Basi di dati — 13 novembre 2017 — Prova parziale — Compito A

Possibili soluzioni

Tempo a disposizione: un'ora.

Cognome: _____ **Nome:** _____ **Matricola:** _____

Domanda 1 (25%) Considerare lo schema con le seguenti relazioni

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

con le seguenti cardinalità

- `studenti`: cardinalità $S = 1000$
- `corsi`: cardinalità $C = 200$
- `esami`: cardinalità $E = 10.000$

Indicare la cardinalità del risultato di ciascuna delle seguenti interrogazioni SQL, specificando l'intervallo nel quale essa può variare; indicare simboli e numeri.

	Min (simboli e valore)	Max (simboli e valore)
`SELECT matricola, codice` `FROM studenti, corsi`	$S \times C$ 200.000	$S \times C$ 200.000

`SELECT *`
`FROM studenti JOIN esami`
`  ON matricola = studente`
`WHERE voto > 27`
	0
	0
		E
		10.000

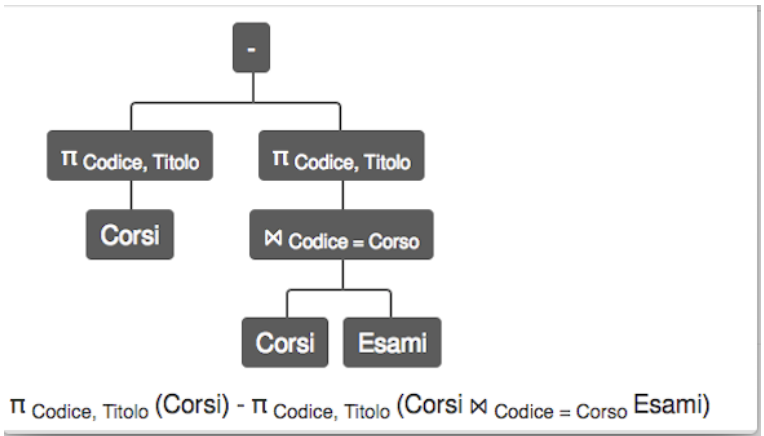
`SELECT matricola, codice`
`FROM studenti, esami, corsi`
`WHERE matricola = studente`
`  AND corso = codice`
`  AND voto > 24`
	0
	0
		E
		10.000

Domanda 2 (55%) Considerare nuovamente lo schema utilizzato nella domanda precedente

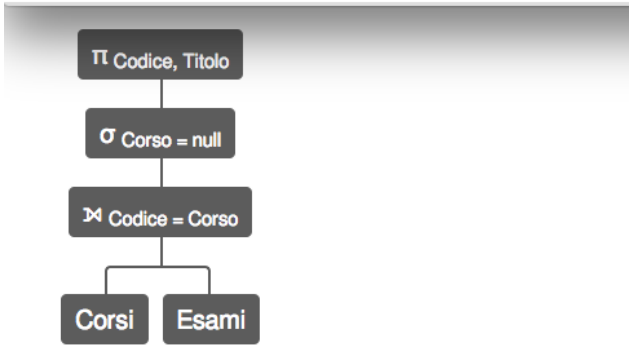
```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

Formulare la seguente interrogazione in **algebra relazionale**

1. Mostrare codice e titolo dei corsi per i quali non è stato registrato nessun esame.



$\pi_{\text{Codice, Titolo}}(\text{Corsi}) - \pi_{\text{Codice, Titolo}}(\text{Corsi} \bowtie_{\text{Codice = Corso}} \text{Esami})$



$\pi_{\text{Codice, Titolo}}(\sigma_{\text{Corso = null}}(\text{Corsi} \bowtie_{\text{Codice = Corso}} \text{Esami}))$

Più correttamente si dovrebbe scrivere 'is null' invece di '= null'

Formulare le seguenti interrogazioni in **SQL**

2. Per ciascuno studente, mostrare matricola, numero di CFU conseguiti e media dei voti.

```
select studente, sum(CFU) as CFU, avg(voto)::numeric(4,1) as media
from esami join corsi on corso = codice
group by studente
```

3. Per ciascun corso, mostrare codice, numero di esami registrati e media dei voti.

```
select corso as codice, count(*) as numeroEsami, avg(voto)::numeric(4,1) as media
from esami
group by corso
```

4. Per ciascun corso, mostrare codice e numero di esami registrati, includendo anche i corsi per i quali non è stato registrato alcun esame (con il valore 0 per il numero di esami).

