

Corso
di
Basi di Dati Spaziali

**La progettazione fisica:
gli indici spaziali - 2**

Angelo Montanari

Donatella Gubiani

Indici data-driven

- Gli indici data-driven si adattano alla distribuzione degli oggetti (rettangoli) sul piano
- Ad ogni nodo è associato un rettangolo, detto *Directory Rectangle* (DR), che rappresenta l'MBB dei rettangoli dei nodi figli (o degli oggetti ad esso associati)
- Per accedere ad un rettangolo della collezione indicizzata, si percorre l'albero dalla radice verso una o più foglie, verificando, per ogni Directory Rectangle incontrato, le relazioni di sovrapposizione o di inclusione

Tipologie di indici

- La struttura originaria
R-alberi
- Diverse varianti, tra le quali
R*-alberi
R⁺-alberi

R-alberi

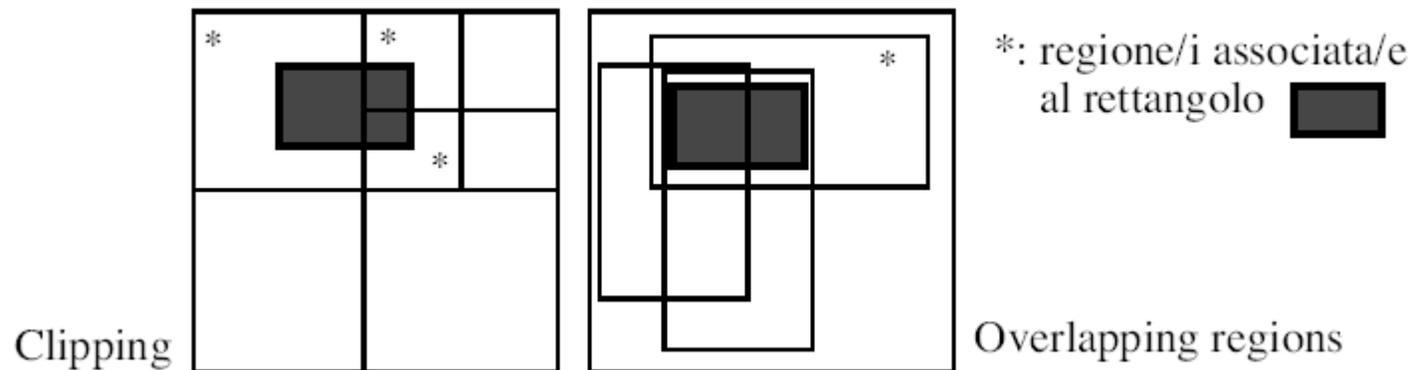
- Sono una estensione multidimensionale dei B⁺-alberi (struttura ad albero bilanciata e paginata)
- Ogni nodo rappresenta un rettangolo. I nodi sono organizzati in base alla relazione di inclusione:
 - ogni nodo foglia contiene un vettore di entry [MBB,OID] e rappresenta un contenitore di oggetti
 - ogni nodo interno contiene un vettore di entry [DR,NODEID] e rappresenta la più piccola area rettangolare (MBB) che contiene gli oggetti rappresentati dai corrispondenti nodi figli

Struttura dell'R-albero

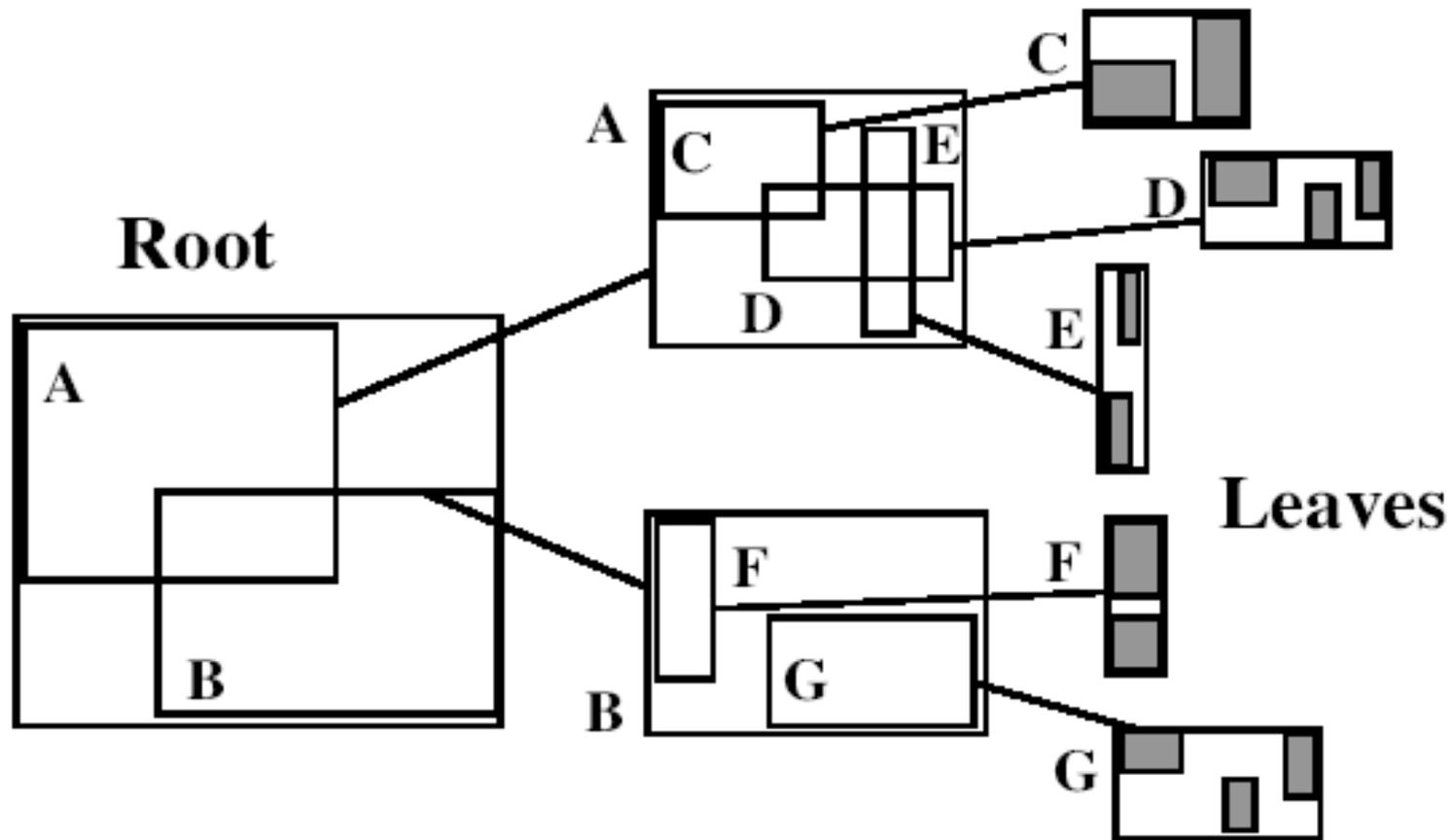
- per ogni nodo dell'albero, ad eccezione della radice, il numero di entry è compreso fra m ed M con $m \in [0, M/2]$
- ogni nodo interno N contiene entry della forma $[DR, NODEID]$, dove DR è un Directory Rectangle di un nodo figlio di N e $NODEID$ è l'indirizzo della pagina corrispondente
- ogni nodo foglia N contiene entry della forma $[MBB, OID]$, dove MBB è un minimal bounding box della geometria dell'oggetto memorizzato all'indirizzo OID
- la radice ha almeno due entry
- tutte le foglie si trovano alla stessa profondità

