Indici non tradizionali

Indici non tradizionali

- Nel seguito introdurremo tre particolari tipi di indici:
  - indici bitmap: permettono di valutare efficientemente condizioni multiple
  - indici per dati multidimensionali
  - indici per documenti digitali

Indici bitmap

- Come abbiamo visto, l'obiettivo di un indice è associare un valore chiave all'insieme delle tuple che contengono tale valore
- In un B-albero, ad esempio, si associa ogni valore chiave ad un puntatore al file dei dati
  - il puntatore è unico se il file è clusterizzato (puntatore alla prima tupla con un certo valore chiave)
  - se il file non è clusterizzato, si può associare il valore chiave ad una lista di puntatori
  - la lista implementa un insieme

Indici bitmap

- L'idea di un indice bitmap è quella di rappresentare l'insieme dei puntatori al file dei dati non come lista ma come bit vector
  - Bitmap
  - **Indice bitmap**: insieme di bitmap per diversi attributi

Indici bitmap

- In una bitmap:
  - Una riga per ogni valore dell'attributo
  - Una colonna per ogni tupla della relazione
  - In una bitmap per l'attributo A:
    - Se bitmap(i,j) = 1, allora la tupla j-esima ha i come valore per l'attributo A
    - Se bitmap(i,j) = 0, allora la tupla j-esima non ha i come valore per l'attributo A
- Esiste una funzione che converte la posizione della colonna in un puntatore alla tupla

Indici bitmap - esempio

Clienti

1	112	Joe	M	3
2	115	John	M	5
3	119	Sue	F	5
4	112	Mary	F	4

Indici bitmap - esempio

Bitmap per sesso

	1	2	3	4
M	1	1	0	0
F	0	0	1	1

Bitmap per valutazione

	1	2	3	4
3	1	0	0	0
4	0	0	0	1
5	0	1	1	0

Indici bitmap

- Gli indici bitmap possono essere utilizzati per valutare:
  - Condizioni di uguaglianza o range su singolo attributo
    - Si usa una singola bitmap
  - Condizioni di uguaglianza o range su più attributi
    - Si usano più bitmap, utilizzando operazioni bit a bit

Indici bitmap - esempio

- Determinare i clienti maschi che hanno ottenuto una valutazione pari a 5

	1	2	3	4
M	1	1	0	0
5	0	1	1	0
	0	1	0	0

- Accedo solo le tuple che soddisfano TUTTE le condizioni

Indici bitmap – confronto con B-alberi

- Vantaggi:
  - Per attributi con bassa cardinalità, lo spazio richiesto è ridotto e le bitmap possono essere mantenute in memoria centrale
    - nessun costo aggiuntivo per accedere l'indice
  - Supportano condizioni che coinvolgono più attributi in modo molto efficiente
  - Facili da costruire e da mantenere
- Svantaggi:
  - Alta occupazione di spazio se costruiti su attributi con alta cardinalità
    - Utilizzo di tecniche compressione (non le vediamo)
  - Non adatte in situazioni molto dinamiche (molte operazioni di aggiornamento)

Indici bitmap – confronto con B-alberi

- I vantaggi nell'utilizzare una bitmap sono maggiori quando l'attributo assume un limitato insieme di valori rispetto al numero di tuple nella tabella
- In generale, se ogni valore viene ripetuto più di 100 volte, allora per l'attributo può essere ragionevolmente creato un indice bitmap
- Per attributi con valori ripetuti un numero inferiore di volte può essere conveniente creare un indice bitmap se compaiono spesso in condizioni complesse nella clausola WHERE di interrogazioni SQL
- Al contrario, i B-alberi sono più convenienti per attributi che assumono molti valori diversi

Indici bitmap in SQL

- La creazione di un indice bitmap in SQL ha la seguente forma:

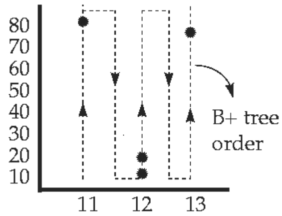
```
CREATE BITMAP INDEX NomeIndice ON
NomeRelazione(NomeAttributo);
```

Indici per dati multidimensionali

- Come abbiamo visto, i B-alberi permettono di indicizzare informazioni mono-dimensionale
  - ogni valore per la chiave può essere visto come un punto in uno spazio mono-dimensionale
- Quando creiamo ad esempio un B+-albero su più attributi, ad esempio due, <età,stipendio>, lo spazio 2-dimensionale viene linearizzato
  - prima si ordina rispetto ad età e poi rispetto a stipendio