```
select corso, count(*) as numeroEsami
from esami
group by corso
union
select codice as corso, 0 as numeroEsami
from corsi
where not exists (select * from esami where corso = codice)

OPPURE

select codice, count(corso) as numeroEsami
from corsi left join esami ON codice=corso
group by codice
```

5. Mostrare matricola, cognome e nome dello studente che ha conseguito il maggior numero di credti.

```
create view totaleCFU as
  select studente, sum(CFU) as CFU
  from esami join corsi on corso = codice
  group by studente;

select matricola, cognome, nome, CFU
from studenti join totaleCFU on matricola = studente
where CFU = (select max(CFU) from totaleCFU)
```

Domanda 3 (20%)
Considerare una relazione

STIPENDI(Matricola,StipLordo,Tasse,Netto,Verifica)

e definire su di essa

1. un vincolo che imponga che, se il valore di Verifica è "OK", allora Netto è uguale alla differenza fra StipLordo e Tasse (si noti che non si vuole invece imporre nessuna condizione se il valore di Verifica è diverso da "OK").

```
CHECK ( (NOT (Verifica = 'OK') ) OR (Netto = StipLordo - Tasse) )
```

2. un vincolo che imponga che il valore di Verifica è "OK" se e solo se Netto è uguale alla differenza fra StipLordo e Tasse.

```
CHECK ( (NOT (Verifica = 'OK') AND NOT (Netto = StipLordo - Tasse) )
      OR ( (Verifica = 'OK') AND (Netto = StipLordo - Tasse) ) )
```

Basi di dati — 13 novembre 2017 — Prova parziale — Compito B

Possibili soluzioni

Tempo a disposizione: un'ora.

Cognome: _____ **Nome:** _____ **Matricola:** _____

Domanda 1 (25%) Considerare lo schema con le seguenti relazioni

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

con le seguenti cardinalità

- `studenti`: cardinalità $S = 1000$
- `corsi`: cardinalità $C = 200$
- `esami`: cardinalità $E = 10.000$

Indicare la cardinalità del risultato di ciascuna delle seguenti interrogazioni SQL, specificando l'intervallo nel quale essa può variare; indicare simboli e numeri.

	Min (simboli e valore)	Max (simboli e valore)

`SELECT matricola, codice`
`FROM studenti, corsi`
`WHERE eta > 20`
	0
	0
		$S \times C$
		200.000

`SELECT *`
`FROM studenti JOIN esami`
`  ON matricola = studente`
`WHERE voto > 27`
	0
	0
		E
		10.000

`SELECT matricola, codice`
`FROM studenti, esami, corsi`
`WHERE matricola = studente`
`  AND corso = codice`
	E
	10.000
		E
		10.000

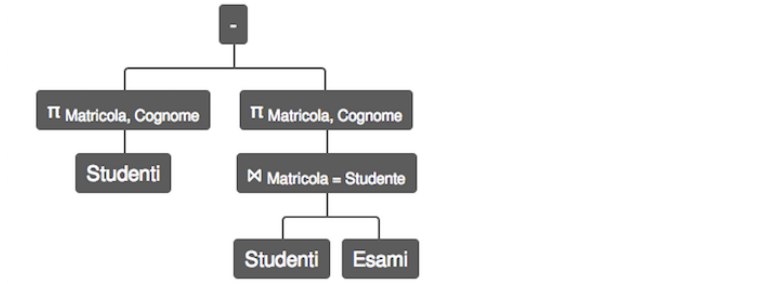
Basi di dati I — 13 novembre 2017 — Compito B

Domanda 2 (55%) Considerare nuovamente lo schema utilizzato nella domanda precedente

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

Formulare la seguente interrogazione in **algebra relazionale**

1. Mostrare matricola e cognome degli studenti per i quali non è stato registrato nessun esame



$\pi_{\text{Matricola, Cognome}}(\text{Studenti}) - \pi_{\text{Matricola, Cognome}}(\text{Studenti} \bowtie_{\text{Matricola = Studente}} \text{Esami})$

$\pi_{\text{Matricola, Cognome}}(\sigma_{\text{Studente = null}}(\text{Studenti} \bowtie_{\text{Matricola = Studente}} \text{Esami}))$

Più correttamente si dovrebbe scrivere 'is null' invece di '= null'

Formulare le seguenti interrogazioni in **SQL**

2. Per ciascun corso, mostrare codice, numero di esami registrati e media dei voti.

