Generalized Search Trees

Mihail Bogojeski, Alexander Svozil

May 11, 2013

Index

GiST Definition

What is Gist?

- Data Structure that can be used to build a variety of height-balanced search trees.

GiST Definition

What is Gist?

- Data Structure that can be used to build a variety of height-balanced search trees.
- Makes no assumptions about type of data being stored or queries being serviced

GiST Definition

What is Gist?

- Data Structure that can be used to build a variety of height-balanced search trees.
- Makes no assumptions about type of data being stored or queries being serviced
- Allows easy implementations of well known indexed trees like B+-Trees, R-Trees

How is GiST different?

- B+-Tree Functions
  - range predicates (e.g. $c_1 \leq i \leq c_2$)

How is GiST different?

- B+-Tree Functions
  - range predicates (e.g. $c_1 \leq i \leq c_2$)
- R-Tree Functions
  - region predicates (e.g. "find all i such that (x_1, y_1, x_2, y_2) overlaps i")

How is GiST different?

- B+-Tree Functions
  - range predicates (e.g. $c_1 \leq i \leq c_2$)
- R-Tree Functions
  - region predicates (e.g. "find all i such that (x_1, y_1, x_2, y_2) overlaps i")
- GiST Functions
  - GiST can work with any arbitrary predicate and data type (with any number of free variables)

Why use GiST?

- Extensibility in the conext of database systems

Why use GiST?

- Extensibility in the conext of database systems
- Allows the easy evolution of a database system to support new tree-based indexes

Why use GiST?

- Extensibility in the conext of database systems
- Allows the easy evolution of a database system to support new tree-based indexes
- Allows developers to focus on new features of index types without becoming experts in database system internals

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$
- $chooseSubtree :: GiST \rightarrow GiST \rightarrow Entry \rightarrow GiST$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$
- $chooseSubtree :: GiST \rightarrow GiST \rightarrow Entry \rightarrow GiST$
- $split :: GiST \rightarrow Node \rightarrow Entry \rightarrow GiST$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$
- $chooseSubtree :: GiST \rightarrow GiST \rightarrow Entry \rightarrow GiST$
- $split :: GiST \rightarrow Node \rightarrow Entry \rightarrow GiST$
- $adjustKeys :: GiST \rightarrow Node \rightarrow GiST$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$
- $chooseSubtree :: GiST \rightarrow GiST \rightarrow Entry \rightarrow GiST$
- $split :: GiST \rightarrow Node \rightarrow Entry \rightarrow GiST$
- $adjustKeys :: GiST \rightarrow Node \rightarrow GiST$
- $delete :: LeafEntry \rightarrow GiST \rightarrow GiST$

Tree Functions

- $search :: Predicate \rightarrow GiST \rightarrow [LeafEntry]$
- $insert :: Entry \rightarrow GiST \rightarrow Level \rightarrow GiST$
- $chooseSubtree :: GiST \rightarrow GiST \rightarrow Entry \rightarrow GiST$
- $split :: GiST \rightarrow Node \rightarrow Entry \rightarrow GiST$
- $adjustKeys :: GiST \rightarrow Node \rightarrow GiST$
- $delete :: LeafEntry \rightarrow GiST \rightarrow GiST$
- $condenseTree :: GiST \rightarrow Node \rightarrow GiST$

Key Functions

- $consistent :: Entry \rightarrow Predicate \rightarrow Bool$

Key Functions

- $consistent :: Entry \rightarrow Predicate \rightarrow Bool$
- $union :: [Entry] \rightarrow Predicate$

Key Functions

- $consistent :: Entry \rightarrow Predicate \rightarrow Bool$
- $union :: [Entry] \rightarrow Predicate$
- $penalty :: Entry \rightarrow Entry \rightarrow Integer$

Key Functions

- $consistent :: Entry \rightarrow Predicate \rightarrow Bool$
- $union :: [Entry] \rightarrow Predicate$
- $penalty :: Entry \rightarrow Entry \rightarrow Integer$
- $pickSplit :: [Entry] \rightarrow [[Entry]]$