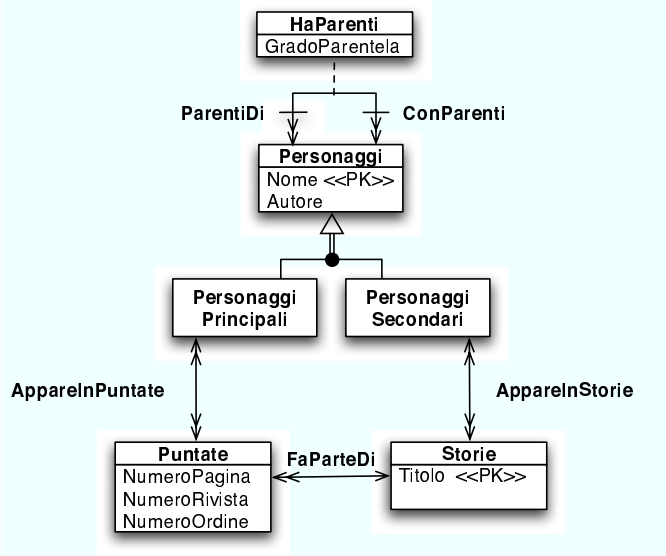


Figura 1: Schema concettuale

2. Interrogazioni

   (a) Per ogni film in cui appare un attore nato nel 1970 restituire il titolo del film e il nome dell'attore

```
SELECT    f.Titolo, a.Nome
FROM      Attori a, AttoriFilm af, Film f
WHERE     a.CodiceAtt = af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm
          AND a.AnnoNascita = 1970;
```

   (b) Restituire, senza duplicati, il CodiceFilm di tutti i film per i quali c'è stata una proiezione con incasso maggiore di un milione in una sala che avesse meno di 100 posti oppure che si trovasse a Pisa

```
SELECT    DISTINCT p.CodiceFilm
FROM      Proiezioni p, Sale s
WHERE     p.CodiceSala=s.CodiceSala AND p.Incasso > 1000000
          AND (s.Posti < 100 OR s.Città = 'Pisa' );
```

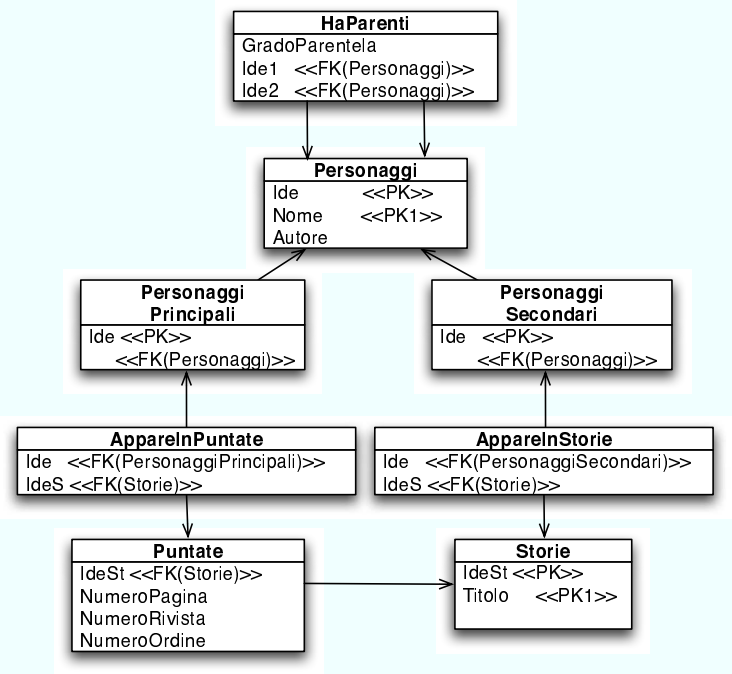


Figura 2: Schema relazionale

(c) Per ogni film che è stato proiettato a Pisa restituire il titolo del film e il nome della sala, evitando le duplicazioni nella risposta

```
SELECT    DISTINCT f.Titolo, s.Nome
FROM      Film f, Proiezioni p, Sale s
WHERE     f.CodiceFilm = s.CodiceFilm AND p.CodiceSala=s.CodiceSala
          AND s.Città = 'Pisa';
```

(d) Per ogni film in cui appaiono due attori diversi nati lo stesso anno trovare il codice del film ed i nomi dei due attori

```
SELECT    af1.CodiceFilm, a1.Nome, a2.Nome
FROM      Attori a1, AttoriFilm af1, AttoriFilm af2, Attori af2
WHERE     a1.CodiceAtt = af1.CodiceAtt AND af1.CodiceFilm = af2.CodiceFilm
          AND af2.CodiceAtt = a2.CodiceAtt AND a1.AnnoNascita = a2.AnnoNascita
          AND a1.CodiceAtt < a2.CodiceAtt ;
```

(e) Trovare il titolo di tutti i film prodotti prima del 1975.

```
SELECT      f.Titolo
FROM        Film f
WHERE       f. AnnoProduzione <1975 ;
```