```
select corso as codice, count(*) as numeroEsami, avg(voto)::numeric(4,1) as media
from esami
group by corso
```

3. Per ciascuno studente, mostrare matricola, numero di CFU conseguiti e media dei voti.

```
select studente, sum(CFU) as CFU, avg(voto)::numeric(4,1) as media
from esami join corsi on corso = codice
group by studente
```

4. Per ciascuno studente, mostrare matricola e numero di CFU conseguiti, includendo anche gli studenti che non hanno superato esami (con valore 0 per il numero di CFU conseguiti).

```
select studente, sum(CFU) as CFU
from esami join corsi on corso = codice
group by studente
union
select matricola as studente, 0 as CFU
from studenti
where not exists (select * from esami where studente = matricola)

OPPURE

select matricola as studente, sum(CFU) as CFU
from studenti left join (esami join corsi on corso = codice) on matricola = studente
group by matricola
```

5. Mostrare codice e titolo del corso per il quale sono stati registrati più esami.

```
create view numeroEsami as
  select corso, count(*) as numeroEsami
  from esami
  group by corso;

select codice, titolo, numeroEsami
from corsi join numeroEsami on codice = corso
where numeroEsami = (select max(numeroEsami) from numeroEsami)
```

Domanda 3 (20%)
Considerare una relazione

PAGHE(Matricola,StipLordo,Ritenute,StipNetto,OK)

e definire su di essa

1. un vincolo che imponga che, se il valore di OK è "OK", allora StipNetto è uguale alla differenza fra StipLordo e Ritenute (si noti che non si vuole invece imporre nessuna condizione se il valore di OK è diverso da "OK").

```
CHECK ( (NOT (OK = 'OK') ) OR (StipNetto = StipLordo - Ritenute) )
```

2. un vincolo che imponga che il valore di OK è "OK" se e solo se StipNetto è uguale alla differenza fra StipLordo e Ritenute.

```
CHECK ( (NOT (OK = 'OK') AND NOT (StipNetto = StipLordo - Ritenute) )
       OR ( (OK = 'OK') AND (StipNetto = StipLordo - Ritenute) ) )
```

Basi di dati — 13 novembre 2017 — Prova parziale — Compito C

Possibili soluzioni

Tempo a disposizione: un'ora.

Cognome: _____ **Nome:** _____ **Matricola:** _____

Domanda 1 (25%) Considerare lo schema con le seguenti relazioni

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

con le seguenti cardinalità

- `studenti`: cardinalità $S = 1000$
- `corsi`: cardinalità $C = 200$
- `esami`: cardinalità $E = 10.000$

Indicare la cardinalità del risultato di ciascuna delle seguenti interrogazioni SQL, specificando l'intervallo nel quale essa può variare; indicare simboli e numeri.

	Min (simboli e valore)	Max (simboli e valore)
`SELECT matricola, codice` `FROM studenti, corsi`	$S \times C$ 200.000	$S \times C$ 200.000

`SELECT *`
`FROM studenti JOIN esami`
`  ON matricola = studente`
	E
	10.000
		E
		10.000

`SELECT matricola, codice`
`FROM studenti, esami, corsi`
`WHERE matricola = studente`
`  AND corso = codice`
`  AND voto > 24`
	0
	0
		E
		10.000

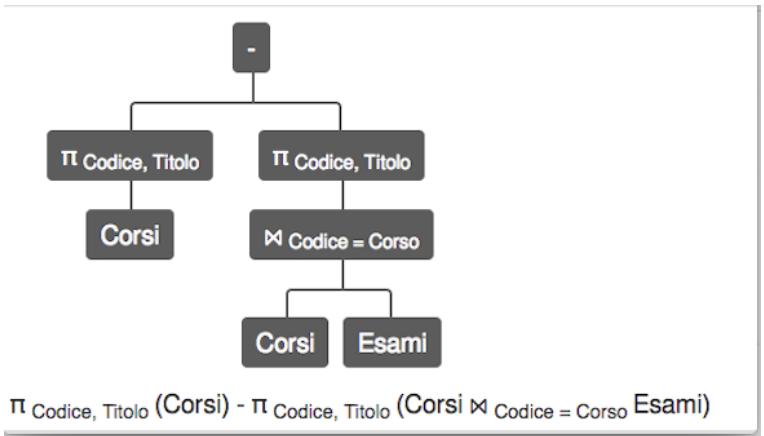
Basi di dati I — 13 novembre 2017 — Compito C

Domanda 2 (55%) Considerare nuovamente lo schema utilizzato nella domanda precedente

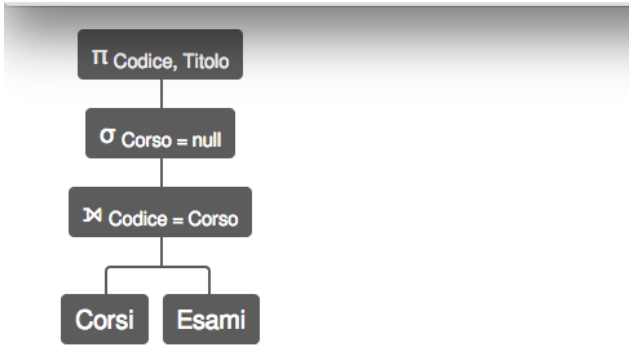
```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

