3. Esercizi tipologia C

ESERCIZIO 1

Data la seguente relazione:

MODELLO(codice, capacita', peso, anno_p, cod_produttore)

su cui sono definiti un indice clusterizzato su codice, un indice non clusterizzato su anno_p e un indice non clusterizzato su cod_produttore:

1. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su anno_p implementato come indice hash;

INDICE HASH = USATO PER UGUAGLIANZE. SICCOME IL TESTO PARLA DI ANNO_P DOBBIAMO USARE UN'OPERAZIONE DI UGUAGLIANZA SU ANNO_P. AD ESEMPIO:

```
SELECT *
FROM MODELLO
WHERE ANNO_P = '2003'
```

2. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su anno_p implementato come indice ordinato;

INDICE ORDINATO = B+ TREE QUINDI DOBBIAMO FARE UN'OPERAZIONE DI INEGUAGLIANZA:

```
SELECT *
FROM MODELLO
WHERE ANNO_P BETWEEN 2003 AND 2005
```

3. fornire un esempio di aggiornamento che è velocizzato dalla presenza di tali indici;

BISOGNA USARE UPDATE E SET. NEL SET CI METTO GLI ATTRIBUTI SENZA INDICI E NEL WHERE QUELLI NON CLUSTERIZZATI:

```
UPDATE MODELLO
SET PESO = 1100
WHERE ANNO_P > 2020 AND CODICE = 135
```

4. fornire un esempio di aggiornamento che è rallentato dalla presenza di tali indici.

UN INDICE VIENE RALLENTATO DALLA PRESENZA DI INDICI SE LA CLAUSOLA SET RIFERISCE ALMENO UN ATTRIBUTO INDICIZZATO E LA CLAUSOLA DI QUALIFICAZIONE NON CONTIENE ATTRIBUTI INDICIZZATI.

```
UPDATE MODELLO
SET COD_PRODUTTORE = 'XYZ'
WHERE PESO > 1300
```

ESERCIZIO 2

Data la seguente relazione:

IMPIEGATO(imp#, nome, stip, età, dip#)

con un indice clusterizzato su imp# e un indice non clusterizzato su età:

1. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su imp# implementato come indice hash;
2. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su imp# implementato come indice ordinato;
3. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su età implementato come indice hash;
4. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su età implementato come indice ordinato;
5. la presenza degli indici velocizza il controllo del vincolo di chiave? in che casi?
6. fornire un esempio di operazione di aggiornamento rallentata dall'indice;
7. fornire un esempio di operazione di aggiornamento velocizzata dall'indice.

HASH: VELOCISSIMO PER UGUAGLIANZE

ORDINATO (B+ TREE): OTTIMO PER UGUAGLIANZE E RANGE

CLUSTER: I RECORD SONO FISICAMENTE ORDINATI PER QUELLA CHIAVE

NOTA: "IMP #" È SOLO IL CODICE IDENTIFICATIVO DELL'IMPIEGATO. LA NOTAZIONE "#" INDICA UNA CHIAVE.

1)
```
SELECT *
FROM IMPIEGATO
WHERE IMP # = 12345
```

2)
```
SELECT *
FROM IMPIEGATO
WHERE IMP # > 1234
```

OPPURE

```
SELECT *
FROM IMPIEGATO
WHERE IMP # BETWEEN 2'000 AND 2'009
```

3)
```
SELECT *
FROM IMPIEGATO
WHERE ETA' = 18
```

4)
```
SELECT *
FROM IMPIEGATO
WHERE ETA' BETWEEN 30 AND 40
```

5) SI, IN INSERIMENTO E AGGIORNAMENTO

La presenza di un indice su un attributo che è anche una chiave **velocizza il controllo dell'unicità** del valore. Quando si inserisce un nuovo record o si aggiorna un record esistente, il database deve verificare che il valore per la chiave non sia già presente. Se esiste un indice sulla chiave, questa ricerca è molto più rapida, riducendo significativamente il tempo necessario per la verifica.

In particolare, il vincolo di chiave (primary key) richiede che i valori siano unici e non nulli. L'indice su una chiave primaria è in genere un **indice B-tree**, che è particolarmente efficiente per le ricerche di unicità e l'ordinamento dei dati.

In quali casi la presenza di indici velocizza il controllo del vincolo di chiave?

La velocizzazione avviene principalmente in due casi:

1. **Inserimento di un nuovo record:** Quando si aggiunge una nuova tupla (riga) alla relazione `IMPIEGATO`, il sistema deve verificare che il valore `imp#` (che è la chiave) non sia già utilizzato. Grazie all'indice, la ricerca di `imp#` è molto più veloce rispetto a una scansione completa di tutta la tabella.
2. **Aggiornamento di un record esistente:** Se si aggiorna il valore dell'attributo `imp#` di un record, il database deve nuovamente controllare che il nuovo valore non sia già presente. Anche in questo caso, l'indice permette di effettuare la verifica in modo efficiente.

6) UPDATE IMPIEGATO
SET ETA' = ETA' + 1
WHERE STIP > 1000

7) UPDATE IMPIEGATO
SET STIP = 4'000
WHERE ETA' > 30

REGOLA DA SAPERE PER LA DOMANDA 5:

📌 **Regole/definizione generali da sapere (per qualsiasi esercizio)**

1. **Chiave primaria** = deve essere **univoca** e **non nulla**.
   - Per controllare l'unicità senza scansionare tutta la tabella si usa un **indice univoco**.
   - Molti DBMS (Postgres, MySQL, Oracle) creano automaticamente un indice univoco sulla chiave primaria.
2. **Chiave esterna (foreign key)** = i valori devono **esistere** nella tabella padre.
   - L'indice sulla chiave primaria della tabella padre velocizza il controllo.
   - L'indice sulla colonna figlia non è obbligatorio, ma può velocizzare i `DELETE` / `UPDATE` nella tabella padre (perché il DB deve cercare figli da invalidare).
3. Quindi: **gli indici velocizzano i controlli di vincoli di chiave** perché trasformano verifiche che richiederebbero una *full scan* in una ricerca rapida (puntuale con hash o range con B+tree).