(f) Per ogni film di 'John Waters' restituire il numero totale di proiezioni a Pisa e l'incasso totale (sempre a Pisa)

```
SELECT      COUNT(*), SUM(p.Incasso)
FROM        Film f, Proiezioni p, Sale s
WHERE       f.CodiceFilm = p.CodiceFilm AND p.CodiceSala = s.CodiceSala
            AND f.Regista = 'John Waters' AND s.Città = 'Pisa'
GROUP BY    f.CodiceFilm ;
```

(g) Per ogni regista, restituire il nome e l'incasso totale di tutte le proiezioni dei suoi film

```
SELECT      f.Regista, SUM(p.Incasso)
FROM        Film f, Proiezioni p
WHERE       f.CodiceFilm = p.CodiceFilm
GROUP BY    f.Regista ;
```

(h) Per ogni città restituire il nome della città ed il numero di sale con più di 100 posti

```
SELECT      s.Città, COUNT(*)
FROM        Sale s
WHERE       s.Posti > 100
GROUP BY    s.Città ;
```

Seconda prova di verifica del 20/12/2001

1. Si consideri il seguente schema relazionale:

   Attori(CodiceAtt, Nome, AnnoNascita);
   AttoriFilm(CodiceAtt*, CodiceFilm*)
   Film(CodiceFilm, Titolo, AnnoProduzione, Regista)
   Proiezioni(CodiceFilm*, CodiceSala*, Incasso, DataProiezione)
   Sala(CodiceSala, Posti, Nome, Città)

   Si scrivano in SQL le seguenti interrogazioni:

   (a) Per ogni film in cui appaiono solo attori nati prima del 1970 restituire il titolo del film

   (b) Per ogni film in cui appaiono solo attori nati prima del 1970 restituire il regista del film ed il numero di attori

   (c) Scrivere le due interrogazioni seguenti:

   - Restituire il nome ed il codice di tutte le sale di Pisa in cui ogni proiezione del film con codice 100 **che è avvenuta** il 10/12/2001 ha incassato almeno 300.000 Lire
   - Restituire il nome ed il codice di tutte le sale di Pisa in cui ogni proiezione del film con codice 100 **è avvenuta** il 10/12/2001 ed ha incassato almeno 300.000 Lire

2. Per tenere traccia delle allocazioni delle aule di un polo didattico durante una certa settimana è necessario trattare i seguenti attributi: OraInizioLezione, OraFineLezione, GiornoSettimana, NomeCorso (es: Analisi), LetteraCorso (es: A, B, C, Unico), TitolareCorso, Aula, PianoAula (es.: Primo, Secondo), CorsoDiLaureaDelCorso. Lo stesso nome di corso può essere usato, con la stessa lettera, in diversi corsi di laurea. Specificate, per ciascuna delle seguenti affermazioni, la dipendenza funzionale che ne deriva: (2.1) Tutte le lezioni durano lo stesso tempo (2.2) Ogni corso ha un solo titolare (2.3) Non si possono tenere due corsi contemporaneamente nella stesa aula (2.4) Uno stesso docente non può essere in due aule contemporaneamente.

3. Si consideri il seguente insieme di dipendenze funzionali in forma canonica:

   R(A,B,C,D,E), $\{AD \rightarrow B, CB \rightarrow A, DE \rightarrow A, A \rightarrow E\}$

   (a) Trovate tutte le chiavi

   (b) Dite se lo schema è in 3FN o in BCNF

   (c) Applicate l'algoritmo di sintesi

4. Si consideri lo schema dell'esercizio (1) e, per l'interrogazione seguente, si fornisca un piano di accesso efficiente, supponendo che sia presente un indice sugli attributi Film(CodiceFilm), Proiezioni(CodiceFilm).

```
SELECT      f.Titolo, COUNT(*), SUM(p.Incasso)
FROM        Film f, Proiezioni p
WHERE       f.CodiceFilm = p.CodiceFilm AND f.Regista = 'John Waters'
GROUP BY    f.CodiceFilm, f.Titolo
HAVING      SUM(p.Incasso) > 100 ;
```

5. Per l'interrogazione seguente, si forniscano due piani di accesso, uno che a vostro parere sarebbe particolarmente efficiente, ed uno che vi pare molto inefficiente, supponendo che sia presente un indice sugli attributi Attori(CodiceAtt), AttoriFilm(CodiceAtt), AttoriFilm(CodiceFilm), Film(CodiceFilm)

```
SELECT      f.Titolo, a.Nome
FROM        Attori a, AttoriFilm af, Film f
WHERE       a.CodiceAtt = af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm
            AND a.AnnoNascita = 1970 ;
```

Soluzione seconda prova di verifica del 20/12/2001

1. Interrogazioni

   (a) Per ogni film in cui appaiono solo attori nati prima del 1970 restituire il titolo del film

```
SELECT  f.Titolo
FROM    Film f
WHERE   (FOR ALL af IN AttoriFilm, a IN Attori
         WHERE   a.CodiceAtt=af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm
          : a.AnnoNascita < 1970);
```

```
SELECT  f.Titolo
FROM    Film f
WHERE   NOT EXISTS (
                        SELECT  *
                        FROM    AttoriFilm af, Attori a
                        WHERE   a.CodiceAtt=af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm
                                AND NOT(a.AnnoNascita < 1970) );
```

```
SELECT  f.Titolo
FROM    Film f
WHERE   1970 >ALL (
                        SELECT  a.AnnoNascita
                        FROM    AttoriFilm af, Attori a
                        WHERE   a.CodiceAtt=af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm);
```

