Basi di dati II

Prova parziale — 10 aprile 2017 — Compito A

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 2$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $N = 1.000.000$ ennuple di $r = 50$ Byte ciascuna, con $v = 10.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $N/v = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.

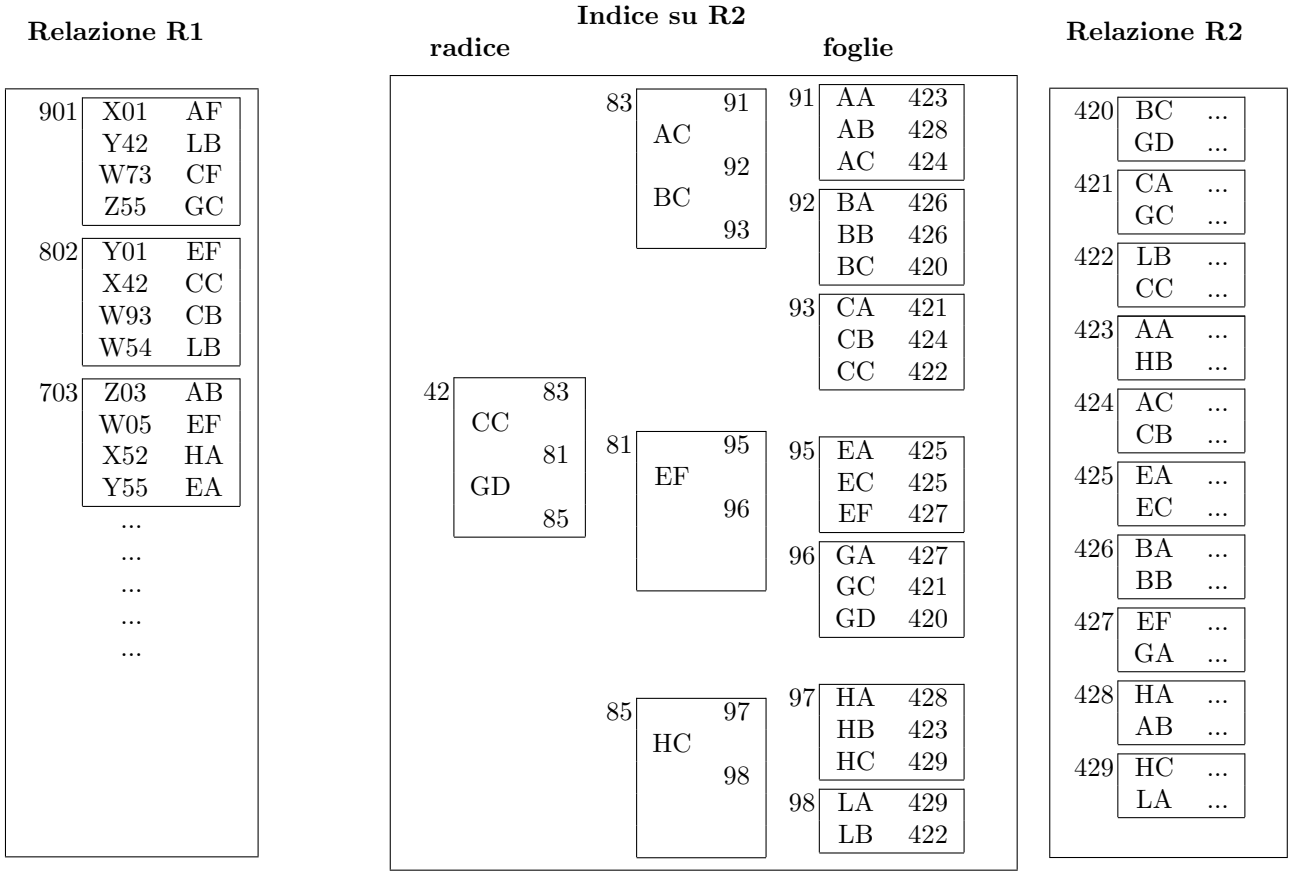
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

5. inserimento di $N/10 = 100.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	2	1	0
blocco	93	33	47	35
istante load	1	7	9	3
istante unpin				8
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)									
17	pin(93)									
20	pin(49)									
22	pin(93)									

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito B

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 3$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $L = 2.000.000$ ennuple di $r = 50$ Byte ciascuna, con $c = 20.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $L/c = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.

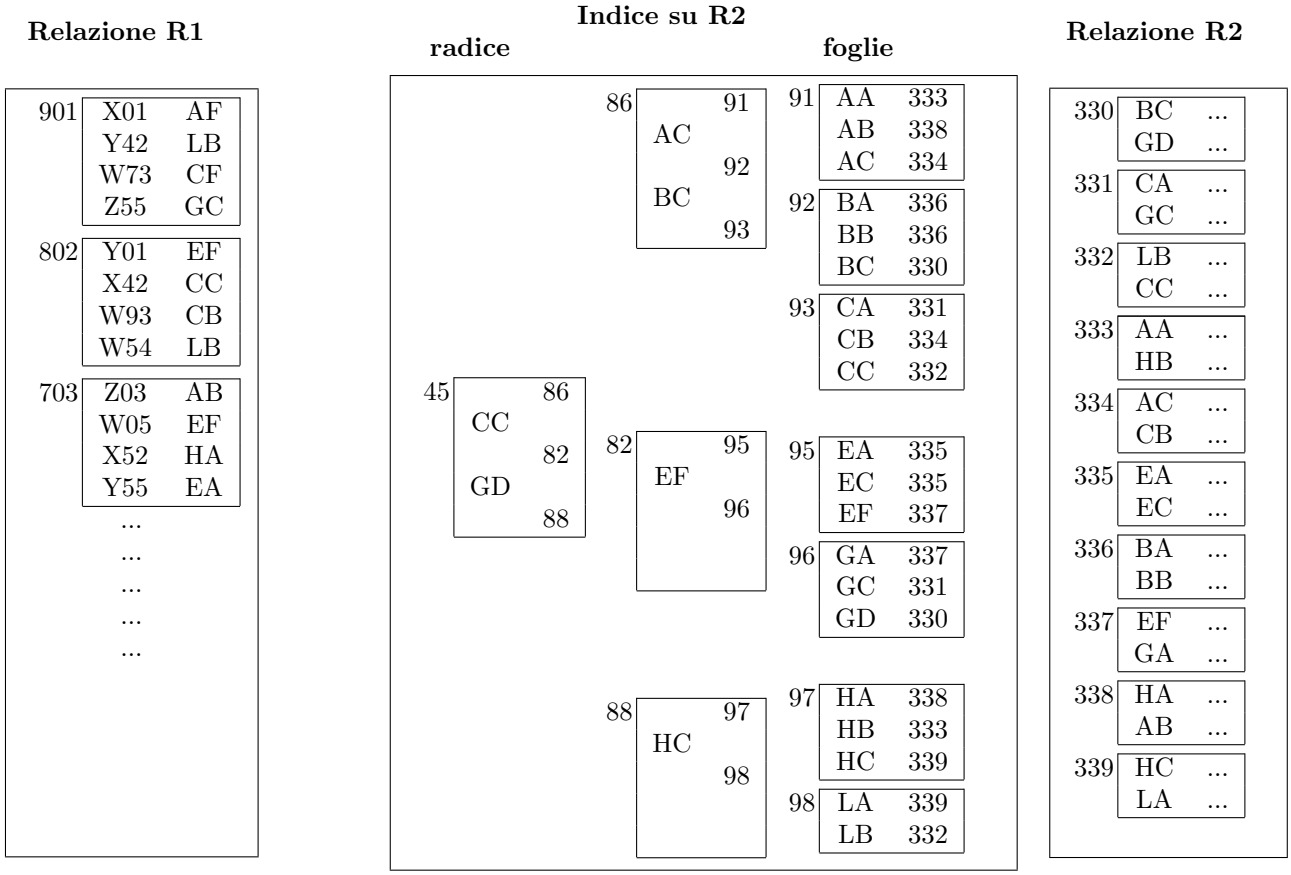
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

