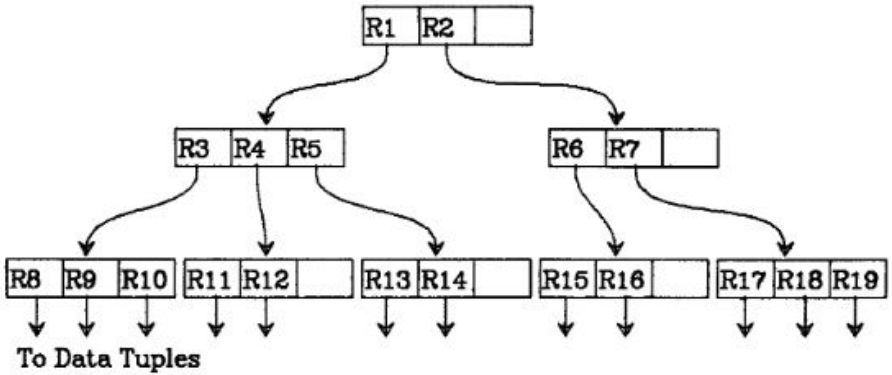
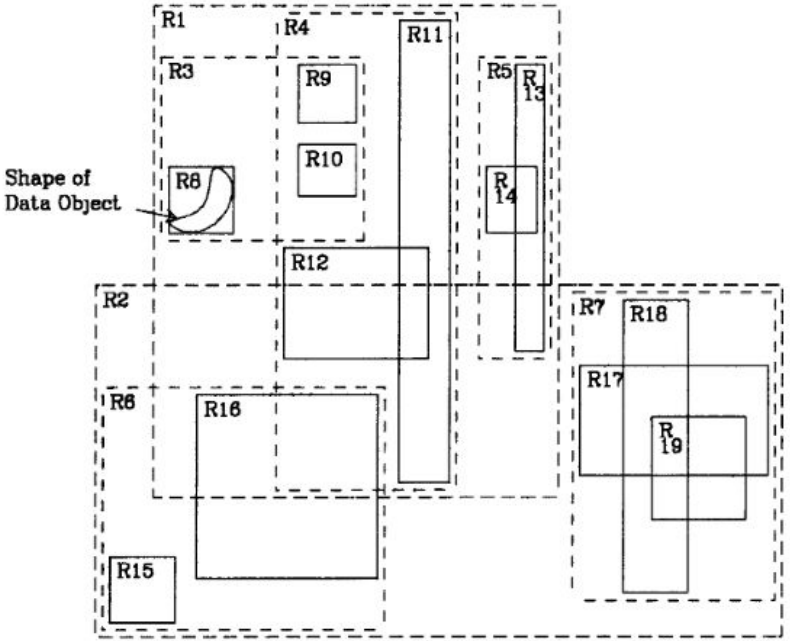
R Tree

R Tree

- Almacenar datos espaciales
  - Datos geolocalizados
    - Ej. Buscar los países cerca de un punto
  - Implementar mapas virtuales
  - Almacenar información espacial de un videojuego
  - Almacenar datos de tipo non-zero size (ej. polígono)

R Tree


R1 R2
R3 R4 R5
R6 R7
R8 R9 R10
R11 R12
R13 R14
R15 R16
R17 R18 R19
To Data Tuples


R1
R4
R11
R3
R9
R5
R13
R10
Shape of
Data Object
R8
R14
R12
R2
R7
R18
R17
R6
R16
R19
R15

R tree

- Height balanced Tree
- Los datos están contenidos en las hojas.
- Podemos insertar y suprimir datos
- Una hoja contiene un tuples(objeto) y un bounding box en n-dimension
- Un nodo no hoja contiene datos, punteros hacia hijos y un bounding box en n-dimension
    - M en la cantidad máxima de hijos y m=M/2 la cantidad **mínima**