Formulare la seguente interrogazione in **algebra relazionale**

1. Mostrare codice e titolo dei corsi per i quali non è stato registrato nessun esame.



$\pi_{\text{Codice, Titolo}}(\text{Corsi}) - \pi_{\text{Codice, Titolo}}(\text{Corsi} \bowtie_{\text{Codice = Corso}} \text{Esami})$



$\pi_{\text{Codice, Titolo}}(\sigma_{\text{Corso = null}}(\text{Corsi} \bowtie_{\text{Codice = Corso}} \text{Esami}))$

Più correttamente si dovrebbe scrivere 'is null' invece di '= null'

Formulare le seguenti interrogazioni in **SQL**

2. Per ciascuno studente, mostrare matricola, numero di CFU conseguiti e media dei voti.

```
select studente, sum(CFU) as CFU, avg(voto)::numeric(4,1) as media
from esami join corsi on corso = codice
group by studente
```

3. Per ciascun corso, mostrare codice, numero di esami registrati e media dei voti.

```
select corso as codice, count(*) as numeroEsami, avg(voto)::numeric(4,1) as media
from esami
group by corso
```

4. Per ciascun corso, mostrare codice e numero di esami registrati, includendo anche i corsi per i quali non è stato registrato alcun esame (con il valore 0 per il numero di esami).

```
select corso, count(*) as numeroEsami
from esami
group by corso
union
select codice as corso, 0 as numeroEsami
from corsi
where not exists (select * from esami where corso = codice)

OPPURE

select codice, count(corso) as numeroEsami
from corsi left join esami ON codice=corso
group by codice
```

5. Mostrare codice e titolo del corso per il quale sono stati registrati più esami.

```
create view numeroEsami as
  select corso, count(*) as numeroEsami
  from esami
  group by corso;

select codice, titolo, numeroEsami
from corsi join numeroEsami on codice = corso
where numeroEsami = (select max(numeroEsami) from numeroEsami)
```

Domanda 3 (20%)
Considerare una relazione

RETRIBUZIONI(Matricola,Lordo,Imposte,StipNetto,Verifica)

e definire su di essa

1. un vincolo che imponga che, se il valore di Verifica è "OK", allora StipNetto è uguale alla differenza fra Lordo e Imposte (si noti che non si vuole invece imporre nessuna condizione se il valore di Verifica è diverso da "OK").

```
CHECK ( (NOT (Verifica = 'OK') ) OR (StipNetto = Lordo - Imposte) )
```

2. un vincolo che imponga che il valore di Verifica è "OK" se e solo se StipNetto è uguale alla differenza fra Lordo e Imposte.

```
CHECK ( (NOT (Verifica = 'OK') AND NOT (StipNetto = Lordo - Imposte) )
      OR ( (Verifica = 'OK') AND (StipNetto = Lordo - Imposte) ) )
```

Basi di dati — 13 novembre 2017 — Prova parziale — Compito D

Possibili soluzioni

Tempo a disposizione: un'ora.

Cognome: _____ **Nome:** _____ **Matricola:** _____

Domanda 1 (25%) Considerare lo schema con le seguenti relazioni

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

con le seguenti cardinalità

- `studenti`: cardinalità $S = 1000$
- `corsi`: cardinalità $C = 200$
- `esami`: cardinalità $E = 10.000$

Indicare la cardinalità del risultato di ciascuna delle seguenti interrogazioni SQL, specificando l'intervallo nel quale essa può variare; indicare simboli e numeri.