   (b) Per ogni film in cui appaiono solo attori nati prima del 1970 restituire il regista del film ed il numero di attori

```
SELECT      f.Regista, COUNT(*) AS NumeroAttori
FROM        Attori a, AttoriFilm af, Film f
WHERE       a.CodiceAtt=af.CodiceAtt AND af.CodiceFilm = f.CodiceFilm
GROUP BY    f.CodiceFilm, f.Regista
HAVING      MAX(a.AnnoNascita) < 1970 ;
```

   (c) Scrivere le due interrogazioni seguenti:

   - Restituire il nome ed il codice di tutte le sale di Pisa in cui ogni proiezione del film con codice 100 **che è avvenuta** il 10/12/2001 ha incassato almeno 300.000 Lire

```
SELECT   s.Nome, s.CodiceSala
FROM     Sale s
WHERE    s.Citta = 'Pisa' AND
         (FOR ALL p IN Proiezioni
         WHERE   p.CodiceSala=s.CodiceSala AND p.CodiceFilm = 100
                 AND p.DataProiezione = '10/12/2001'
          : p.Incasso >= 300000);
```

```
SELECT   s.Nome, s.CodiceSala
FROM     Sale s
WHERE    s.Citta = 'Pisa' AND
         NOT EXISTS (
                        SELECT   *
                        FROM     Proiezioni p
                        WHERE    p.CodiceSala=s.CodiceSala AND p.CodiceFilm = 100
                                 AND p.DataProiezione = '10/12/2001'
                                 AND NOT(p.Incasso >= 300000));
```

- Restituire il nome ed il codice di tutte le sale di Pisa in cui ogni proiezione del film con codice 100 **è avvenuta** il 10/12/2001 ed ha incassato almeno 300.000 Lire

```
SELECT   s.Nome, s.CodiceSala
FROM     Sale s
WHERE    s.Citta = 'Pisa' AND
         (FOR ALL p IN Proiezioni
         WHERE   p.CodiceSala=s.CodiceSala AND p.CodiceFilm = 100
          : p.DataProiezione = '10/12/2001' AND p.Incasso >= 300000);
```

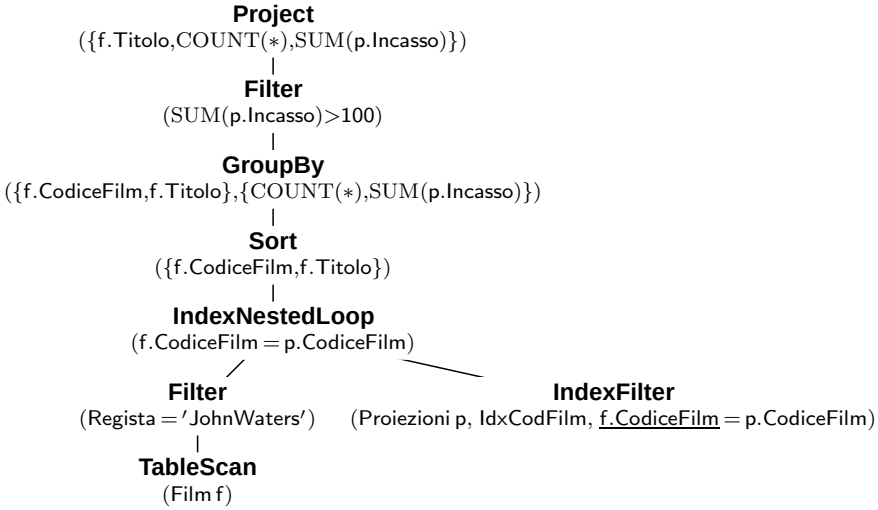
```
SELECT   s.Nome, s.CodiceSala
FROM     Sale s
WHERE    s.Citta = 'Pisa' AND
         NOT EXISTS (
                        SELECT   *
                        FROM     Proiezioni p
                        WHERE    p.CodiceSala=s.CodiceSala AND p.CodiceFilm = 100
                                 AND NOT    (p.DataProiezione = '10/12/2001'
                                             AND p.Incasso >= 300000));
```

2.
- OraInizioLezione → OraFineLezione, OraFineLezione → OraInizioLezione
- CorsoDiLaureaDelCorso, NomeCorso, LetteraCorso → TitolareCorso
- GiornoSettimana, OraInizioLezione, Aula → NomeCorso
- GiornoSettimana, OraInizioLezione, TitolareCorso → Aula