Esempio

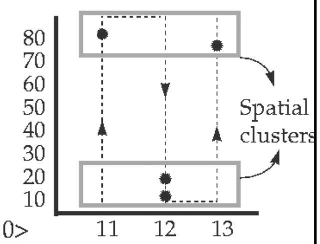
<11, 80>, <12, 10>
<12, 20>, <13, 75>



Indici per dati multidimensionali

- Un indice multidimensionale raggruppa le varie entrate in relazione alla loro vicinanza nello spazio multidimensionale
  - utilizzo di cluster rappresentati da rettangoli o, in generale in uno spazio n-dimensionale, iper-rettangoli

Esempio



<11, 80>, <12, 10>
<12, 20>, <13, 75>

Indici per dati multidimensionali

- I cluster devono essere costruiti sulla base della posizione dei dati nello spazio
- Quali dati si possono indicizzare?
  - Punti n-dimensionali
  - oggetti spaziali in genere (linee, poligono, n-dimensionali)

Indici per dati multidimensionali

- Utili per le seguenti applicazioni:
  - query spaziali
    - trova tutti gli hotel in un raggio di 5 km dalla sede della conferenza
    - trova tutte le città bagnate dal Nilo in Egitto
    - trova il comune più vicino a Genova con più di un certo numero di abitanti
  - query per similitudine (content-based retrieval)
    - data una fotografia di un volto, trovare tutti i volti simili
  - query di tipo range, multidimensionali
    - 50 < età < 55 AND 80k < sal< 90k

Indici per dati multidimensionali

- Si vogliono gestire i cluster utilizzando una struttura bilanciata
- R-alberi: struttura definita per supportare queste esigenze

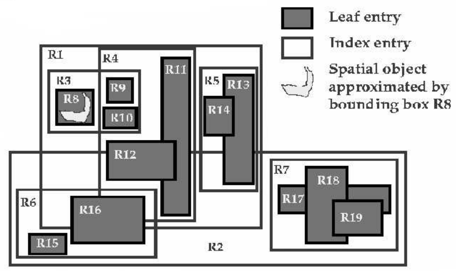
R-alberi

- Indice basato su una struttura ad albero bilanciato
- ogni chiave corrisponde ad un iper-rettangolo, composto da una collezione di intervalli, uno per ogni dimensione
- il sottoalbero associato ad un certo valore chiave E contiene iper-rettangoli contenuti in E
- nello spazio 2-dimensionale, ogni valore chiave rappresenta un rettangolo
  - $a \leq X \leq b$ AND $c \leq Y \leq d$

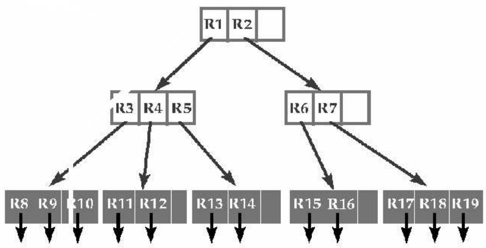
Proprietà degli R-alberi in spazio n-dimensionale

- **Nodi foglia**: contengono elementi del tipo:
  - $\langle l_1,\ldots,l_n,r \rangle$
  - dove l_i è un intervallo sulla dimensione i-esima
  - r puntatore ai dati
- **nodi interni**: contengono elementi del tipo:
  - $\langle l_1,\ldots,l_n,p \rangle$
  - dove l_i è un intervallo sulla dimensione i-esima
  - r puntatore al nodo figlio
- anche per gli R-tree si garantisce il riempimento dei nodi al 50% (esclusa la radice)

Esempio



Esempio (continua)



Puntatori ai dati

R-alberi: ricerca

INPUT: Q = iper-rettangolo da cercare
root = il puntatore alla radice dell'albero
OUTPUT: S = un insieme di iper-rettangoli, inizialmente S = ∅

