Progettazione fisica
Esercizio

Esempio

Dept(dno,dname,mgr^{Emp})

Emp(ename, age, sal, hobby, dno^{Dept})

Query 1

```
SELECT  E.dno
FROM  Emp E
WHERE  E.age > 40
```

- *Interrogazione con una sola condizione di selezione di tipo intervallo*

- E.age > 40 è fattore booleano
- Cammini di accesso a Emp (oltre a scansione sequenziale): (I_{Emp}(age), age > 40)
- Indice I_{Emp}(age) ordinato (hash non supporta disuguaglianza)
- Accesso con indice più efficiente rispetto ad accesso con scansione sequenziale se la condizione è molto selettiva (restituisce poche tuple)
- Clusterizzazione conveniente (indice secondario)

- Fattore di selettività: probabilità che una tupla di una relazione soddisfi una condizione di selezione
  - Casi favorevoli/casi possibili
  - Numero di valori che soddisfano la condizione/ numero di tutti i possibili valori per l'attributo in Emp
  - Numero di tutti i possibili valori per l'attributo in Emp = V(age, Emp)
  - Max(age, Emp), Min (age, Emp),…
- Age > 40
  - F(age > 40) = (Max(age, Emp) - 40) /V(age, Emp)
  - F(age > 40) vicino a 1 => indice non conviene
  - F(age > 40) vicino a 0 => indice conviene (condizione molto selettiva)

Query 2

```
SELECT  E.dno
FROM  Emp E
WHERE  E.hobby = `Stamps`
```

- *Interrogazione con una sola condizione di selezione di uguaglianza*

- E.hobby = `Stamps` è fattore booleano
- Cammini di accesso a Emp (oltre a scansione sequenziale): (I_{Emp}(hobby), hobby = `Stamps)
- Indice I_{Emp}(hobby) ordinato o hash (in assenza di overflow hash può essere meglio rispetto a indice ordinato)
- Accesso con indice più efficiente rispetto ad accesso con scansione sequenziale se la condizione è molto selettiva
- Clusterizzazione conveniente (indice secondario)

Query 3

```
SELECT  E.dno
FROM  Emp E
WHERE age=30 AND sal=4000
```

- *Interrogazione con due condizioni di uguaglianza (forma normale congiuntiva)*

- Entrambe le condizioni sono fattori booleani
- Cammini di accesso a Emp (oltre a scansione sequenziale):
  - (I_{Emp}(age), age = 30)
  - (I_{Emp}(sal), sal = 4000)
  - (I_{Emp}(age,sal), age = 30 AND sal=4000)
  - (I_{Emp}(sal,age), age = 30 AND sal=4000)
- Accesso con indice più efficiente rispetto ad accesso con scansione sequenziale se la condizione è molto selettiva

Query 3

```
SELECT  E.dno
FROM  Emp E
WHERE age=30 AND sal=4000
```

- Diverse alternative (da valutare empiricamente e tenendo conto di eventuali altre operazioni nel carico di lavoro):
  - 1 solo indice tra $I_{Emp}(age)$ e $I_{Emp}(sal)$ (quello corrispondente alla condizione più selettiva) (poi verrà applicato filtro al risultato)
  - entrambi gli indici $I_{Emp}(age)$ e $I_{Emp}(sal)$, se il sistema è in grado di intersecare i risultati di accessi distinti alla stessa relazione
  - Indice multiattributo $I_{Emp}(age,sal)$ o $I_{Emp}(sal,age)$, ricerca con valore (30,4000)

Query 3

```
SELECT  E.dno
FROM  Emp E
WHERE age=30 AND sal=4000
```

- Gli indici possono essere create ordinati o hash (in assenza di overflow hash può essere meglio rispetto a indice ordinato)
- Clusterizzazione conveniente per qualunque indice creato (indici secondari), se creiamo più di un indice conviene clusterizzare l'indice relativo alla condizione meno selettiva (maggior vantaggio)

- Esempio
  - V(age, Emp) =
  - V(sal, Emp) =

Query 4

```
SELECT  E.dno
FROM  Emp E
WHERE 20<age<30  AND  3000<sal<5000
```

- *Interrogazione con due condizioni di intervallo (forma normale congiuntiva)*

- Entrambe le condizioni sono fattori booleani
- Cammini di accesso a Emp (oltre a scansione sequenziale):
  - (I_{Emp}(age), 20<age < 30)
  - (I_{Emp}(sal), 3000<sal <4000)
  - (I_{Emp}(age,sal), 20<age<30 AND 3000<sal<5000)
  - (I_{Emp}(sal,age), 20<age<30 AND 3000<sal<5000)
- Accesso con indice più efficiente rispetto ad accesso con scansione sequenziale se la condizione è molto selettiva

Query 4

```
SELECT  E.dno
FROM  Emp E
WHERE 20<age<30  AND  3000<sal<5000
```

- Diverse alternative (da valutare empiricamente e tenendo conto di eventuali altre operazioni nel carico di lavoro):
  - 1 solo indice tra $I_{Emp}(age)$ e $I_{Emp}(sal)$ (quello corrispondente alla condizione più selettiva) (poi verrà applicato filtro al risultato)
  - entrambi gli indici $I_{Emp}(age)$ e $I_{Emp}(sal)$, se il sistema è in grado di intersecare i risultati di accessi distinti alla stessa relazione
  - Indice multiattributo $I_{Emp}(age,sal)$ o $I_{Emp}(sal,age)$ non convenienti: nelle foglie si accedono anche a valori della chiave di ricerca che non appartengono al risultato