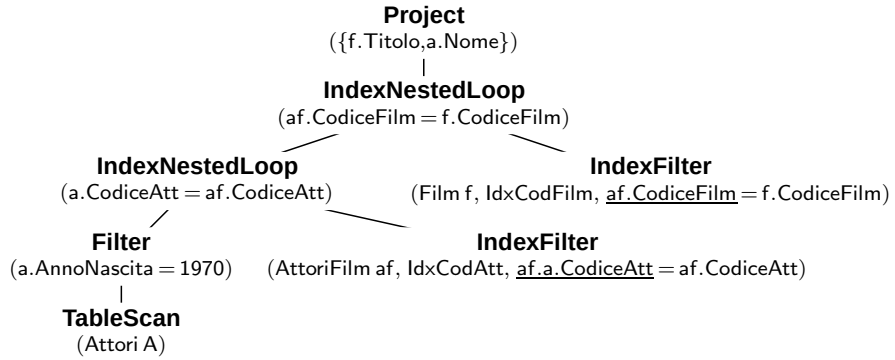
3.
- Le chiavi: CDA, CDB, CDE
- Lo schema è solo in 3FN

- Algoritmo di sintesi:
  Raggruppo gli attributi: R1(ADB), R2(CBA), R3(DEA), R4(AE)
  Elimino la relazione in più: R1(ADB), R2(CBA), R3(DEA)
  Aggiungo la chiave: R1(ADB), R2(CBA), R3(DEA), R4(CDA)

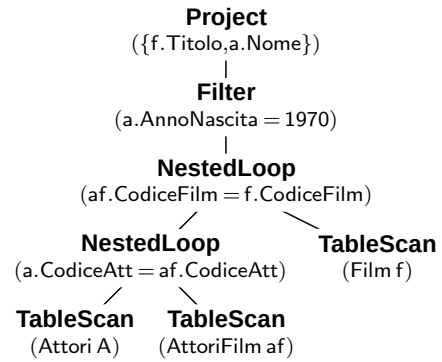
4. Un piano efficiente



5. Un piano efficiente



Un piano inefficiente



Prima prova di verifica del 3/11/2003

1. Una compagnia che gestisce un piccolo motore di ricerca sul web vuole usare una base di dati per tenere traccia della struttura delle pagine indicizzate, e delle sessioni che gli utenti hanno con il motore di ricerca. Per quanto riguarda la struttura delle pagine indicizzate, interessa conoscere, per ogni pagina, l'URL, il titolo, l'insieme delle pagine puntate dai riferimenti (anchor–link) che appaiono nella pagina, l'insieme dei termini che vi appaiono e, per ogni termine, al numero di volte in cui il termine appare nella pagina. Le sessioni sono così strutturate: l'utente presenta un termine di ricerca, il sistema segnala tutte le pagine correlate, e l'utente legge alcune di queste pagine; specifichiamo ora a quali informazioni la compagnia è interessata, per quanto riguarda le sessioni. Per ogni sessione, identificata da un codice, l'organizzazione è interessata al giorno in cui si è svolta ed al termine di ricerca presentato. La sessione può essere eseguita da un utente anonimo, sul quale il sistema non ha informazioni, oppure da un utente registrato, del quale il sistema conosce username, password, indirizzo e-mail, e l'elenco di tutti gli indirizzi IP dai quali l'utente si è collegato nel passato. Per le sessioni con utenti registrati interessa memorizzare anche l'ora di inizio e di fine, l'utente, e la liste delle pagine effettivamente lette dall'utente.

   (a) Si definisca lo schema concettuale della base di dati.

   (b) Si traduca lo schema concettuale in uno schema relazionale grafico

2. Si consideri il seguente schema relazionale:

   Studenti(Matricola: integer, SNome: string, CorsoLaurea: string, TipoLaurea: string, Età: integer)
   Corsi(SiglaC: string, OraRicevimento: time, Aula: string, DocenteId*: integer)
   Iscrizioni(Matricola*: integer, SiglaC*: string)
   Docenti(DocenteId: integer, DNome: string, Dipartimento: string)

   Si scrivano in SQL le seguenti interrogazioni per produrre risultati privi di duplicati:

   (a) Trovare i nomi degli studenti di un corso di laurea (TipoLaurea = 'L') che seguono un corso del docente Tizio

   (b) Trovare la sigla dei corsi frequentati da più di 5 studenti e tenuti da docenti del Dipartimento di Informatica.

   (c) Trovare i nomi degli studenti iscritti a qualche corso.

   (d) Per ogni tipo di laurea, restituire il TipoLaurea e l'età media degli studenti.

   (e) Trovare il nome e l'età dello studente più anziano (o degli studenti più anziani) tra quelli iscritti ad una qualunque laurea specialistica (TipoLaurea = 'LS').

   (f) (Opzionale) Per ogni coppia di corsi che hanno almeno dieci studenti in comune, trovare la sigla e l'aula dei due corsi ed il numero di studenti in comune.

(g) Disegnare l’albero di sintassi astratta dell’espressione algebrica (albero logico) per la quarta interrogazione.