5. inserimento di $L/10 = 200.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	0	1	2
blocco	93	35	47	33
istante load	1	3	9	7
istante unpin		8		
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)									
17	pin(93)									
20	pin(49)									
22	pin(93)									

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito C

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 4$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 10.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $L = 1.000.000$ ennuple di $r = 50$ Byte ciascuna, con $v = 10.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $L/v = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.

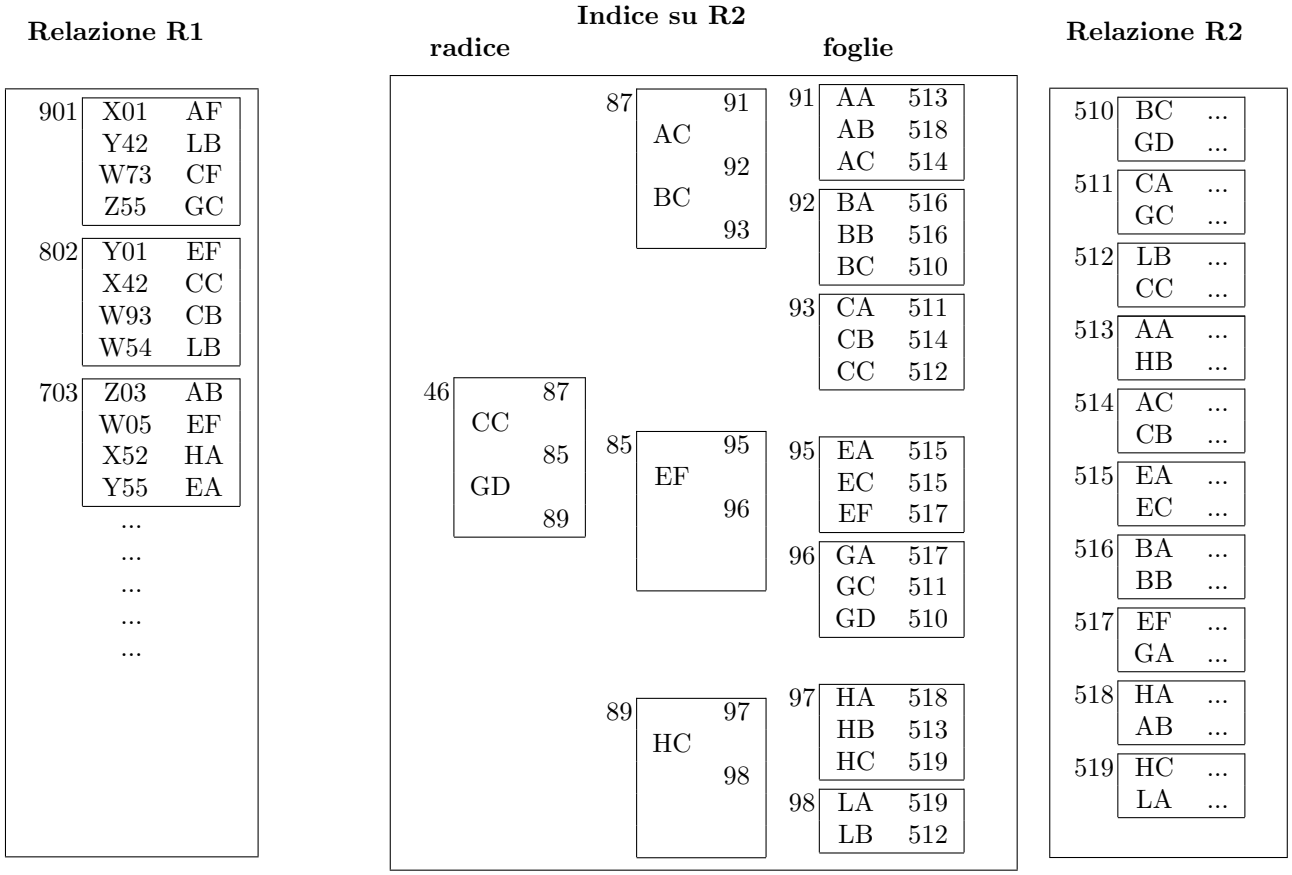
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

5. inserimento di $L/10 = 100.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	2	1	0
blocco	93	33	47	35
istante load	1	7	9	3
istante unpin				8
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)									
17	pin(93)									
20	pin(49)									
22	pin(93)									

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito D

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 3$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $N = 2.000.000$ ennuple di $r = 50$ Byte ciascuna, con $c = 20.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $N/c = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.