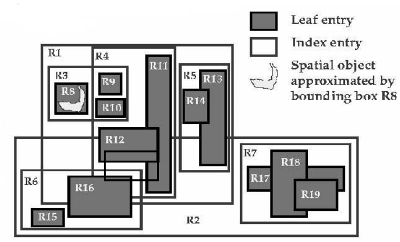
```
procedure ricerca(Q: IN,root:IN,S:IN-OUT):boolean

    P(p) = il nodo puntato da p
    E1, . . . , Ej = iper-rettangoli contenuti in P(p)
    p0, . . . , pj = i puntatori ai figli in P(p)
    if (P(p) non è una foglia) then
            for each i:=1,…,j
                if Ei ∩ Q ≠ ∅  then ricerca(Q,pi,S)  endif
            endfor
    else
            for each i:=1,…,j
                if Ei ∩ Q ≠ ∅  then S := S U Ei endif
            endfor
```

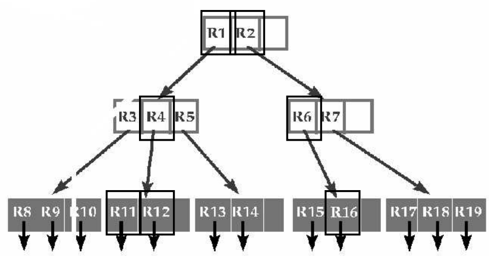
R-alberi ricerca: osservazione

- La ricerca può comportare la scansione di più cammini nell'albero
- questo comportamento è dovuto al fatto che gli iper-rettangoli associati ai nodi interni non sono necessariamente disgiunti

R-alberi ricerca: osservazione



R-alberi ricerca: osservazione



R-alberi - inserimenti e cancellazioni

- Inserimento:
  - si inserisce nell'iper-rettangolo che richiede la minore estensione spaziale per coprire il nuovo oggetto
  - se si supera il limite massimo di entrate
    - splitting: si cerca di minimizzare l'area dei due nuovi iper-rettangoli creati
- cancellazione:
  - anche in questo caso problemi di concatenazione e bilanciamento

R-alberi - estensioni

- Si sostituiscono gli iper-rettangoli con poligoni convessi
- Varianti:
  - R+-alberi: gli iper-rettangoli associati ai nodi interni sono disgiunti
  - ogni ricerca richiede la scansione di un unico cammino ma lo stesso oggetto può essere memorizzato in più foglie

R-alberi in Oracle

- Oracle mette a disposizione tipi particolari per la memorizzazione di dati spaziali
- tali dati possono essere indicizzati utilizzando R-alberi

Indici per documenti digitali

- Necessità di supportare l'accesso a collezioni di documenti digitali
- Applicazioni:
  - Web
  - digital libraries
  - information filtering
- Problemi:
  - le interrogazioni non sono precise ma imprecise e informali
    - es: trovare i documenti che parlano di energia nucleare
    - non siamo interessati a stabilire quali documenti soddisfano la query ma quali documenti sono **rilevanti** per la query

Indici per documenti digitali

- Il settore dell'**Information Retrieval (IR)** propone tecniche per rappresentare i documenti e ritrovarli, a fronte di una certa query
- tecniche di base:
  - i documenti sono completamente non-strutturati
  - ci concentreremo su queste
- tecniche avanzate:
  - assumono che il documento sia associato ad una certa struttura (es. Documenti XML) e permettono ricerche che si basano anche sulla struttura
- Multimedia IR:
  - estensione tecniche IR alla gestione di documenti multimediali

Indici per documenti digitali - interrogazioni

- **Query booleane**:
  - ogni query è una combinazione booleana di parole
    - 'uranium' AND (('nuclear' AND 'energy') OR ('atomic' AND 'bomb'))
  - estensioni di vario tipo:
    - 'bomb*': parole che iniziano con la stringa 'bomb'
      - bomb, bombs, bombing,...
    - prossimità: 'nuclear' ADJACENT 'energy'
      - la vicinanza può essere quantificata precisando un certo numero di parole ammesse tra le due cercate
  - spesso non è facile convertire una richiesta in una query booleana

Indici per documenti digitali - interrogazioni

- **Query ranked**:
  - la query è un'espressione in linguaggio naturale o una sequenza di parole
    - 'Use of nuclear or atomic energy as a power source'
  - si confronta la query con i documenti e si restituiscono quelli con una certa similarità, in genere i primi N, dove N è specificato dall'utente
  - si utilizzano funzioni ad hoc per il calcolo della similarità (es. Cosine measure)
  - la similarità generalmente dipende da:
    - quanti termini della query compaiono nel documento
    - quanto tali termini sono frequenti nel documento
    - discriminazione tra parole più importanti e meno importanti