Soluzione prima prova di verifica del 3/11/2003

1. Il progetto concettuale è in Figura 3 e quello relazionale in Figura 4.

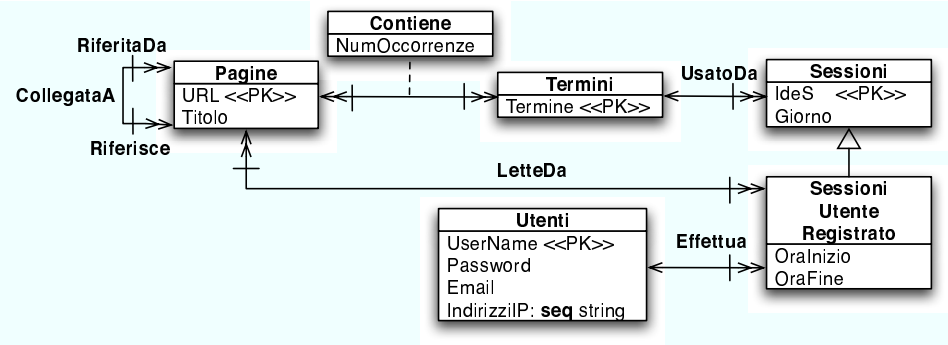


Figura 3: Schema concettuale

Vincoli: Ogni pagina letta durante una sessione è una delle pagine in cui appare il termine cercato nella sessione Osservazione: Non c'è motivo di inserire classi per gli utenti non registrati.

2. Interrogazioni

   (a) Trovare i nomi degli studenti di un corso di laurea (TipoLaurea = 'L') che seguono un corso del docente Tizio

```
SELECT   DISTINCT s.SNome
FROM     Iscrizioni i, Corsi c, Docenti d
WHERE    i.SiglaC = c.SiglaS AND and i.SiglaC = c.SiglaS AND c.DocenteId = d.DocenteId
         AND s.TipoLaurea = 'L' AND d.DNome = 'Tizio' ;
```

   (b) Trovare la sigla dei corsi frequentati da più di 5 studenti e tenuti da docenti del Dipartimento di Informatica.

```
SELECT      c.SiglaC
FROM        Iscrizioni i, Corsi c, Docenti d
WHERE       i.SiglaC = c.SiglaS AND c.DocenteId = d.DocenteId
            AND d.Dipartimento = 'Informatica'
GROUP BY    c.SiglaC
HAVING      COUNT(*) > 5;
```

   (c) Trovare i nomi degli studenti iscritti a qualche corso.

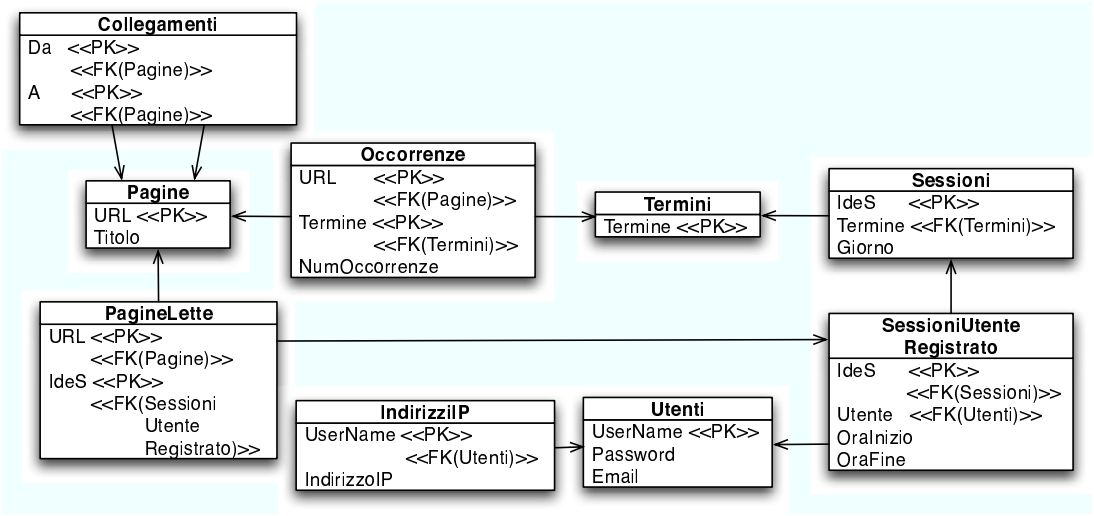


Figura 4: Schema relazionale

```
SELECT    DISTINCT s.SNome
FROM      Studenti s, Iscrizioni i
WHERE     s.Matricola = i.Matricola;
```

(d) Per ogni tipo di laurea, restituire il TipoLaurea e l'età media degli studenti.

```
SELECT      s.TipoLaurea, AVG(s.Età)
FROM        Studenti s
GROUP BY    s.TipoLaurea;
```

(e) Trovare il nome e l'età dello studente più anziano (o degli studenti più anziani) tra quelli iscritti ad una qualunque laurea specialistica (TipoLaurea = 'LS').