Regioni: clipping vs. overlapping

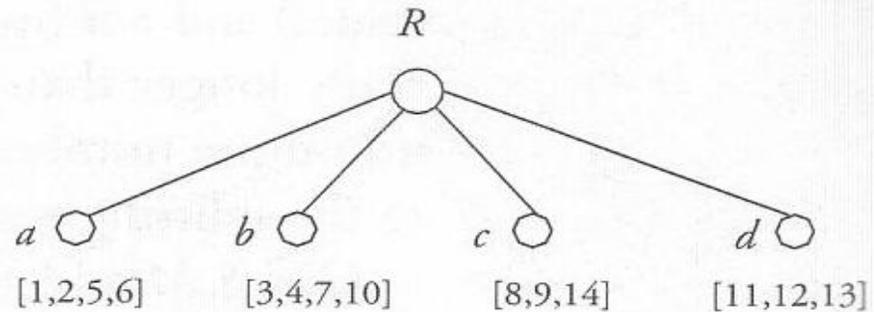
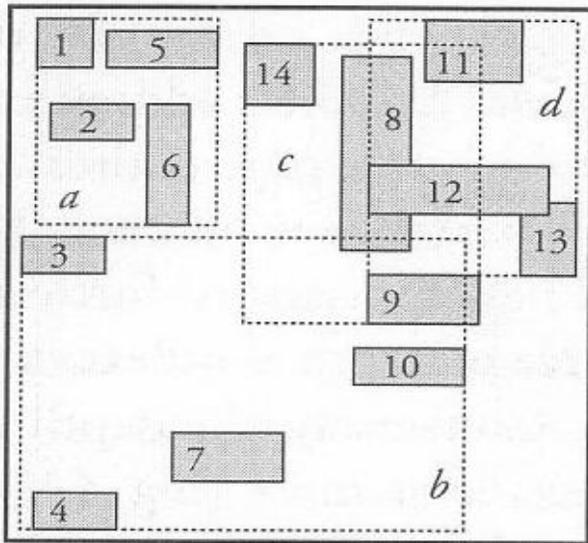
- Dato un poligono, gli R-alberi assegnano tale poligono ad una regione in base alla tecnica di *overlapping* delle regioni
- A differenza di quanto accade nei metodi space-driven, un poligono è assegnato ad una sola regione. Le regioni dell'indice possono, però, sovrapporsi



Esempio 1



Esempio 2



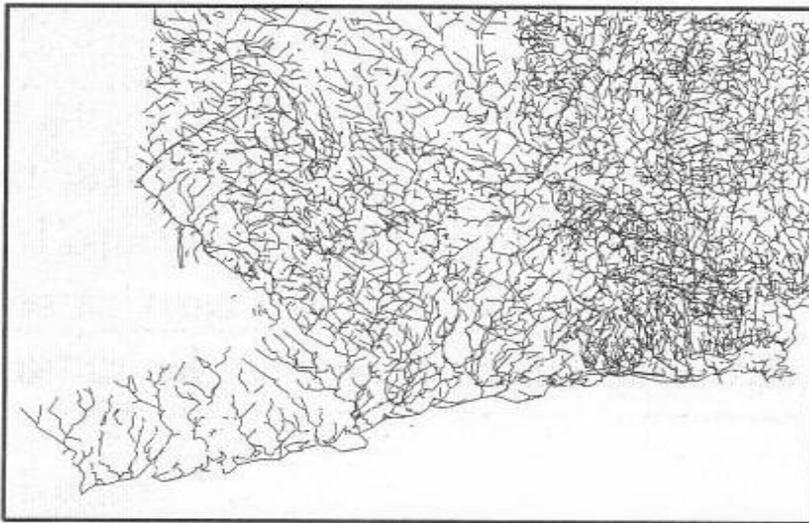
R-albero con $m=2$ ed $M=4$ (i rettangoli delle foglie sono rappresentati da linee punteggiate)

Inserimenti/cancellazioni

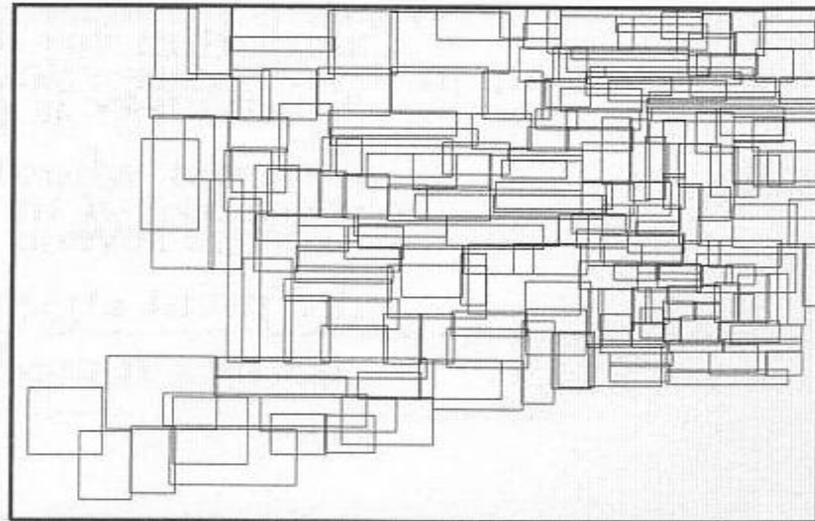
- Le proprietà dell'R-albero sono preservate dalle operazioni di inserimento e cancellazione
- La struttura si adatta alla distribuzione dei dati: a un gran numero di oggetti presente in una certa regione spaziale corrisponde un gran numero di foglie (vicine) dell'R-albero

Un esempio reale

- Directory rectangle delle foglie di un R-albero costruito per raccogliere i dati idrografici dello Stato del Connecticut



(a)



(b)