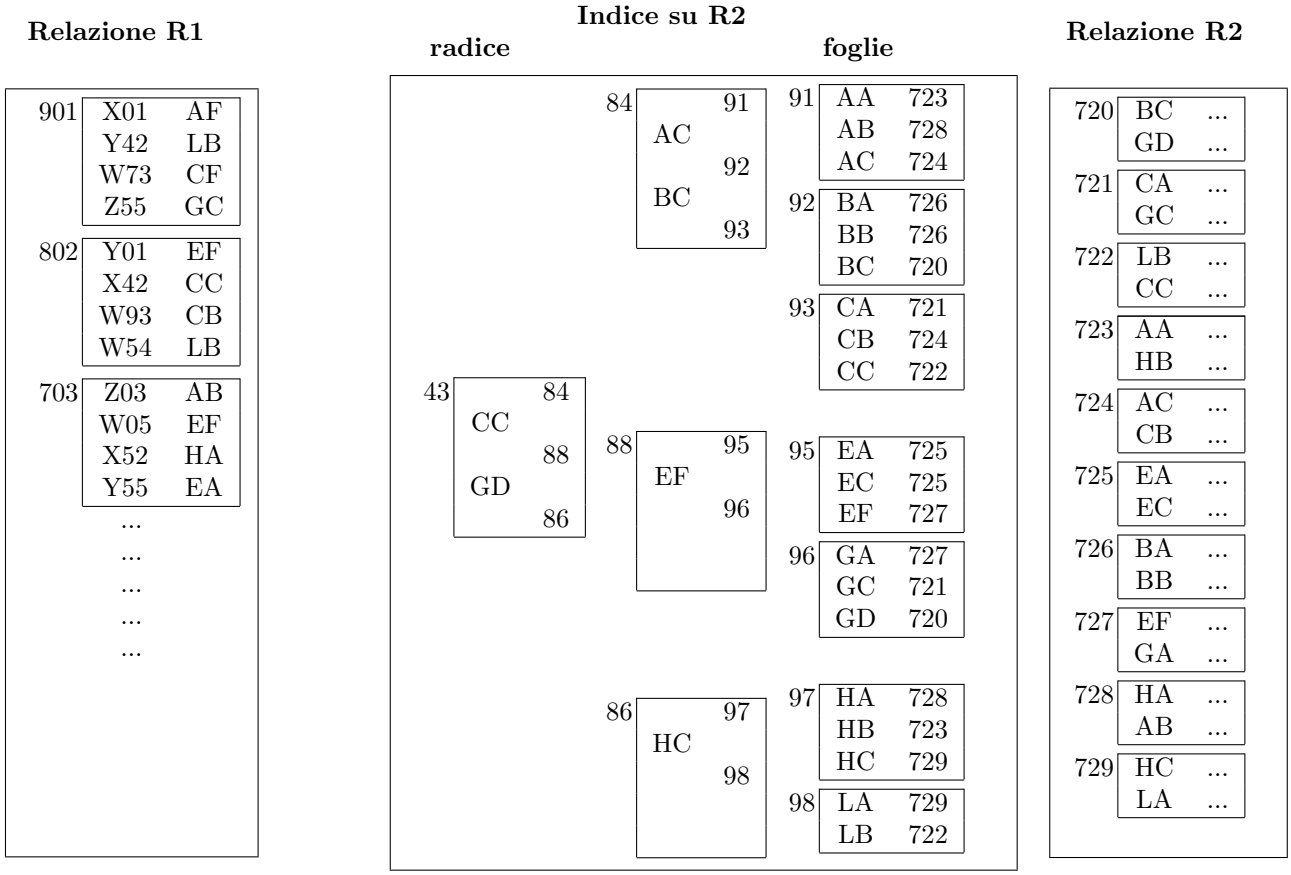
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

5. inserimento di $N/10 = 200.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	0	1	2
blocco	93	35	47	33
istante load	1	3	9	7
istante unpin		8		
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)									
17	pin(93)									
20	pin(49)									
22	pin(93)									

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito A

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 2$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

La somma del tempo medio di seek, del tempo medio di latenza e del tempo minimo di lettura di un blocco

$$t_{tot} = t_S + t_L + t_B = 5 + 2 + 0,01 \text{ msec } = \text{ ca } 7 \text{ msec}$$

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

Il tempo medio necessario per leggere un blocco (il primo) più il tempo minimo di lettura per ciascuno degli altri

$$t_{tot} + (F - 1) \times t_B = \text{ ca } 9 \text{ msec}$$

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

Si può immaginare all'inizio nessuno nodo dell'indice si trovi nel buffer. Quindi, per il primo record, $p = 4$ accessi per l'indice e uno per il record del file, per gli altri la radice si trova nel buffer, quindi $p - 1$ più uno:

$$((p + 1) + 3 \times ((p - 1) + 1)) \times t_{tot} = \text{ ca } 120 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Si può immaginare che la radice e altri due livelli dell'indice restino nel buffer, dopo il primo caricamento, quindi, per ogni record, due accessi all'indice e uno al file, in posizioni non prevedibili (qualche accesso in più per il caricamento iniziale e qualcuno in meno per blocchi acceduti piè volte):

$$(p - 3 + 1) \times m \times t_{tot} = \text{ ca } 280 \text{ sec}$$

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $N = 1.000.000$ ennuple di $r = 50$ Byte ciascuna, con $v = 10.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $N/v = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

