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`

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`

Costo dell'operazione 4 SELECT …:

p + 2

In aggiunta: SELECT eseguita dopo l'operazione 1:

p +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`

Costo dell'operazione 6 SELECT …:

p + 2 + 10