Dimensionamento di un R-albero

- Il numero massimo M di entry in un nodo dipende ovviamente dalle dimensioni dell'entry e della pagina
- Come nei B^+ -alberi, M può assumere un valore diverso nei nodi interni e nei nodi foglia (OID è l'indirizzo di un record appartenente ad una data pagina ed è, di norma, più grande di NODEID, indirizzo di una pagina)
- La scelta del valore di m dipende dalle strategie di splitting dei nodi

Point query in un R-albero

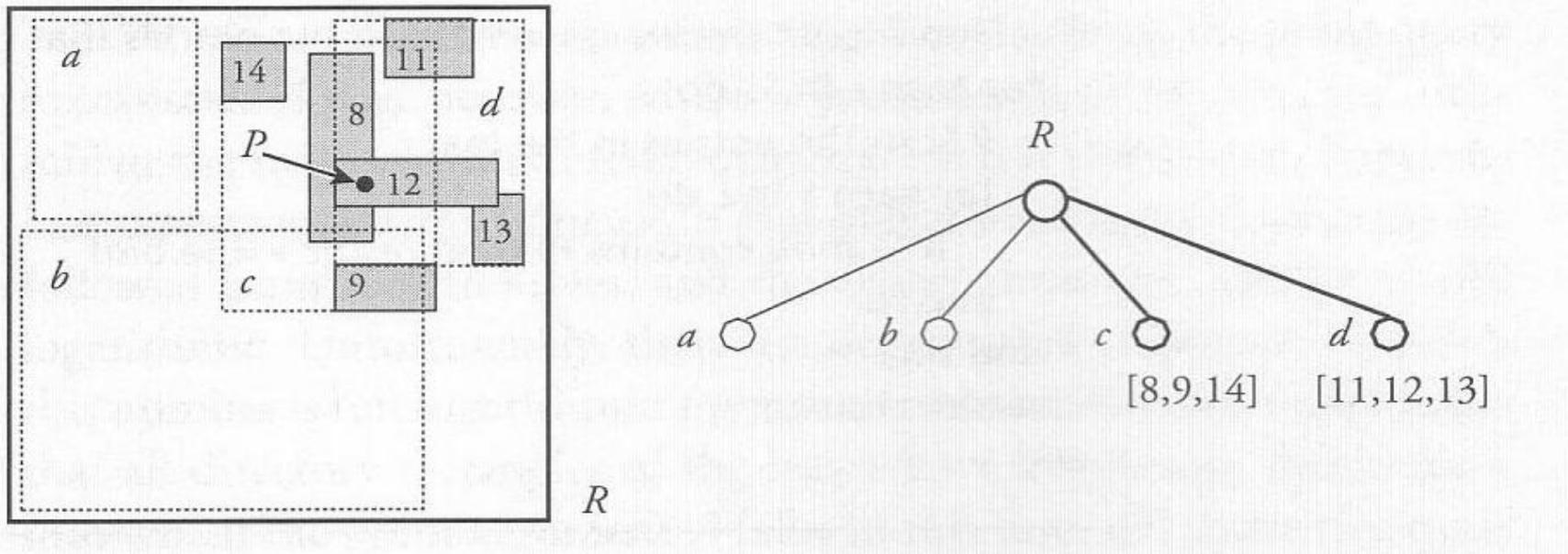
- Viene eseguita in due passi:
 - PRIMO PASSO: seleziona tutti i figli della radice il cui Directory Rectangle contiene il punto P. Ripete tale operazione per tutti nodi individuati, ad ogni livello, fino a raggiungere il livello delle foglie

In ogni nodo interno visitato N, si possono presentare due situazioni:

- se non esiste alcuna entry il cui rettangolo contenga P, la ricerca termina (P cade nello *spazio morto* del nodo)
- se P è contenuto nel rettangolo di una o più entry, si visita il/i corrispondente/i sottoalbero/i

Point query in un R-albero - 2

- SECONDO PASSO: l'insieme delle foglie individuate al termine del primo passo viene esaminato per stabilire se il loro MBB contiene P



NOTA. Le window query sono molto simili alle point query (dettagli omessi)

Point query: algoritmo - 1

```
RT-POINTQUERY (P:point):set(oid)
```

```
begin
```

```
result =  $\emptyset$ 
```

```
#passo 1: attraversa l'albero dalla radice e calcola
```

```
#SL, insieme delle foglie il cui DR contiene P
```

```
SL = RTREETRAVERSAL(root,P)
```

```
#passo 2: scorre le foglie e determina le entry
```

```
#che contengono P
```

```
for each L in SL do
```

```
  #scansione entry nella foglia L
```

```
  for each L in SL do
```

```
    if (e.mbb contains P) then result += {e.oid}
```

```
  end for
```

```
end for
```

```
return result
```

```
end
```

Point query: algoritmo - 2

```
RTREETRAVERSAL (nodeid:PageID, P:point): set of leaves
  begin
  result =  $\emptyset$ 
  #restituisce la pagina associata al nodo
  N = READPAGE(nodeid)
  if (N is in leaf) then return {N}
  else
    #scorre le entry di N e visita quelle
    #che contengono P
    for each e in N do
      if (e.dr contains P) then
        result += RTREETRAVERSAL(e.nodeid,P)
      end if
    end for
  end if
  return result
end
```

Inserimento

- Gli R-alberi sono una struttura di indicizzazione dinamica gestita in modo analogo ai B⁺-alberi
- Inserimento: si scende lungo l'albero scegliendo ad ogni nodo il nodo figlio corrispondente alla regione che soddisfa particolari criteri (ad esempio, minor incremento di area)
- Saturazione di una foglia: si esegue un'operazione di split che propaga le modifiche al livello superiore

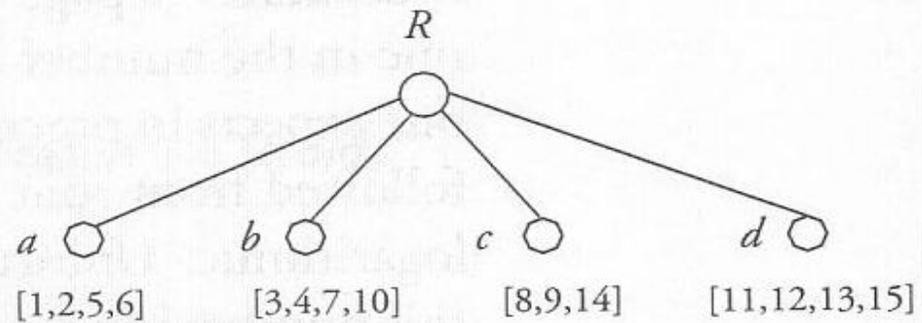
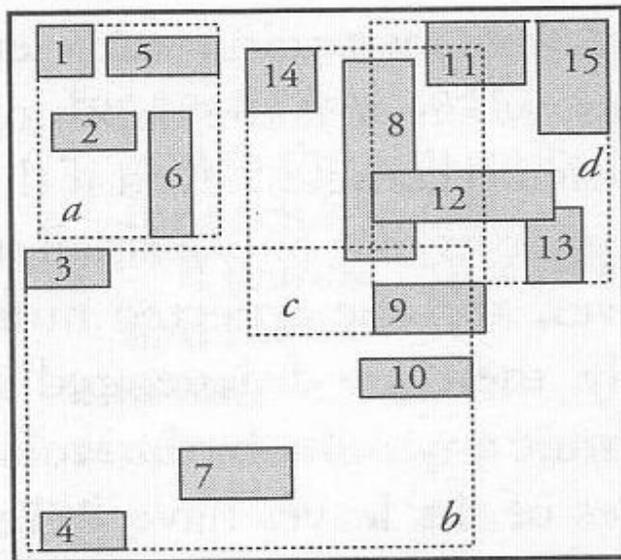
Più in dettaglio.. - 1