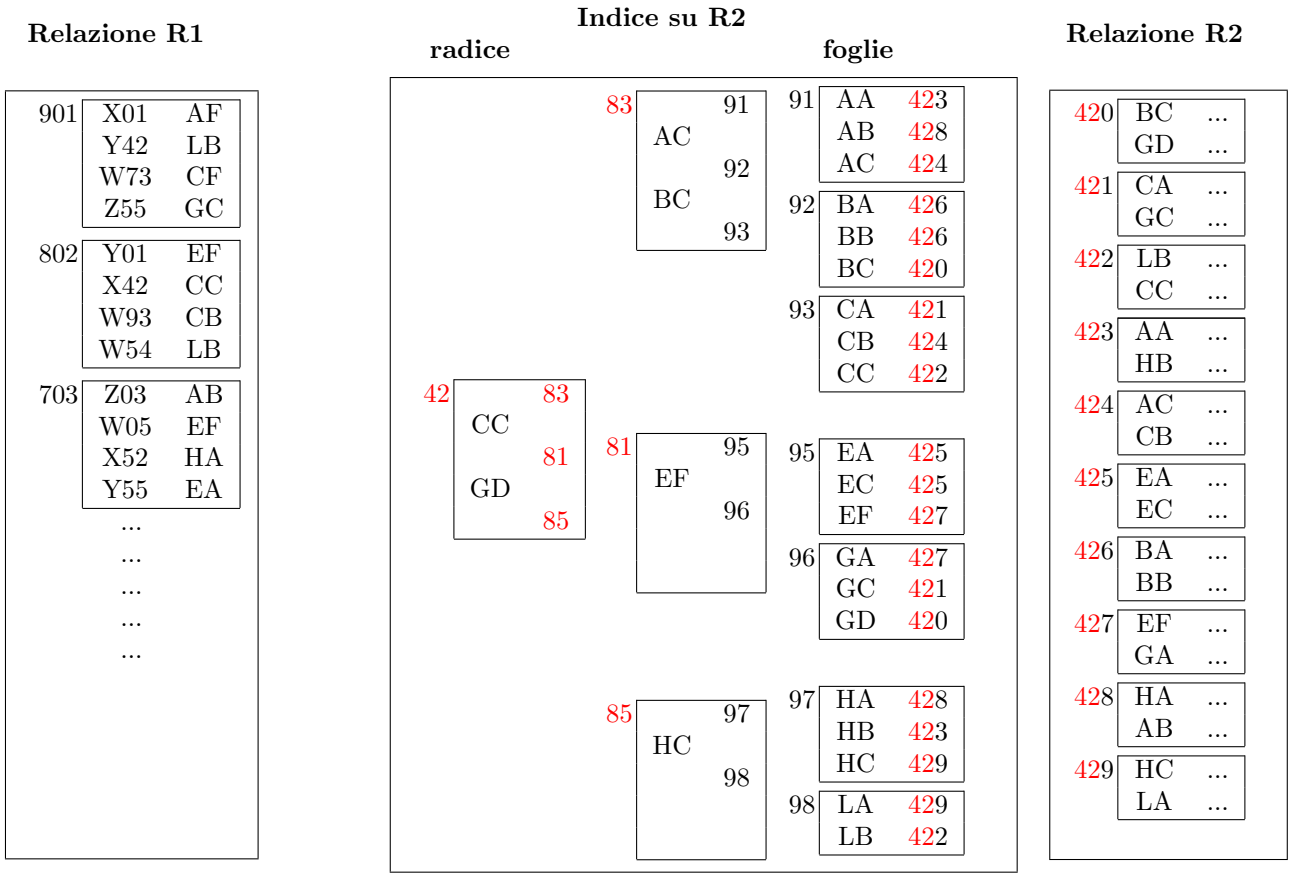
I record con lo stesso valore di `C` sono $N/v = 100$ e si trovano in uno stesso blocco o in blocchi consecutivi. Visto che il fattore di blocco è $B/r = 80$, i blocchi che li contengono sono $(N/v)/(B/r)$, arrotondato per eccesso ed eventualmente aumentato di uno, quindi 2 o 3, e quindi serviranno 5 o 6 accessi ($p = 3$ per l'indice più 2 o 3 per i blocchi). L'indice si visita una volta sola.

5. inserimento di $N/10 = 100.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

I 10 nuovi record con `C` $= 100$ si troveranno presumibilmente in 10 blocchi diversi e quindi il numero di accessi sarà pari al valore indicato nella risposta al punto precedente più 10 e quindi circa 15. L'indice, come nel caso precedente, si visita una volta sola

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.
901, 42, 83, 92,
42, 85, 98, 422, — produce la ennupla (Y42, LB, ...)
42, 81, 95,
42, 81, 96, 421, — produce la ennupla (Z55, GC, ...)
802, 42, 81, 95, 427 — produce la ennupla (Y01, EF, ...)

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.
901, 42, 83, 92,
85, 98, 422,
81, 95,
96, 421,
802, 95, 427
In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.
901, 802, 42, 81, 95, 427

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.
901, 42, 83, 92,
85, 98, 422,
42, 81, 95,
96, 421,
802, 42, 81, 95, 427

Basi di dati II — 10 aprile 2017 — Compito A

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	2	1	0
blocco	93	33	47	35
istante load	1	7	9	3
istante unpin				8
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

Soluzione corretta solo per i compiti A e C

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)	0	sì	sì	3	sì	no	3	sì	no
17	pin(93)	0	sì	no	0	no	no	0	no	no
20	pin(49)	0	sì	sì	3	no	no	3	no	no
22	pin(93)	0	sì	no	0	no	no	0	no	no

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito B

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 3$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

La somma del tempo medio di seek, del tempo medio di latenza e del tempo minimo di lettura di un blocco

$$t_{tot} = t_S + t_L + t_B = 4 + 3 + 0,01 \text{ msec } = \text{ ca } 7 \text{ msec}$$

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

Il tempo medio necessario per leggere un blocco (il primo) più il tempo minimo di lettura per ciascuno degli altri

$$t_{tot} + (F - 1) \times t_B = \text{ ca } 9 \text{ msec}$$

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

Si può immaginare all'inizio nessuno nodo dell'indice si trovi nel buffer. Quindi, per il primo record, $p = 4$ accessi per l'indice e uno per il record del file, per gli altri la radice si trova nel buffer, quindi $p - 1$ più uno:

$$((p + 1) + 3 \times ((p - 1) + 1)) \times t_{tot} = \text{ ca } 120 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Si può immaginare che la radice e altri due livelli dell'indice restino nel buffer, dopo il primo caricamento, quindi, per ogni record, due accessi all'indice e uno al file, in posizioni non prevedibili (qualche accesso in più per il caricamento iniziale e qualcuno in meno per blocchi acceduti piè volte):

$$(p - 3 + 1) \times m \times t_{tot} = \text{ ca } 280 \text{ sec}$$

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $L = 2.000.000$ ennuple di $r = 50$ Byte ciascuna, con $c = 20.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $L/c = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