	Min (simboli e valore)	Max (simboli e valore)

`SELECT matricola, codice`
`FROM studenti, corsi`
`WHERE eta > 20`
	0
	
	0
		$S \times C$
		
		200.000

`SELECT *`
`FROM studenti JOIN esami`
`  ON matricola = studente`
	E
	
	10.000
		E
		
		10.000

`SELECT matricola, codice`
`FROM studenti, esami, corsi`
`WHERE matricola = studente`
`  AND corso = codice`
	E
	
	10.000
		E
		
		10.000

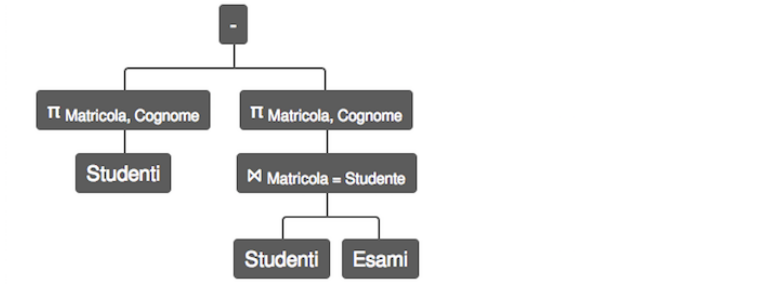
Basi di dati I — 13 novembre 2017 — Compito D

Domanda 2 (55%) Considerare nuovamente lo schema utilizzato nella domanda precedente

```
create table studenti (matricola numeric not null primary key,
                       cognome char(20) not null,
                       nome char(20) not null,
                       eta numeric);
create table corsi (codice numeric not null primary key,
                    titolo char(20) not null,
                    CFU numeric not null);
create table esami (corso numeric not null references corsi(codice),
                    studente numeric not null references studenti(matricola),
                    data date not null,
                    voto numeric not null,
                    primary key (corso, studente));
```

Formulare la seguente interrogazione in **algebra relazionale**

1. Mostrare matricola e cognome degli studenti per i quali non è stato registrato nessun esame



$\pi_{\text{Matricola, Cognome}}(\text{Studenti}) - \pi_{\text{Matricola, Cognome}}(\text{Studenti} \bowtie_{\text{Matricola = Studente}} \text{Esami})$

$\pi_{\text{Matricola, Cognome}}(\sigma_{\text{Studente = null}}(\text{Studenti} \bowtie_{\text{Matricola = Studente}} \text{Esami}))$

Più correttamente si dovrebbe scrivere 'is null' invece di '= null'

Formulare le seguenti interrogazioni in **SQL**

2. Per ciascun corso, mostrare codice, numero di esami registrati e media dei voti.

```
select corso as codice, count(*) as numeroEsami, avg(voto)::numeric(4,1) as media
from esami
group by corso
```

3. Per ciascuno studente, mostrare matricola, numero di CFU conseguiti e media dei voti.

```
select studente, sum(CFU) as CFU, avg(voto)::numeric(4,1) as media
from esami join corsi on corso = codice
group by studente
```

4. Per ciascuno studente, mostrare matricola e numero di CFU conseguiti, includendo anche gli studenti che non hanno superato esami (con valore 0 per il numero di CFU conseguiti).

```
select studente, sum(CFU) as CFU
from esami join corsi on corso = codice
group by studente
union
select matricola as studente, 0 as CFU
from studenti
where not exists (select * from esami where studente = matricola)

OPPURE

select matricola as studente, sum(CFU) as CFU
from studenti left join (esami join corsi on corso = codice) on matricola = studente
group by matricola
```

5. Mostrare matricola, cognome e nome dello studente che ha conseguito il maggior numero di credti.

```
create view totaleCFU as
  select studente, sum(CFU) as CFU
  from esami join corsi on corso = codice
  group by studente;

select matricola, cognome, nome, CFU
from studenti join totaleCFU on matricola = studente
where CFU = (select max(CFU) from totaleCFU)
```

Domanda 3 (20%)
Considerare una relazione

SALARI(Matricola,StipLordo,Trattenute,Netto,OK)

e definire su di essa

1. un vincolo che imponga che, se il valore di OK è "OK", allora Netto è uguale alla differenza fra StipLordo e Trattenute (si noti che non si vuole invece imporre nessuna condizione se il valore di OK è diverso da "OK").

```
CHECK ( (NOT (OK = 'OK') ) OR (Netto = StipLordo - Trattenute) )
```

2. un vincolo che imponga che il valore di OK è "OK" se e solo se Netto è uguale alla differenza fra StipLordo e Trattenute.

```
CHECK ( (NOT (OK = 'OK') AND NOT (Netto = StipLordo - Trattenute) )
       OR ( (OK = 'OK') AND (Netto = StipLordo - Trattenute) ) )
```