- Per inserire un oggetto in un R-albero si attraversa l'albero a partire dalla radice fino a raggiungere una foglia (funzione INSERT)
- Ad ogni livello (nodi interni), si seleziona un nodo figlio tale che l'allargamento del suo DR necessario per contenere l'MBB dell'oggetto da inserire risulti minimo (funzione CHOOSESUBTREE); tale allargamento è nullo se l'MBB è contenuto nel DR
- Raggiunta una foglia, si possono presentare due situazioni:
 - foglia non piena: una nuova entry [MBB,OID] viene aggiunta alla pagina associata alla foglia; se il DR della foglia va allargato, bisogna propagare tale allargamento verso l'alto (funzione ADJUSTPATH). Nel caso peggiore, fino alla radice

Più in dettaglio.. - 2

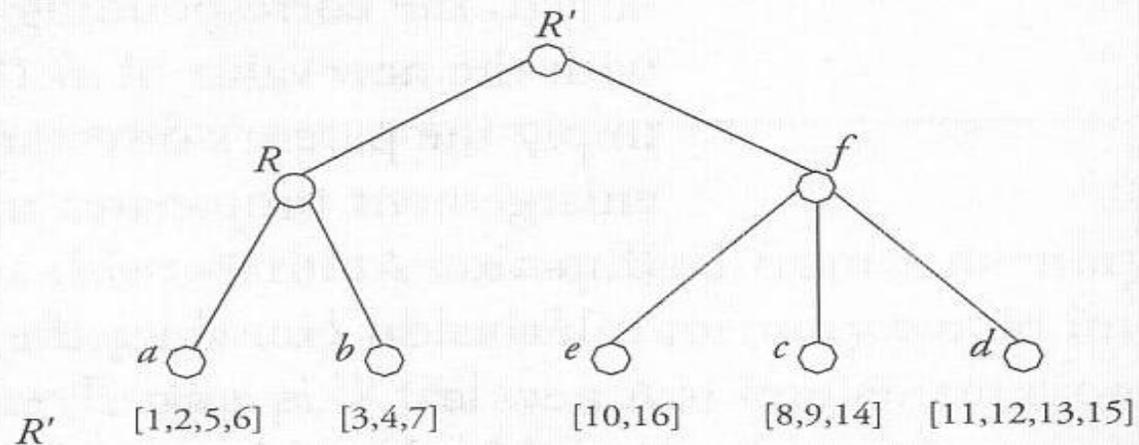
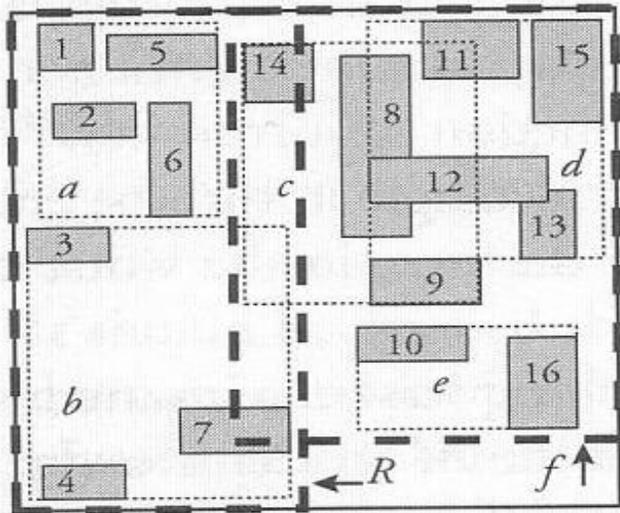
- foglia l piena: deve essere effettuato uno split (funzione SPLIT). Viene creata una nuova foglia l' e le $M+1$ entry vanno distribuite tra l e l' . Terminato lo split, va aggiornata l' entry di l nel nodo genitore f e va inserita in f una nuova entry per l' (funzione SPLITANDADJUST). Tale inserimento può provocare lo splitting di f (se pieno). Nel caso peggiore, lo splitting può propagarsi fino alla radice incrementando di 1 la profondità dell'albero

Esempio 1



Inserimento del rettangolo 15 (assumiamo $M=4$)

Esempio 2



Inserimento del rettangolo 16 (16 va inserito nella foglia b che risulta essere già piena; una nuova entry va inserita anche nel genitore di b che risulta essere a sua volta già pieno..)

Insert: algoritmo

INSERT (e: LeafEntry)

begin

#inizializza la ricerca dalla radice

node = root

#sceglie un percorso

while (node is not a leaf) **do**

node = CHOOSESUBTREE(node,e)

end while

#inserisce nella foglia

INSERTINLEAF(node,e)

#esegui lo split e sistema l'albero se la foglia va in

#overflow, altrimenti sistema il percorso

if (node overflows) **then** SPLITANDADJUST (node)

else ADJUSTPATH (node)

end if

end

AdjustPath: algoritmo

```
ADJUSTPATH (node: Node)
  begin
    if (node is root) then return
    else
      #trova il genitore del nodo
      parent = GETPARENT(node)
      #sistema l'entry del nodo genitore
      if (ADJUSTENTRY (parent,[node.mbb,node.id])
        then
          #entry modificata: sistema il genitore
          ADJUSTPATH (parent)
        end if
      end if
    end if
  end
```

SplitAndAdjust: algoritmo

SPLITANDADJUST (node: Node)

begin

#crea un nuovo nodo e distribuisce le entry

new-node = SPLIT (node)

if (node is root) **then**

 CREATENEWROOT (node,new-node)

else

 #trova il genitore

 parent = GETPARENT(node)

 #sistema l'entry del nodo genitore

 ADJUSTENTRY (parent,[node.mbb,node.id])

 #inserisce il nuovo nodo nel genitore

 INSERTNODE (parent, [new-node.mbb,new-node.id])

 #sistema l'entry del nodo genitore

if (parent overflows) **then** SPLITANDADJUST(parent)

else ADJUSTPATH (parent)