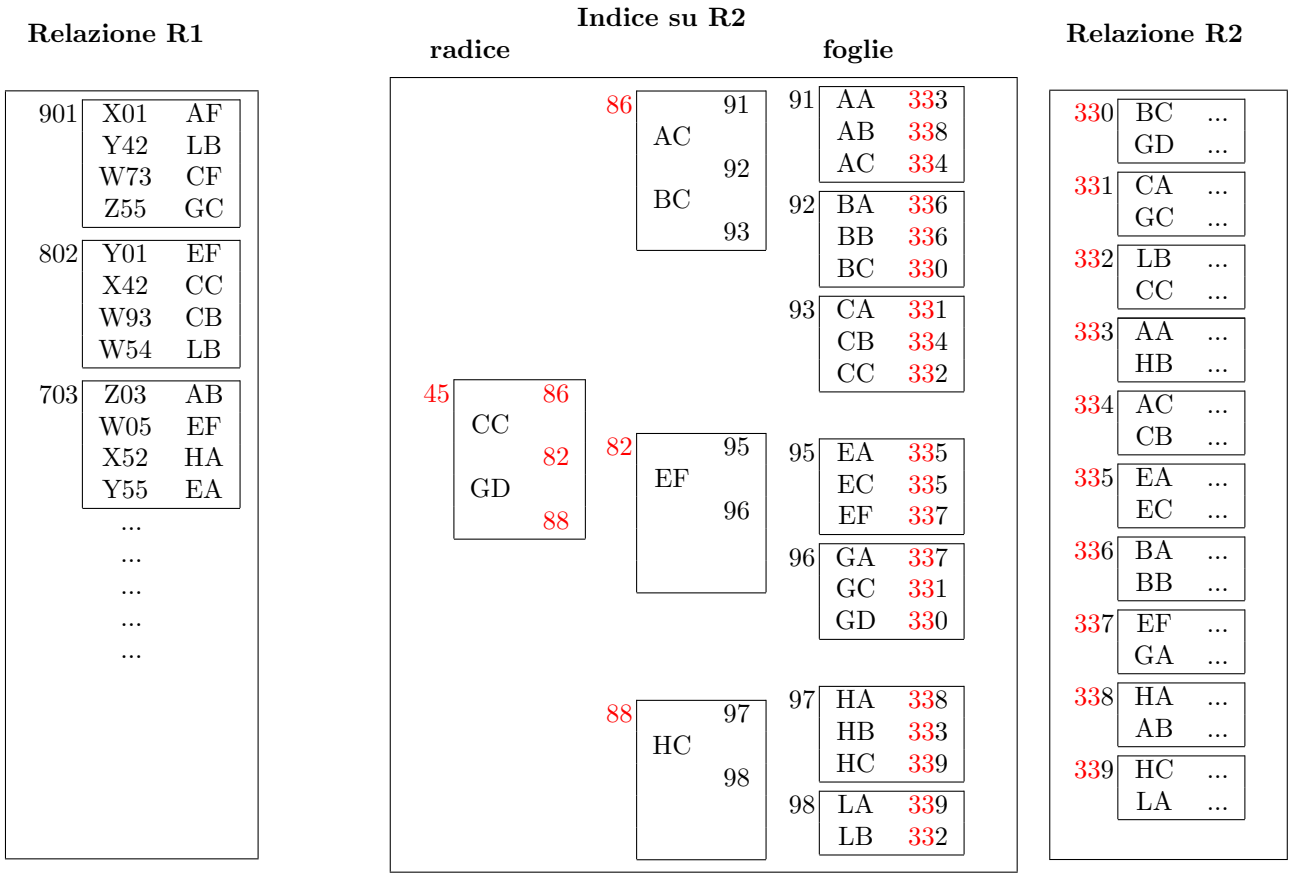
I record con lo stesso valore di `C` sono $L/c = 100$ e si trovano in uno stesso blocco o in blocchi consecutivi. Visto che il fattore di blocco è $B/r = 80$, i blocchi che li contengono sono $(L/c)/(B/r)$, arrotondato per eccesso ed eventualmente aumentato di uno, quindi 2 o 3, e quindi serviranno 5 o 6 accessi ($p = 3$ per l'indice più 2 o 3 per i blocchi). L'indice si visita una volta sola.

5. inserimento di $L/10 = 200.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

I 10 nuovi record con `C` $= 100$ si troveranno presumibilmente in 10 blocchi diversi e quindi il numero di accessi sarà pari al valore indicato nella risposta al punto precedente più 10 e quindi circa 15. L'indice, come nel caso precedente, si visita una volta sola

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

901, 45, 86, 92,
45, 88, 98, 332, — produce la ennupla (Y42, LB, ...)
45, 82, 95,
45, 82, 96, 331, — produce la ennupla (Z55, GC, ...)
802, 45, 82, 95, 337 — produce la ennupla (Y01, EF, ...)

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

901, 45, 86, 92,
88, 98, 332,
82, 95,
96, 331,
802, 95, 337

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

901, 802, 45, 82, 95, 337

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

901, 45, 86, 92,
88, 98, 332,
45, 82, 95,
96, 331,
802, 45, 82, 95, 337

Basi di dati II — 10 aprile 2017 — Compito B

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	0	1	2
blocco	93	35	47	33
istante load	1	3	9	7
istante unpin		8		
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

Soluzione corretta solo per i compiti A e C

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)	0	sì	sì	3	sì	no	3	sì	no
17	pin(93)	0	sì	no	0	no	no	0	no	no
20	pin(49)	0	sì	sì	3	no	no	3	no	no
22	pin(93)	0	sì	no	0	no	no	0	no	no

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito C

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 4$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 4$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

La somma del tempo medio di seek, del tempo medio di latenza e del tempo minimo di lettura di un blocco

$$t_{tot} = t_S + t_L + t_B = 4 + 4 + 0,01 \text{ msec } = \text{ ca } 8 \text{ msec}$$

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

Il tempo medio necessario per leggere un blocco (il primo) più il tempo minimo di lettura per ciascuno degli altri

$$t_{tot} + (F - 1) \times t_B = \text{ ca } 10 \text{ msec}$$

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

Si può immaginare all'inizio nessuno nodo dell'indice si trovi nel buffer. Quindi, per il primo record, $p = 4$ accessi per l'indice e uno per il record del file, per gli altri la radice si trova nel buffer, quindi $p - 1$ più uno:

$$((p + 1) + 3 \times ((p - 1) + 1)) \times t_{tot} = \text{ ca } 135 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 10.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Si può immaginare che la radice e altri due livelli dell'indice restino nel buffer, dopo il primo caricamento, quindi, per ogni record, due accessi all'indice e uno al file, in posizioni non prevedibili (qualche accesso in più per il caricamento iniziale e qualcuno in meno per blocchi acceduti piè volte):

$$(p - 3 + 1) \times m \times t_{tot} = \text{ ca } 160 \text{ sec}$$

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $L = 1.000.000$ ennuple di $r = 50$ Byte ciascuna, con $v = 10.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $L/v = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