Query 4

```
SELECT  E.dno
FROM  Emp E
WHERE 20<age<30  AND  3000<sal<5000
```

- Gli indici possono solo essere ordinati (condizioni di intervallo)
- Clusterizzazione conveniente (indici secondari), se creiamo più di un indice conviene clusterizzare l'indice relativo alla condizione meno selettiva (maggior vantaggio)

Query 5

```
SELECT  E.dno
FROM  Emp E
WHERE age=30  AND  3000<sal<5000
```

- *Interrogazione con una condizione di uguaglianza e una di intervallo (forma normale congiuntiva)*

- Entrambe le condizioni sono fattori booleani
- Cammini di accesso a Emp (oltre a scansione sequenziale):
  - (I_{Emp}(age), 20<age < 30)
  - (I_{Emp}(sal), 3000<sal <4000)
  - (I_{Emp}(age,sal), age = 30 AND 3000<sal<5000)
  - (I_{Emp}(sal,age), age = 30 AND 3000<sal<5000)
- Accesso con indice più efficiente rispetto ad accesso con scansione sequenziale se la condizione è molto selettiva

Query 5

```
SELECT  E.dno
FROM  Emp E
WHERE age=30  AND  3000<sal<5000
```

- Diverse alternative (da valutare empiricamente e tenendo conto di eventuali altre operazioni nel carico di lavoro):
  - 1 solo indice tra $I_{Emp}(age)$ e $I_{Emp}(sal)$ (quello corrispondente alla condizione più selettiva) (poi verrà applicato filtro al risultato)
  - entrambi gli indici $I_{Emp}(age)$ e $I_{Emp}(sal)$, se il sistema è in grado di intersecare i risultati di accessi distinti alla stessa relazione
  - Indice multiattributo $I_{Emp}(age,sal)$ conveniente: ricerca con valore (30,3000), nelle foglie si accedono solo valori della chiave di ricerca che appartengono al risultato

Query 5

```
SELECT  E.dno
FROM  Emp E
WHERE age=30  AND  3000<sal<5000
```

- I_{Emp}(age) può essere ordinato o hash
- I_{Emp}(sal) può solo essere ordinato (condizione di intervallo)
- I_{Emp}(age,sal) può solo essere ordinato (condizione di intervallo)
- Clusterizzazione conveniente (indici secondary), se creiamo più di un indice conviene clusterizzare l'indice relative alla condizione meno selettiva (maggior vantaggio)

Query 6

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  D.dname= 'Toy'  AND E.dno=D.dno
```

- *Equijoin e condizione di selezione su Dept*

- Cammino di accesso per Dept oltre a scansione sequenziale:
  - (I_{Dept}(dname), dname = 'Toy')
- Se entrambe le relazioni sono grandi, l'indice I_{Dept}(dname) permette di considerare Dept come relazione outer nell'implementazione nested loop del join e accederla con indice
- Un ulteriore indice I_{Emp}(dno) permette al sistema di considerare anche index nested loop (meglio se clusterizzato)
- In aggiunta, l'indice I_{Dept}(dno) clusterizzato porta il sistema a considerare anche il merge join (ma non sappiamo se verrà usato)
  - In questo caso l'indice I_{Dept}(dname) non verrebbe usato
- Gli indici possono essere ordinati o hash

Query 7

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  D.dname= 'Toy'  AND E.dno=D.dno  AND  E.age=25
```

- *Equijoin, 1 condizione di selezione su Dept e 1 condizione di selezione su Emp*

- Cammini di accesso per Dept oltre a scansione sequenziale: (I_{Dept}(dname), dname = 'Toy')
- Cammini di accesso per Emp oltre a scansione sequenziale: (I_{Emp}(age), age = 25)

Query 7

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  D.dname= 'Toy'  AND E.dno=D.dno  AND  E.age=25
```

- Nella realizzazione del join con una tecnica nested loop, solo la relazione outer può essere acceduta con indice
  - Conviene creare un indice sulla condizione più selettiva (che restituisce meno tuple), la relazione corrispondente diventa outer
  - Per portare il sistema anche a considerare l'index nested loop, è poi necessario creare un indice sull'attributo di join della relazione inner (meglio se clusterizzato)
  - La condizione rimanente viene verificata tramite un filtro sulle tuple risultato del join
- In aggiunta, l'indice clusterizzato sugli attributi di join anche della seconda relazione porta il sistema a considerare anche il merge join (ma non sappiamo se verrà usato)
  - In questo caso gli indici sugli attributi non di join non verrebbero usati
- Gli indici possono essere ordinati o hash

Query 8

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  E.sal BETWEEN 10000 AND 20000
 AND E.hobby= ‘Stamps’ AND E.dno=D.dno
```

- Analogo a Query 7 (ma attenzione perché le condizioni di selezione sono entrambe su Emp, per capire come procedure considerare anche Query 5)

Query 9

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  E.hobby= ‘Stamps’ AND E.dno=D.dno
```

- Analogo a Query 6