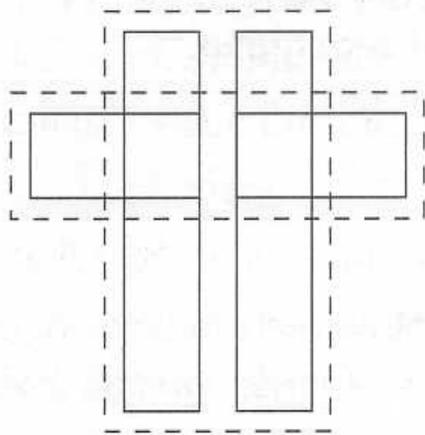
end if

end if

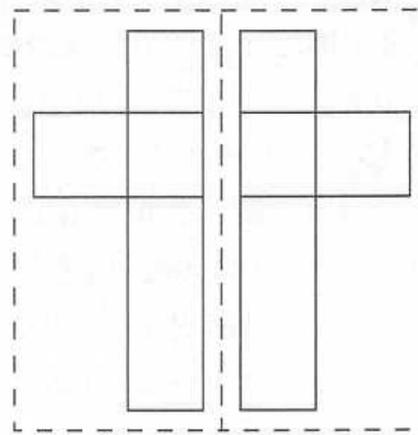
end

Strategie di split

- Ogni strategia che rispetti il vincolo di non avere meno di m entry per nodo è accettabile
- Requisiti fondamentali (non sempre compatibili):
 - minimizzare area (a)
 - minimizzare la sovrapposizione (b)



(a)



(b)

Perché ridurre l'area?

- La minimizzazione dell'area riduce gli spazi morti (si veda l'algoritmo per le point query)

Perché ridurre le sovrapposizioni?

- La presenza di regioni sovrapposte crea dei problemi nella ricerca di uno o più poligoni che contengono un punto dato
 - Non è noto a priori in quale regione dell'indice un poligono è stato inserito e bisogna, quindi, analizzare diversi sottoalberi

Cancellazioni

- La cancellazione di un'entry e da un R-albero è effettuata in 3 passi:
 - PASSO 1. Trovare la foglia L che contiene e
 - PASSO 2. Rimuovere e da L
 - PASSO 3. Riorganizzare l'albero, se necessario.

I passi 1 e 2 sono simili a passi visti in precedenza. Il passo 3 è più complicato. Soluzione semplice: cancellare il nodo e e inserire nuovamente le restanti $m-1$ entry.

Delete: algoritmo

DELETE (e: LeafEntry)

begin

#trova la foglia contenente e

L = FINDLEAF (e)

#rimuove le entry e riorganizza l'albero

#partendo da L ed e;

#il risultato è un insieme Q di nodi

Q = REORGANIZE (L,e)

#reinscrive le entry nei nodi di Q

REINSERT (Q)

end

Reorganize: algoritmo

REORGANIZE (N:Node, e: Entry): set of node entries

begin

Q = \emptyset

#rimuove e da N

N = N - {e}

if (N is not root) **then**

if ($|N| < m$) **then**

 Q = Q U {N}

 #recupera il genitore e lo riorganizza

 F = GETPARENT(N)

 Q = Q U REORGANIZE(F, entry of N in F)

else

 #N modificato: sistema il percorso

 ADJUSTPATH(N)

end if

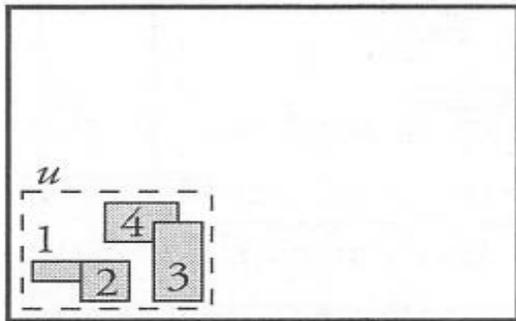
end if

return Q

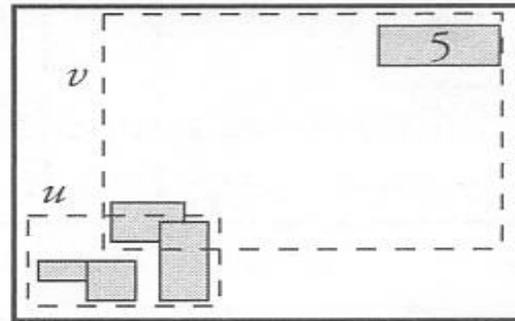
end

Sull'ordine di inserimento

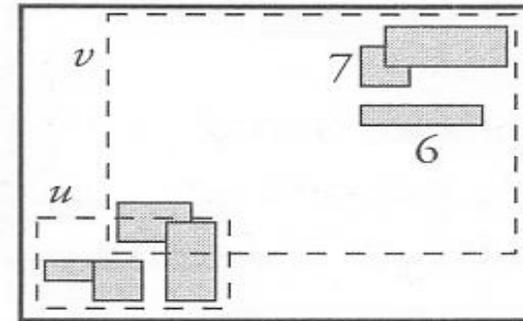
- La qualità dell'indice dipende fortemente dall'ordine di inserimento (rettangoli distribuiti in modo random possono provocare forti sovrapposizioni di nodi e infelici collocazioni delle entry). Si veda, ad esempio, la collocazione del rettangolo 4 al passo b (spreco di spazio)



(a)



(b)



(c)

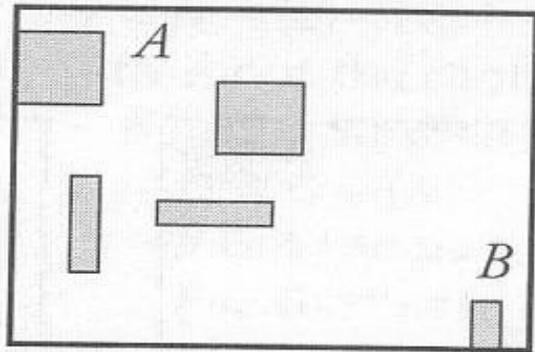
R*-alberi e Packed R-alberi

- R*-alberi e packed R-alberi costituiscono 2 tentativi di superare i limiti dell'organizzazione spaziale degli R-alberi in presenza di inserzioni random
- R*-alberi: sfruttano il reinserimento delle entry per migliorare il clustering degli MBB
- Packed R-alberi: eseguono un pre-processing di un dato insieme di rettangoli per determinare il miglior ordine di inserimento

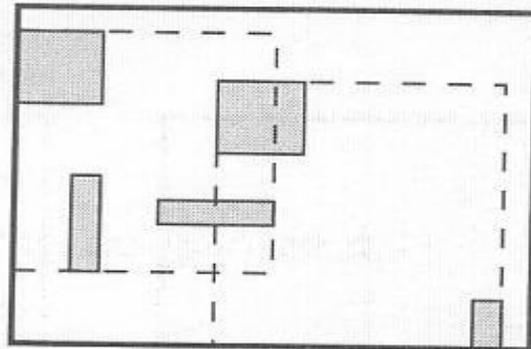
R*-alberi

- R*-alberi realizzano diversi miglioramenti all'algoritmo di inserimento degli R-alberi
- Parametri considerati:
 - perimetro del Directory Rectangle di un nodo
 - sovrapposizione dei nodi
 - area coperta da un nodo