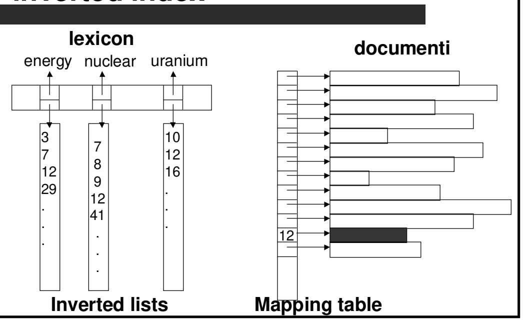
Indici per documenti digitali - esigenze

- Query booleane e ranked:
  - parole presenti nel DB e, per ogni parola, i documenti che la contengono
- query di vicinanza:
  - per ogni parola, l'insieme dei documenti che la contengono e la loro posizione all'interno del documento
    - posizione ordinale (più frequente) o numero di byte dall'inizio del documento
- query ranked:
  - frequenza di ogni parola all'interno del documento

Indici per documenti digitali - inverted index

- Definiti negli anni 70 e attualmente ancora usati
- tre tipi di componenti:
  - **lexicon**: array di parole contenute nella base di dati
  - per ogni parola: **inverted list,** contenente (in relazione al tipo di query che si vuole supportare)
    - identificatore documento
    - frequenza parola nel documento
    - posizioni occorrenze parola nel documento
  - **mapping table**: permette di passare da un riferimento al documento al documento stesso

Indici per documenti digitali - inverted index



Indici per documenti digitali - inverted index: strutture dati

- Lexicon:
  - B+-tree per il lexicon, con puntatori alle inverted list nelle foglie
  - garantisce la memorizzazione ordinata delle parole
- inverted list:
  - clusterizzate
  - uso di tecniche di compressione
- inserimenti e cancellazioni di documenti:
  - costosi: ciascuna operazione richiede di modificare molte entrate del lexicon e molte inverted list
  - approccio batch: vengono gestiti più inserimenti e cancellazioni al lexicon contemporaneamente

Indici per documenti digitali - Signature files

- Idea simile a quella delle bitmap
- ad ogni parola viene associato, tramite una funzione hash chiamata **signature,** un codice binario composto da F bit, di cui esattamente m posti a 1
- ogni documento viene quindi rappresentato da una sequenza di F bit, ponendo a 1 tutti i bit corrispondenti alle parole contenute nel documento

Indici per documenti digitali - Signature files: esempio

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

m (=4 bits/word)
F (=12 bits signature size)

Indici per documenti digitali - Signature files: ricerca

- Per stabilire se un documento contiene una parola, si recupera il codice corrispondente alla parola e si guarda se i bit corrispondenti sono posti a 1 nel codice corrispondente al documento
  - se no: la parola non è contenuta nel documento
  - se sì: la parola **potrebbe** essere contenuta nel documento ma non è certo
- Approccio:
  - si recuperano tutti i documenti che potrebbero contenere la parola
  - si eliminano i **false drops** scandendo sequenzialmente i documenti recuperati

Signatur Indici per documenti digitali - Signature files: esempio

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

data

actual match

Indici per documenti digitali - Signature files: esempio

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

retrieval ↑ ↑ ↑↑

actual dismissal

Indici per documenti digitali - Signature files: esempio

Word	Signature
data	001 000 110 010
base	000 010 101 001
doc. signature	001 010 111 011

nucleotic ↑ ↑ ↑ ↑

false alarm (‘false drop’)

Indici per documenti digitali - Signature files: ricerca

- L’approccio precedente funziona con tutte le query booleane
- idea di base:
  - si costruisce la signature corrispondente alla query
  - si confronta tale signature con quella associata al documento
- esistono ottimizzazioni per evitare la scansione completa signature
  - suddivisione signature files in porzioni,contenenti un numero b di bit
  - scansione delle porzioni che contengono i bit necessari a risolvere la query

Indici per documenti digitali - Signature files: confronto con inverted files

- Signature file molto più compatti
- supporto per query booleane arbitrarie
- inserimento più rapido (viene inserito solo un record)
- problemi nel supportare query avanzate di tipo ranked
- non facilmente esistensibili al supporto di documenti semi-strutturati

Indici testuali in Oracle

- La gestione di documenti testuali è supportata da un tool Oracle
  - Oracle Intermedia Text
- supporta l’indicizzazione e il ritrovamento di informazione testuale, anche strutturata