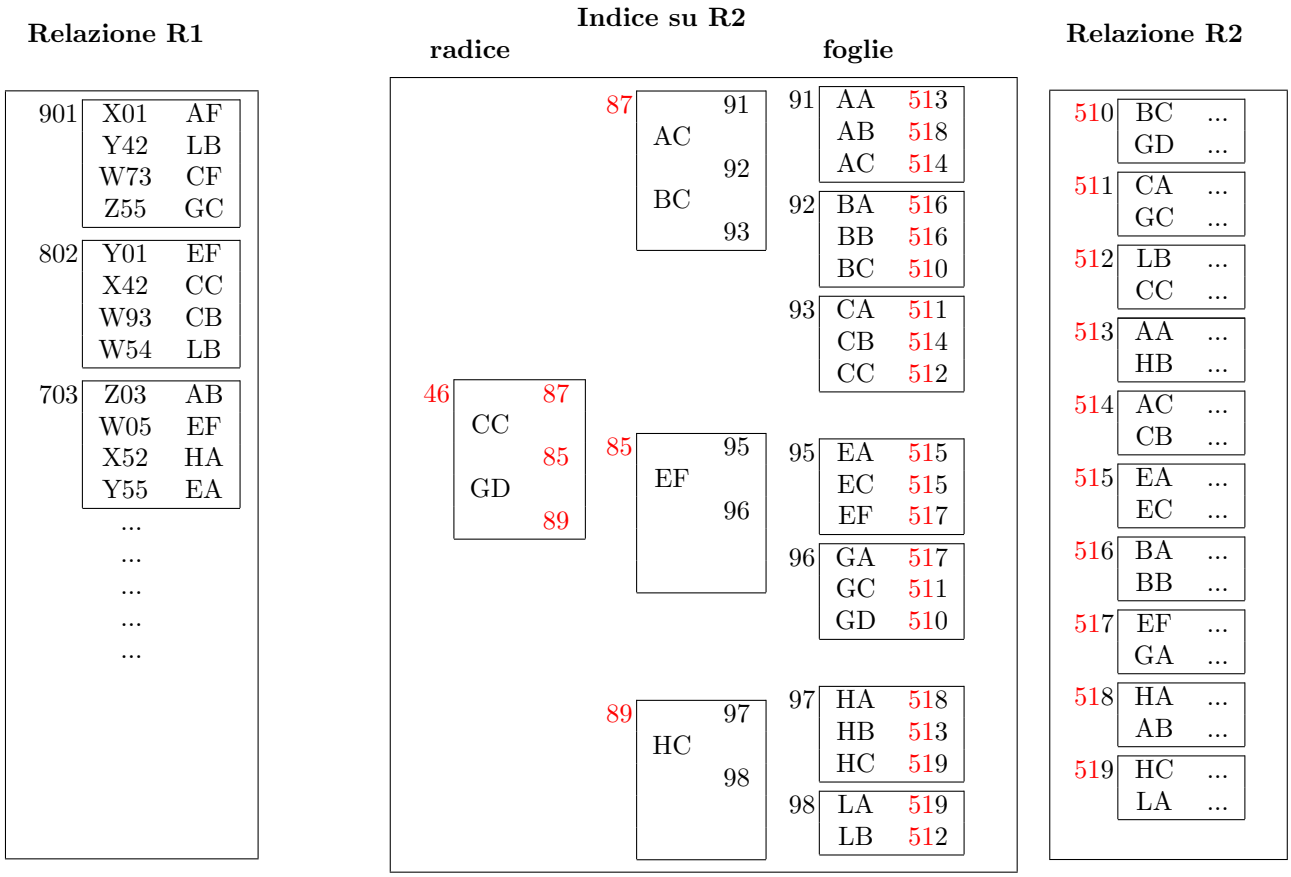
I record con lo stesso valore di `C` sono $L/v = 100$ e si trovano in uno stesso blocco o in blocchi consecutivi. Visto che il fattore di blocco è $B/r = 80$, i blocchi che li contengono sono $(L/v)/(B/r)$, arrotondato per eccesso ed eventualmente aumentato di uno, quindi 2 o 3, e quindi serviranno 5 o 6 accessi ($p = 3$ per l'indice più 2 o 3 per i blocchi). L'indice si visita una volta sola.

5. inserimento di $L/10 = 100.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

I 10 nuovi record con `C` $= 100$ si troveranno presumibilmente in 10 blocchi diversi e quindi il numero di accessi sarà pari al valore indicato nella risposta al punto precedente più 10 e quindi circa 15. L'indice, come nel caso precedente, si visita una volta sola

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

901, 46, 87, 92,
46, 89, 98, 512, — produce la ennupla (Y42, LB, ...)
46, 85, 95,
46, 85, 96, 511, — produce la ennupla (Z55, GC, ...)
802, 46, 85, 95, 517 — produce la ennupla (Y01, EF, ...)

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

901, 46, 87, 92,
89, 98, 512,
85, 95,
96, 511,
802, 95, 517

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

901, 802, 46, 85, 95, 517

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

901, 46, 87, 92,
89, 98, 512,
46, 85, 95,
96, 511,
802, 46, 85, 95, 517

Basi di dati II — 10 aprile 2017 — Compito C

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	2	1	0
blocco	93	33	47	35
istante load	1	7	9	3
istante unpin				8
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

Soluzione corretta solo per i compiti A e C

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)	0	sì	sì	3	sì	no	3	sì	no
17	pin(93)	0	sì	no	0	no	no	0	no	no
20	pin(49)	0	sì	sì	3	no	no	3	no	no
22	pin(93)	0	sì	no	0	no	no	0	no	no

Basi di dati II

Prova parziale — 10 aprile 2017 — Compito D

Cenni sulle soluzioni

Tempo a disposizione: un'ora.

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

Domanda 1 (25%)
Considerare un sistema con dischi con le seguenti caratteristiche

- tempo medio di posizionamento della testina (tempo di seek) $t_S = 5$ msec
- tempo medio di latenza (attesa dovuta alla rotazione) $t_L = 3$ msec
- tempo minimo di lettura di un blocco $t_B = 10$ μsec

Rispondere alle seguenti domande mostrando formula e valore numerico

1. Qual è il tempo medio necessario per leggere un blocco del quale sia dato l'indirizzo fisico?

La somma del tempo medio di seek, del tempo medio di latenza e del tempo minimo di lettura di un blocco

$$t_{tot} = t_S + t_L + t_B = 5 + 3 + 0,01 \text{ msec } = \text{ ca } 8 \text{ msec}$$

2. Qual è il tempo medio necessario per la scansione sequenziale di un file costituito da $F = 200$ blocchi contigui, non letti di recente?