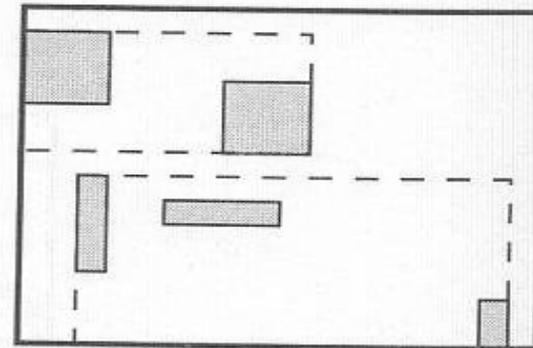
Miglioramento delle strategie di splitting



(a)



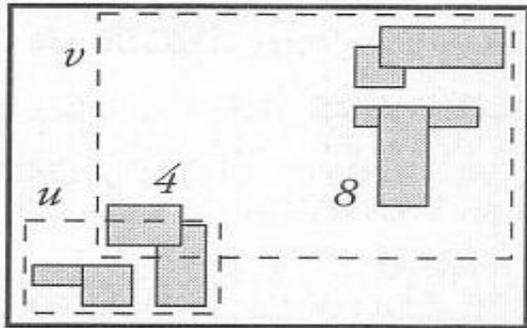
(b)



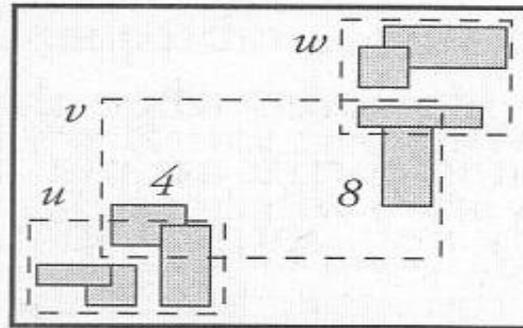
(c)

- (a) nodo in situazione di overflow; (b) split dell'R-albero (definisce la partizione a partire dalle due entry più lontane); (c) split dell'R*-albero (esegue lo split lungo uno dei due assi)

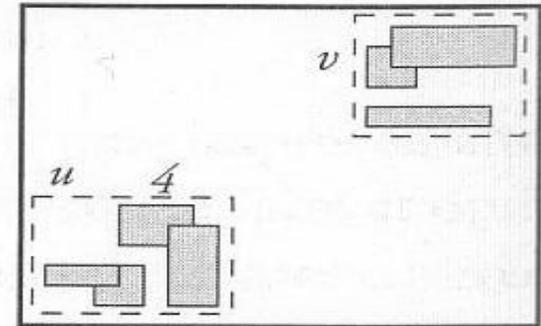
Utilizzo forzato di strategie di reinserimento



(a)



(b)

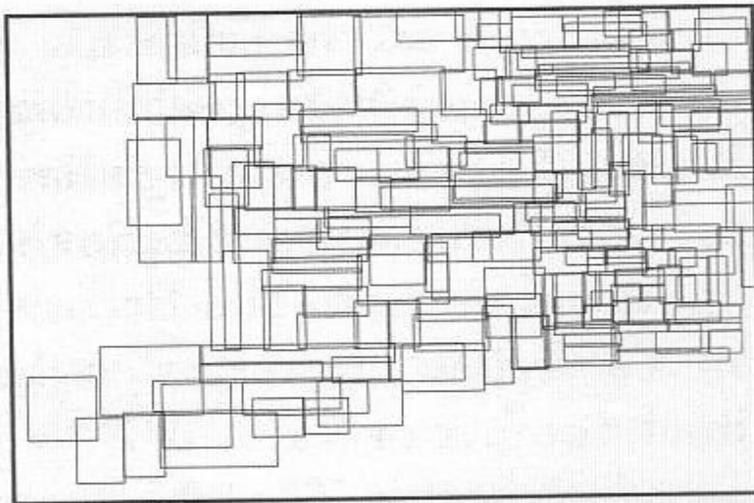


(c)

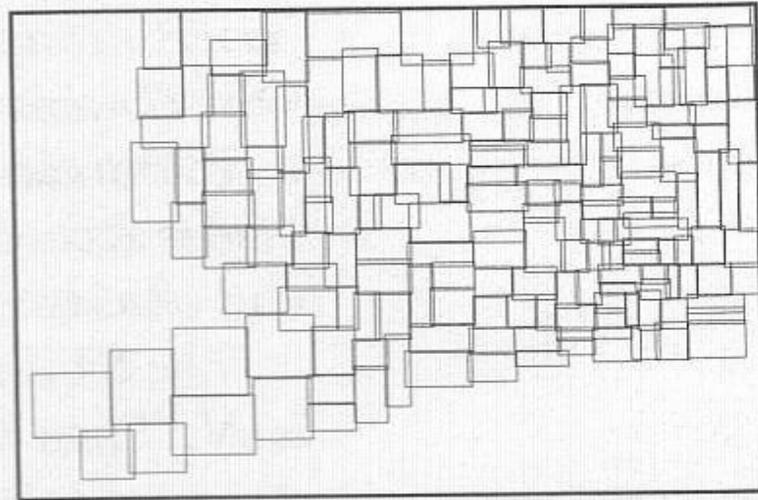
- Reinscrivere alcune entry ogni qualvolta si verifica un overflow in un nodo. Nell'esempio, l'inserimento di 8 causa un overflow (a). Anziché limitarsi ad una riorganizzazione locale (b), si reinscrive preliminarmente in v il nodo 4 che provoca la maggior quantità di spazio morto (c)

R-alberi vs. R*-alberi

- Confronto tra R-alberi e R*-alberi (Stato del Connecticut)



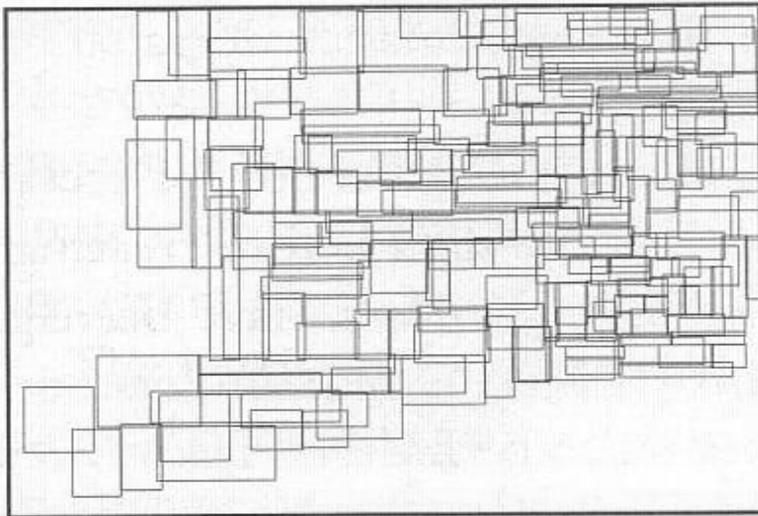
(a)