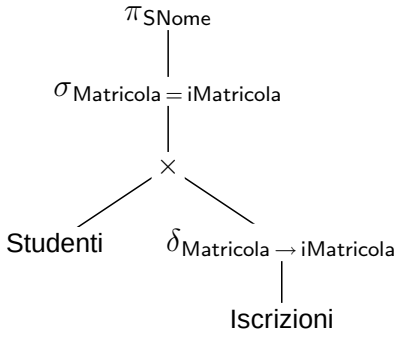
```
SELECT   DISTINCT s.SNome, s.Età
FROM     Studenti s
WHERE    s.TipoLaurea = 'LS' AND
         s.Età = (
                  SELECT   MAX(s2.Età)
                  FROM     Studenti s2
                  WHERE    s2.TipoLaurea = 'LS' );
```

```
SELECT   DISTINCT s.SNome, s.Età
FROM     Studenti s
WHERE    s.TipoLaurea = 'LS' AND
         NOT EXISTS (
                        SELECT   *
                        FROM     Studenti s2
                        WHERE    s2.TipoLaurea = 'LS' AND s2.Età > s.Età );
```

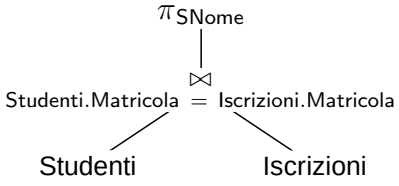
(f) (Opzionale) Per ogni coppia di corsi che hanno almeno dieci studenti in comune, trovare la sigla e l'aula dei due corsi ed il numero di studenti in comune.

```
SELECT      c1.SiglaC, c1.Aula, c2.SiglaC, c2.Aula, COUNT(*)
FROM        Corsi c1, Iscrizioni i1, Iscrizioni i2, Corsi c2
WHERE       c1.SiglaC = i1.SiglaC AND i1.Matricola = i2.Matricola
            AND i2.siglaC = c2.SiglaC AND c1.SiglaC <> c2.SiglaC
GROUP BY    c1.SiglaC, c1.Aula, c2.SiglaC, c2.Aula
HAVING      COUNT(*) >=10;
```

(g) Disegnare l'albero di sintassi astratta dell'espressione algebrica (albero logico) per la quarta interrogazione.



oppure



Seconda prova di verifica del 17/12/2003

1. Si consideri lo schema relazionale R(A,B,C,D,E) con le seguenti DF:

   $A \rightarrow BC$, $CB \rightarrow A$, $CD \rightarrow A$, $D \rightarrow B$

   (a) Si trovino le chiavi di R.

   (b) Dire se lo schema è in 3FN o in FNBC.

   (c) Si applichi allo schema l'algoritmo di sintesi, e si specifichi se il risultato preserva dati e dipendenze, e in che forma normale si trova.

   (d) (Opzionale) Si applichi allo schema l'algoritmo di analisi, e si specifichi se il risultato preserva dati e dipendenze, e in che forma normale si trova.

2. Si consideri lo schema relazionale:

   CorsiDiLaurea(CorsoLaurea: integer, Facoltà: string, TipoLaurea: string)
   Studenti(Matricola: integer, SNome: string, CorsoLaurea*: string)
   Corsi(SiglaC: string, OraRicevimento: time, Aula: string, DocenteId*: integer)
   Iscrizioni(Matricola*: integer, SiglaC*: string)
   Docenti(DocenteId: integer, DNome: string, Dipartimento: string)

   Si scrivano in SQL le seguenti interrogazioni:

   (a) Trovare, senza duplicati, il DocenteId dei corsi che non sono frequentati da nessuno studente

   (b) Trovare la sigla dei corsi seguiti solo da studenti che appartengono al corso di laurea 'Informatica'

   (c) Di ogni studente della Facoltà 'Scienze', trovare la matricola ed il numero di corsi che segue

   (d) Trovare nome e DocenteId dei docenti che insegnano qualche corso seguito da più di 5 studenti

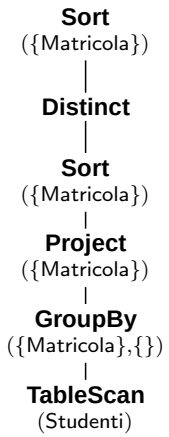
   (e) (Opzionale) Trovare la sigla dei corsi che sono frequentati da tutti gli studenti della Facoltà 'Scienze'

3. Si consideri la relazione Studenti(Matricola, Nome, AnnoNascita), ordinata sulla chiave primaria Matricola, e l'interrogazione

```
SELECT     DISTINCT Matricola, COUNT(*)
FROM       Studenti
WHERE      AnnoNascita = 1974
GROUP BY   Matricola
ORDER BY   Matricola ;
```

(a) Disegnare l'albero di sintassi astratta di un'espressione algebrica (albero logico) per l'interrogazione.

(b) Si dica se il seguente piano d'accesso a destra produce il risultato cercato. Se non va bene, lo si modifichi in due modi: (b) aggiungendo prima solo le parti mancanti e (c) semplificando poi il piano eliminando operatori inutili.



4. (Opzionale) Si consideri il seguente schema, che rappresenta informazioni relative agli spettacoli programmati per una stagione in un insieme di teatri:

   Spettacoli(Compagnia, Regista, Titolo, Data, Teatro)

   Si esprimano i seguenti vincoli come dipendenze funzionali, se possibile:

   (a) Due spettacoli contemporanei hanno il regista diverso.