R tree

- Cada nodo hoja contiene entre m y M entradas
- Una hoja contiene un bounding box en n-dimension (el rectángulo mínimo en n-dimesion)
- Cada nodo no-hoja contiene entre m y M hijos
- Una nodo no-hoja un bounding box en n-dimension (el rectángulo mínimo en n-dimesion) para el nodo hijo
- las hojas aparecen al mismo nivel

Búsqueda

- Algoritmo :
  - Nodo T
- Search entradas que colision con el Rectangle S

  Si T no es hoja buscar cada entrada E ver si tenemos un overlapping, si hay un overlapping buscar en el subtree de E

  Si T es hoja devolver los datos que overlap con todas las entradas de T

Insertion

- Agregamos en las hojas
  - Si el nodo se sobrecarga se subdivide (node splitting)
  - La subdivisión de nodos se propagan hacia arriba
- Algoritmo : Insertar una nueva entrada E en el R Tree
  - Usar la función **ChooseLeaf** para seleccionar una hoja L
  - Si hay espacio en la hoja agregar E sino SPLIT ! en dos nodos L y LL que contendrá E y todas las entradas de L gracias a la funcion **SplitNode**
  - Propagar los cambios hacia arriba con L (y LL si hubo split) gracias a la función **AdjustTree**
  - Si la propagación de de los cambios provocan un split del root crear un nuevo root que contendrá los dos nodos resultantes

ChooseLeaf

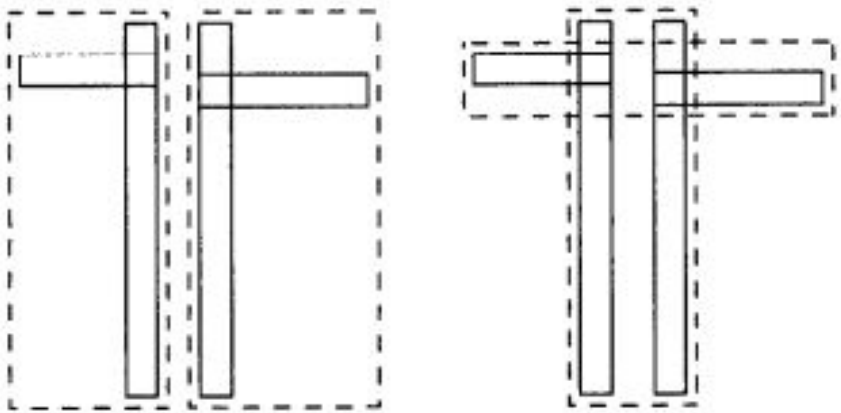
- Entrada E, Root N
- Si N es una hoja escoger N
- Escoger el Nodo N donde el rectángulo necesita el menor crecimiento
- Repetir y descender a partir de nodo N escogido hasta la hoja

AdjustTree

- Ascender del nodo L hasta el root, actualizando los bounding boxes y propagar el split si es necesario
- Algoritmo
  - N=L, NN=LL(si hubo split)
  - si N es el root stop
  - P el padre de N, ajustar el rectángulo para ajustarse a los datos
  - Si existe NN entonces agregar NN a P, si P ya supera el tamaño máximo entonces invocar splitNode

SplitNode

- Dividir un nodo dos para repartir la colección entre ambos nodos
- La suma de las dos áreas debe ser minimizada
-



SplitNode

Algoritmo:

1. Escoger dos entradas para ser la primera entrada de cada grupo con **pickSeeds**
2. Si todos las entradas han sido distribuidas entonces Stop, si un nodo tiene muy pocas entradas, darle las que quedan y Stop
3. Escoger la entrada siguiente con **PickNext** y agregarlo al grupo donde tendrá el menor impacto sobre el área. Repetir de la etapa 2

PickSeeds

- Escoger las dos entradas que maximisan
  - d= Area(E1,E2) - Area(E1)- Area(E2)

PickNext

- Calcular el costo de agregar cada entrada en ambos grupos y calcular
  - d1 y d2
  - Escoger la entrada que maximiza esa diferencia