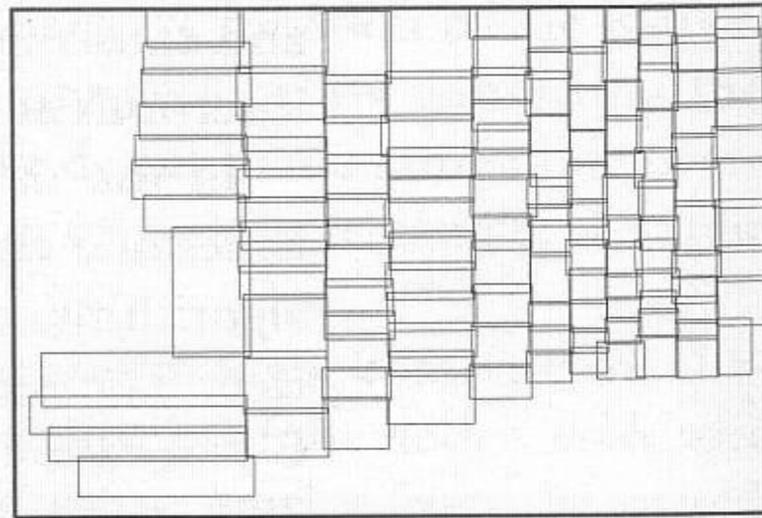
(b)

R-alberi vs. packed R-alberi

- Se la collezione dei rettangoli è stabile nel tempo, si può eseguire un pre-processing dei dati prima di creare l'R-albero (ordinare i rettangoli in base alla loro posizione per massimizzare l'occupazione dei nodi dell'R-albero)



(a)

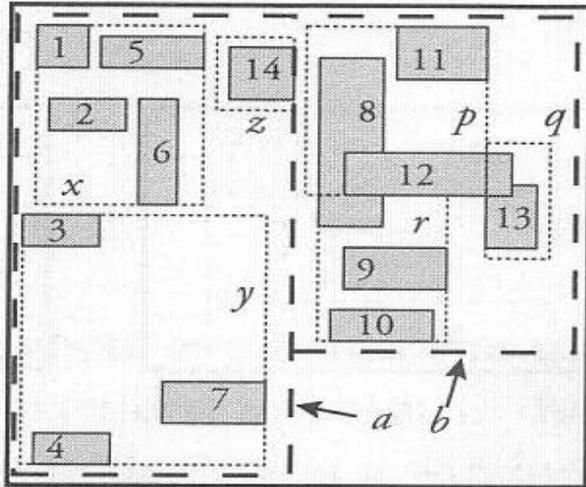


(b)

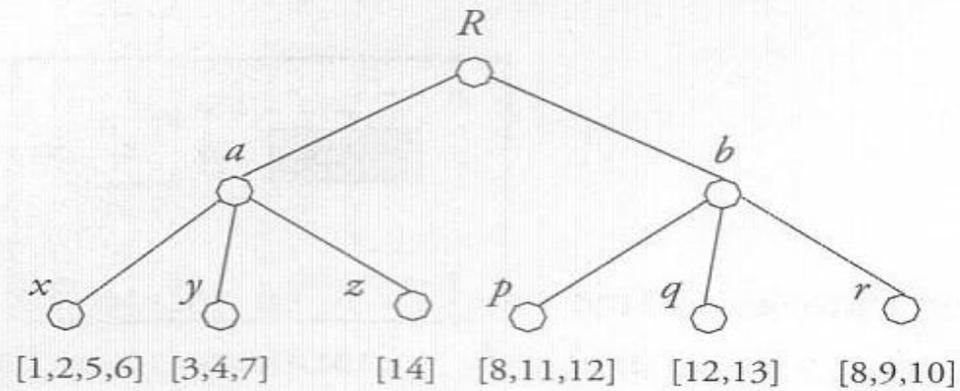
R⁺-alberi

- Obiettivo: ridurre i problemi che nascono dalla presenza di sovrapposizioni fra le regioni
- L'idea è quella di:
 - eseguire il *clipping* delle regioni in modo da ottenere regioni che non si sovrappongono
 - *replicare* l'associazione ad un MBR in ogni regione di livello superiore che lo contiene
- Un R⁺-albero occupa normalmente più spazio di un R-albero, ma tende a migliorare le prestazioni in ricerca. Richiede algoritmi più complessi per l'aggiornamento della struttura