   (b) Non è possibile avere due spettacoli diversi nella stessa data e nello stesso teatro.

   (c) Se due spettacoli hanno il regista diverso anche le compagnie sono diverse.

Soluzione seconda prova di verifica del 17/12/2003

1. (a) DE devono essere in tutte le chiavi. DEA e DEC sono le due chiavi.

   (b) Non è né in 3FN né in FNBC

   (c) Porto le dipendenze in forma canonica.
   Non ci sono attributi estranei, perché $C^+ = C$, $B^+ = B$, $D^+ = DB$
   Elimino le ridondanze.
   $A \rightarrow BC$: $A^+_{\{CB\rightarrow A, CD\rightarrow A, D\rightarrow B\}} = A$
   $CB \rightarrow A$: $CB^+_{\{A\rightarrow BC, CD\rightarrow A, D\rightarrow B\}} = CB$
   $CD \rightarrow A$: $CD^+_{\{A\rightarrow BC, CB\rightarrow A, D\rightarrow B\}} = CDBA$. $CD \rightarrow A$ è ridondante.
   $D \rightarrow B$: $D^+_{\{A\rightarrow BC, CB\rightarrow A\}} = D$

   i. R1(ABC) $\{A \rightarrow BC\}$, R2(BCA) $\{BC \rightarrow A\}$, R3(DB) $\{D \rightarrow B\}$,
   ii. Elimino R2 e porto la dipendenza $BC \rightarrow A$ su R1:
   R1(ABC), $\{A \rightarrow BC, BC \rightarrow A\}$, R3(DB), $\{D \rightarrow B\}$,
   iii. Aggiungo una chiave: DEA:
   R1(ABC) $\{A \rightarrow BC, BC \rightarrow A\}$, R3(DB) $\{D \rightarrow B\}$, R4(DEA) $\{\}$
   dati e dipendenze sono preservati, ed il risultato è in FNBC.

   (d) (Opzionale) R(ABCDE): parto da $A \rightarrow BC$

   i. R1(ABC) $\{A \rightarrow BC, BC \rightarrow A\}$ + R2(ADE) $\{\}$
   La dipendenza $D \rightarrow B$ va perduta (la proiezione di F su ABC e su ADE non produce altre dipendenze). I dati sono preservati e il risultato è in FNBC.

   Oppure: R(ABCDE): parto da $CB \rightarrow A$

   i. R1(ABC) $\{A \rightarrow BC, BC \rightarrow A\}$ + R2(BCDE) $\{D \rightarrow B\}$
   ii. R1(ABC) $\{A \rightarrow BC, BC \rightarrow A\}$ + R3(BD) $\{D \rightarrow B\}$ + R4(DCE) $\{\}$
   Nessuna dipendenza va perduta. I dati sono preservati e il risultato è in FNBC.
   Altre soluzioni sono possibili.

2. Interrogazioni:

   (a) Trovare, senza duplicati, il DocenteId dei corsi che non sono frequentati da nessuno studente

```
SELECT   DISTINCT C.DocenteId
FROM     Corsi C
WHERE    NOT EXISTS (
                        SELECT   *
                        FROM     Iscrizioni I
                        WHERE    C.SiglaC = I.Matricola );
```

```
SELECT   DISTINCT C.DocenteId
FROM     Corsi C
WHERE    C.SiglaC NOT IN (
                              SELECT   I.SiglaC
                              FROM     Iscrizioni I);
```

(b) Trovare la sigla dei corsi seguiti solo da studenti che appartengono al corso di laurea 'Informatica'

Si osservi che questa soluzione riporta anche i corsi che non sono seguiti da nessun studente.

```
SELECT   C.SiglaC
FROM     Corsi C
WHERE    (FOR ALL I IN Iscrizioni, S IN Studenti
          WHERE   C.SiglaC = I.SiglaC AND I.Matricola = S.Matricola
           : S.CorsoLaurea = 'Informatica');
```

```
SELECT   C.SiglaC
FROM     Corsi C
WHERE    NOT EXISTS (
                          SELECT   *
                          FROM     Iscrizioni I, Studenti S
                          WHERE    C.SiglaC = I.SiglaC AND I.Matricola = S.Matricola
                                   AND NOT(S.CorsoLaurea = 'Informatica'));
```

(c) Di ogni studente della Facoltà 'Scienze', trovare la matricola ed il numero di corsi che segue

Questa soluzione riporta solo gli studenti che seguono almeno un corso.

```
SELECT       DISTINCT S.Matricola, COUNT(*)
FROM         CorsiDiLaurea C, Studenti S, Iscrizioni I
WHERE        C.Facoltà = 'Scienze' AND C.CorsoLaurea = S.CorsoLaurea
             AND S.Matricola = I.Matricola
GROUP BY     S.Matricola;
```