Il tempo medio necessario per leggere un blocco (il primo) più il tempo minimo di lettura per ciascuno degli altri

$$t_{tot} + (F - 1) \times t_B = \text{ ca } 10 \text{ msec}$$

3. Qual è il tempo che si può ipotizzare necessario per eseguire quattro accessi diretti a record di un file attraverso un indice che abbia profondità $p = 4$ e fan-out (fattore di blocco dell'indice) $f_I = 50$, non usato di recente?

Si può immaginare all'inizio nessuno nodo dell'indice si trovi nel buffer. Quindi, per il primo record, $p = 4$ accessi per l'indice e uno per il record del file, per gli altri la radice si trova nel buffer, quindi $p - 1$ più uno:

$$((p + 1) + 3 \times ((p - 1) + 1)) \times t_{tot} = \text{ ca } 135 \text{ msec}$$

4. Qual è il tempo che si può ipotizzare necessario per eseguire $m = 20.000$ accessi diretti in tempi ravvicinati a record di un file (molto grande) attraverso un indice che abbia profondità $p = 4$, fan-out $f_I = 50$, con disponibilità di circa $P = 4000$ pagine di buffer?

Si può immaginare che la radice e altri due livelli dell'indice restino nel buffer, dopo il primo caricamento, quindi, per ogni record, due accessi all'indice e uno al file, in posizioni non prevedibili (qualche accesso in più per il caricamento iniziale e qualcuno in meno per blocchi acceduti piè volte):

$$(p - 3 + 1) \times m \times t_{tot} = \text{ ca } 320 \text{ sec}$$

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 2 (25%)
In Postgres (e, con sintassi diverse, negli altri sistemi) è possibile ordinare fisicamente una relazione con il comando `CLUSTER`. L'ordinamento viene specificato sulla base di un indice già definito sulla stessa relazione. L'ordinamento non viene però mantenuto (se non eseguendo nuovamente il comando `CLUSTER`). Dal manuale:

> CLUSTER -- cluster a table according to an index
>
> **Synopsis**
>
> ```
> CLUSTER [VERBOSE] table_name [ USING index_name ]
> CLUSTER [VERBOSE]
> ```
>
> **Description**
>
> `CLUSTER` instructs PostgreSQL to cluster the table specified by ***table_name*** based on the index specified by ***index_name***. The index must already have been defined on ***table_name***.
>
> When a table is clustered, it is physically reordered based on the index information. Clustering is a one-time operation: when the table is subsequently updated, the changes are not clustered. That is, no attempt is made to store new or updated rows according to their index order. (If one wishes, one can periodically recluster by issuing the command again. Also, setting the table's `fillfactor` storage parameter to less than 100% can aid in preserving cluster ordering during updates, since updated rows are kept on the same page if enough space is available there.)

Sia data una relazione $R(\underline{A}, B, C)$ contenente circa $N = 2.000.000$ ennuple di $r = 50$ Byte ciascuna, con $c = 20.000$ valori diversi per l'attributo C, uniformemente distribuiti (quindi si può supporre che per ogni valore di C ci siano $N/c = 100$ ennuple con tale valore). Supporre che i blocchi abbiano dimensione $B = 4$KB approssimabile come 4.000.
Considerare le operazioni sotto elencate e indicare, nei riquadri, i costi delle `SELECT`:

1. `CREATE INDEX RCIX ON R (C)` (creazione di un indice su C; supporre che abbia profondità $p = 3$)
2. `CLUSTER R USING RCIX` (ordinamento di R sulla base dell'indice e quindi sull'attributo C)
3. Altre operazioni che non utilizzano l'indice `RCIX` e non modificano `R`
4. `SELECT * FROM R WHERE C = 100`

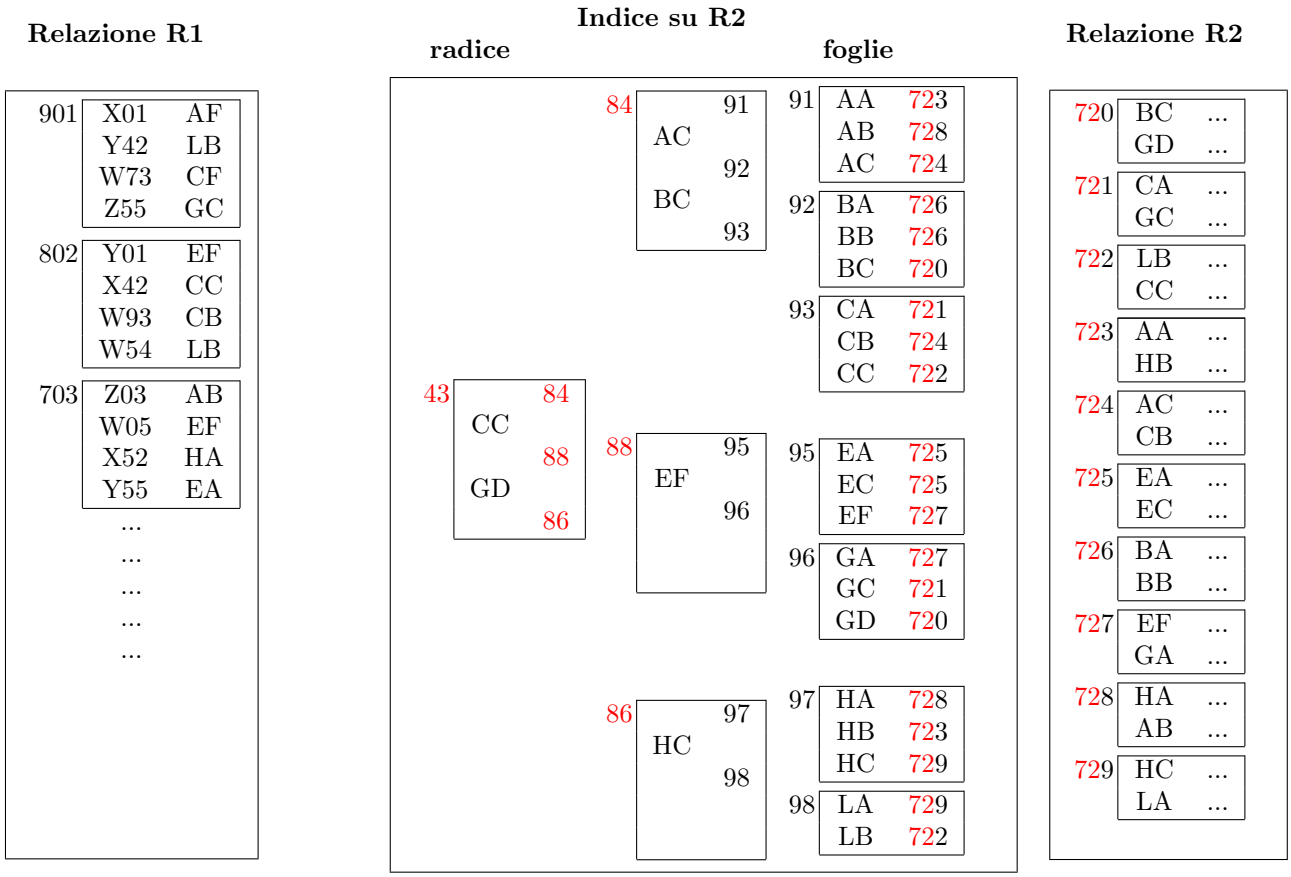
I record con lo stesso valore di `C` sono $N/c = 100$ e si trovano in uno stesso blocco o in blocchi consecutivi. Visto che il fattore di blocco è $B/r = 80$, i blocchi che li contengono sono $(N/c)/(B/r)$, arrotondato per eccesso ed eventualmente aumentato di uno, quindi 2 o 3, e quindi serviranno 5 o 6 accessi ($p = 3$ per l'indice più 2 o 3 per i blocchi). L'indice si visita una volta sola.