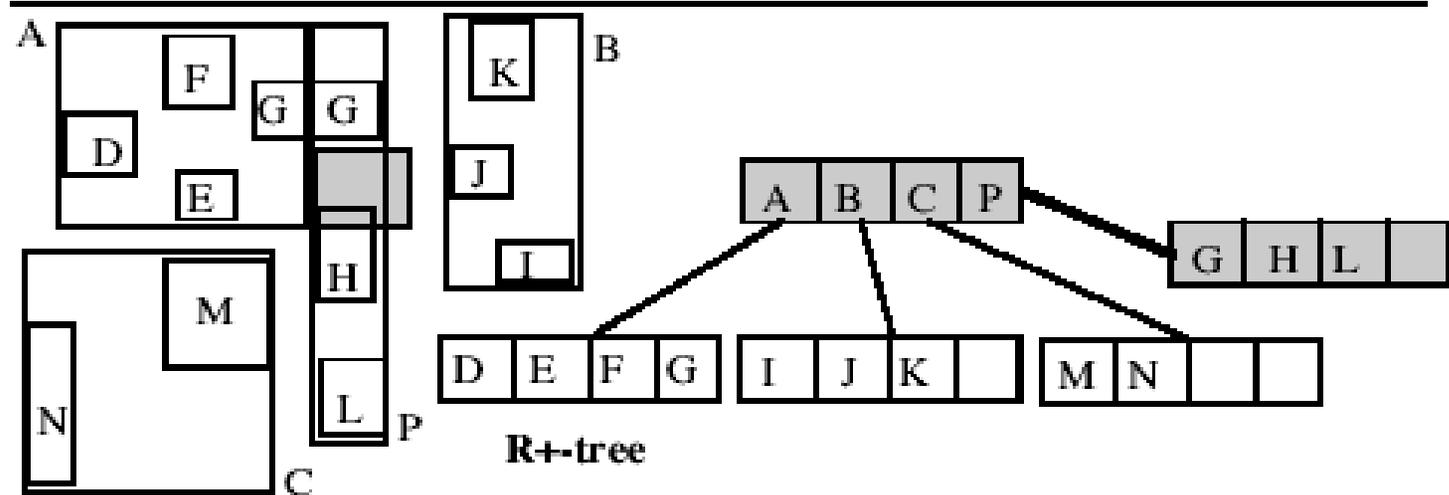
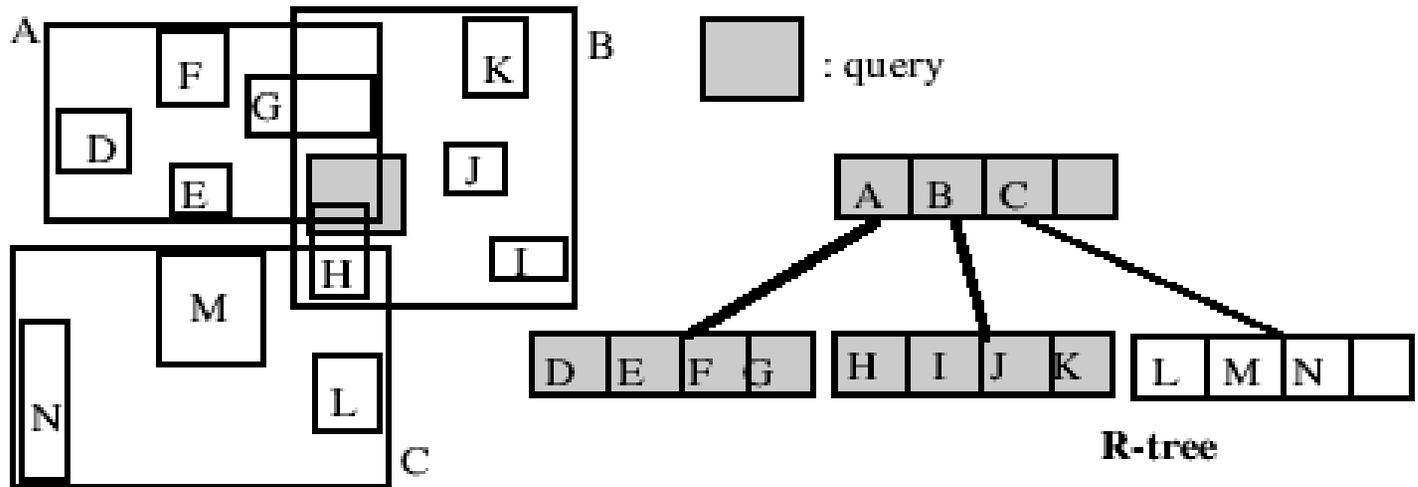
R^+ -alberi: un esempio



R

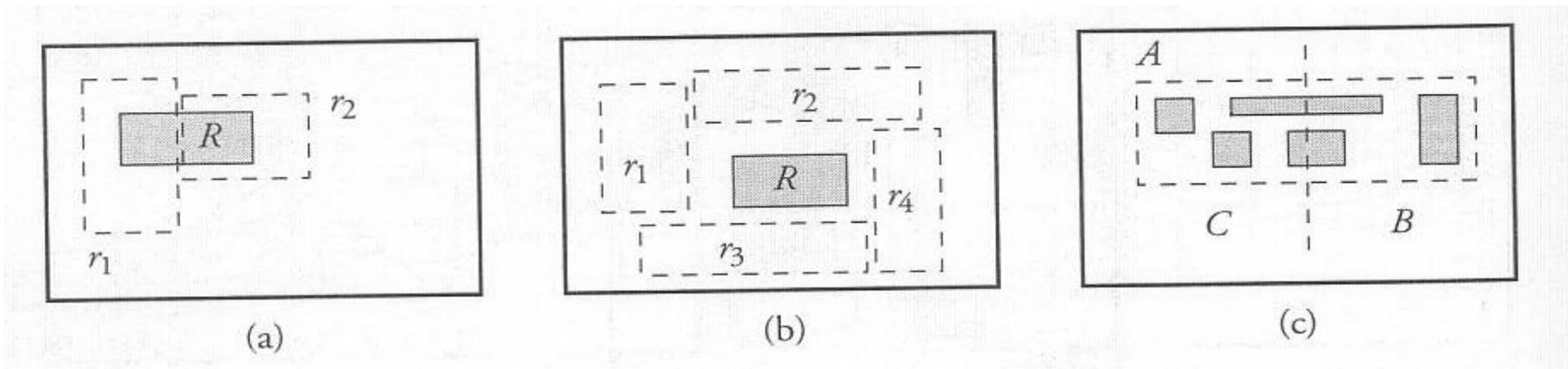


R-alberi vs. R⁺-alberi



Operazioni sugli R^+ -alberi

- Clipping in sottorettangoli (a)
- Deadlock
- Propagazione downward di uno split



Progettazione fisica - 1

- **PROBLEMA:** Come suddividere e memorizzare gli insiemi di valori geometrici presenti in una base di dati geografica?
- **OSSERVAZIONI:**
 - Non è possibile costruire un'unica struttura dati di tipo topologico e/o un unico indice spaziale per tutti i dati contenuti nel sistema
 - Non è sempre utile costruire una struttura topologica/indice per ogni attributo di tipo POINT, LINE o POLYGON contenuto in una relazione (tabella)

Progettazione fisica - 2

- **SOLUZIONE:** Suddivisione della base di dati in *strati (layer)*:
 - Per i dati del medesimo strato viene mantenuta una struttura topologica e/o indice spaziale
 - Non esiste topologia/indice tra strati diversi

Progettazione fisica - 3

- Vincoli tra Strati e Attributi con dominio geometrico in uno schema:
 - ogni strato presente in uno schema contiene solo i valori geometrici di uno o più attributi geometrici appartenenti a relazioni dello schema
 - i valori di un attributo geometrico sono contenuti in uno e un sol strato
- Da ciò deriva che:
 - in uno schema che presenti relazioni (tabelle) con attributi geometrici deve essere presente almeno uno strato
 - in uno schema che presenti relazioni (tabelle) con attributi geometrici, va indicata la corrispondenza attributo geometrico/strato

Esempio - 1

Relazioni

BOSCO (Tipo: STRING, Pg: POLYGON)

PRATO (Pascolo: BOOLEAN, Pg: POLYGON)

Strati

S_MONTAGNA ({POLYGON})

Mapping Attributi geometrici/Strati

BOSCO.Pg . S_MONTAGNA

PASCOLO.Pg . S_MONTAGNA

Esempio - 2

BOSCO

Tipo	Pg
ceduo	8
conifere	23
misto	44

PRATO

Pascolo	Pg
true	9
false	2
false	44
true	14