(d) Trovare nome e DocenteId dei docenti che insegnano qualche corso seguito da più di 5 studenti

```
SELECT       DISTINCT D.DocentiId, D.DNome
FROM         Docenti D, Corsi C, Iscrizioni I
WHERE        D.DocenteId = C.DocenteId AND C.SiglaC = I.SiglaC
GROUP BY     C.SiglaC, D.DocentiId, D.DNome
HAVING       COUNT(*) > 5;
```

```
SELECT  DISTINCT D.DocentiId, D.DNome
FROM    Docenti D, Corsi C
WHERE   D.DocenteId = C.DocenteId
        AND 5 < (
                SELECT  COUNT(*)
                FROM    Iscrizioni I
                WHERE   C.SiglaC = I.SiglaC);
```

```
SELECT  DISTINCT D.DocentiId, D.DNome
FROM    Docenti D, Corsi C
WHERE   D.DocenteId = C.DocenteId
        AND EXISTS (
                SELECT      I.SiglaC
                FROM        Iscrizioni I
                WHERE       C.SiglaC = I.SiglaC
                GROUP BY    I.SiglaC
                HAVING      COUNT(*) > 5);
```

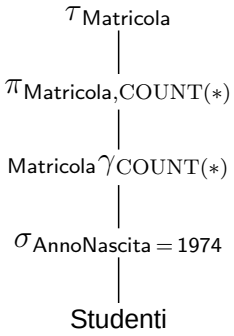
(e) (Opzionale) Trovare la sigla dei corsi che sono frequentati da tutti gli studenti della Facoltà 'Scienze'

```
SELECT  C.SiglaC
FROM    Corsi C
WHERE   (FOR ALL S IN Studenti, Cdl IN CorsiDiLaurea
        WHERE   S.CorsoLaurea = Cdl.CorsoLaurea AND Cld.Facoltà = 'Scienze'
         : (FOR SOME I IN Iscrizioni WHERE I.SiglaC = C.SiglaC AND I.Matricola = S.Matricola));
```

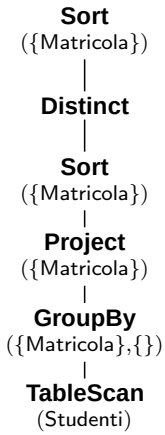
```
SELECT  SiglaC
FROM    Corsi C
WHERE   NOT EXISTS (
                SELECT  *
                FROM    Studenti S, CorsiDiLaurea Cdl
                WHERE   S.CorsoLaurea = Cdl.CorsoLaurea
                        AND Cld.Facoltà = 'Scienze' AND
                        NOT EXISTS (
                                SELECT  *
                                FROM    Iscrizioni I
                                WHERE   I.SiglaC = C.SiglaC
                                        AND I.Matricola = S.Matricola));
```

```
SELECT  SiglaC
FROM    Corsi C
WHERE   NOT EXISTS (
                    SELECT  *
                    FROM    Studenti S, CorsiDiLaurea Cdl
                    WHERE   S.CorsoLaurea = Cdl.CorsoLaurea
                            AND Cld.Facoltà = 'Scienze'
                            AND S.Matricola NOT IN (
                                                    SELECT  I.Matricola
                                                    FROM    Iscrizioni I
                                                    WHERE   I.SiglaC = C.SiglaC));
```

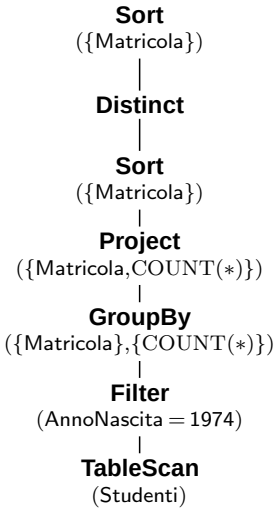
3. Albero logico



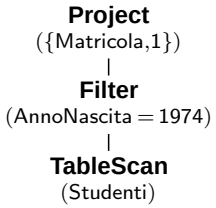
Piano errato



Piano modificato corretto



Piano semplificato togliendo operatori inutili e tenendo conto che la relazione è già ordinata su Matricola e che, essendo Matricola chiave, è inutile raggruppare per Matricola.



4. Opzionale

   (a) Due spettacoli contemporanei hanno il regista diverso

   $\text{Spettacolo}_{\neq} \wedge \text{Data}_{=} \Rightarrow \text{Regista}_{\neq}$

   $\text{Regista}_{=} \wedge \text{Data}_{=} \Rightarrow \text{Spettacolo}_{=}$

   $\text{Regista, Data} \longrightarrow \text{Compagnia, Titolo, Teatro}$

(b) Non è possibile avere due spettacoli diversi nella stessa data e nello stesso teatro

$\text{Spettacolo}_{\neq} \wedge \text{Data}_{=} \wedge \text{Teatro}_{=} \Rightarrow \text{False}$

$\text{Data}_{=} \wedge \text{Teatro}_{=} \Rightarrow \text{Spettacolo}_{=}$

$\text{Data, Teatro} \rightarrow \text{Compagnia, Regista, Titolo}$

(c) Se due spettacoli hanno il regista diverso anche le compagnie sono diverse

$\text{Regista}_{\neq} \Rightarrow \text{Compagnia}_{\neq}$

$\text{Compagnia}_{=} \Rightarrow \text{Regista}_{=}$

$\text{Compagnia} \rightarrow \text{Regista}$