5. inserimento di $N/10 = 200.000$ ennuple, in ordine casuale, con valori di C pure uniformemente distribuiti (quindi si può supporre che vengano inserite 10 ennuple per ogni valore
6. `SELECT * FROM R WHERE C = 100`

I 10 nuovi record con `C` $= 100$ si troveranno presumibilmente in 10 blocchi diversi e quindi il numero di accessi sarà pari al valore indicato nella risposta al punto precedente più 10 e quindi circa 15. L'indice, come nel caso precedente, si visita una volta sola

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 3 (25%)
Considerare le relazioni R1 ed R2 e l'indice I2 su R2 schematizzati sotto. I riquadri interni indicano i blocchi e il numero a fianco a ciascun riquadro indica l'indirizzo del blocco. Nell'indice, i valori numerici sono riferimenti ai blocchi (blocchi dell'indice, per la radice e il livello intermedio, e blocchi di R2 per le foglie).



Supponendo di disporre di un buffer di **sei** pagine, considerare l'esecuzione del join di R1 ed R2, sulla base dei valori del secondo attributo di R1 e del primo di R2, con un **nested loop con accesso diretto** tramite l'indice di R2.

Indicare gli indirizzi dei blocchi su cui si eseguono operazioni di pin (o fix) per produrre le prime tre ennuple del risultato.

901, 43, 84, 92,
43, 86, 98, 722, — produce la ennupla (Y42, LB, ...)
43, 88, 95,
43, 88, 96, 721, — produce la ennupla (Z55, GC, ...)
802, 43, 88, 95, 727 — produce la ennupla (Y01, EF, ...)

Assumendo una politica di rimpiazzo *LRU*, indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato.

901, 43, 84, 92,
86, 98, 722,
88, 95,
96, 721,
802, 95, 727

In tal caso, indicare gli indirizzi dei blocchi che si può presumere si trovino nei buffer nel momento in cui si produce la terza ennupla.

901, 802, 43, 88, 95, 727

Indicare gli indirizzi dei blocchi effettivamente letti da memoria secondaria e caricati nel buffer (nell'ordine) per produrre le prime tre ennuple del risultato, con riferimento ad una politica di rimpiazzo *clock*.

901, 43, 84, 92,
86, 98, 722,
43, 88, 95,
96, 721,
802, 43, 88, 95, 727

Basi di dati II — 10 aprile 2017 — Compito D

Domanda 4 (25%)
Nella figura seguente è schematizzato un piccolissimo buffer con quattro pagine (numerate da 0 a 3), il cui stato viene descritto, per ciascuna pagina, da (i) un intero che indica il numero di pin su di essa (quindi 0 indica che la pagina è libera) (ii) un riferimento al blocco che per ultimo è stato caricato nella pagina; (iii) l'istante in cui è stato effettuato l'ultimo caricamento; (iv) l'istante in cui la pagina è stata per l'ultima volta liberata (se è libera) (v) un booleano che indica se la pagina è sporca (1 indica che è stata modificata dopo il caricamento o dopo l'ultima scrittura).

Pagina del buffer:	0	1	2	3
numero di pin sulla pagina	1	0	1	2
blocco	93	35	47	33
istante load	1	3	9	7
istante unpin		8		
dirty	1	0	0	0

Si supponga ora che vengano eseguite (a partire dall'istante 10, dopo che all'istante 9 è stato caricato il blocco 47 nella pagina 2) le seguenti operazioni (in cui l'argomento delle pin e unpin è il blocco di interesse, con setXXX si indica un aggiornamento di un qualche valore nel blocco e con flushAll il flush dell'intero buffer):

Istante	10	11	12	13	14	15
Operazione	unpin(93)	pin(49)	setXXX(49,...)	unpin(49)	setXXX(47,...)	unpin(47)

Istante	16	17	18	19	20	21	22
Operazione	flushAll	pin(93)	setXXX(93,...)	unpin(93)	pin(49)	unpin(49)	pin(93)

Riempiendo la tabella seguente, indicare, per ciascuna delle strategie e per ciascuna delle pin, (i) quale pagina del buffer viene utilizzata, (ii) se il blocco corrispondente viene letto dal disco e (iii) se viene eseguita una scrittura su disco.

Soluzione corretta solo per i compiti A e C

		naif			LRU			clock		
Istante	Operazione	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?	Pagina	Legge?	Scrive?
11	pin(49)	0	sì	sì	3	sì	no	3	sì	no
17	pin(93)	0	sì	no	0	no	no	0	no	no
20	pin(49)	0	sì	sì	3	no	no	3	no	no
22	pin(93)	0	sì	no	0	no	no	0	no	no