# プログラムの中で使用するのに適したデータベースJungle

金川竜己　並列情報研

## 研究概要

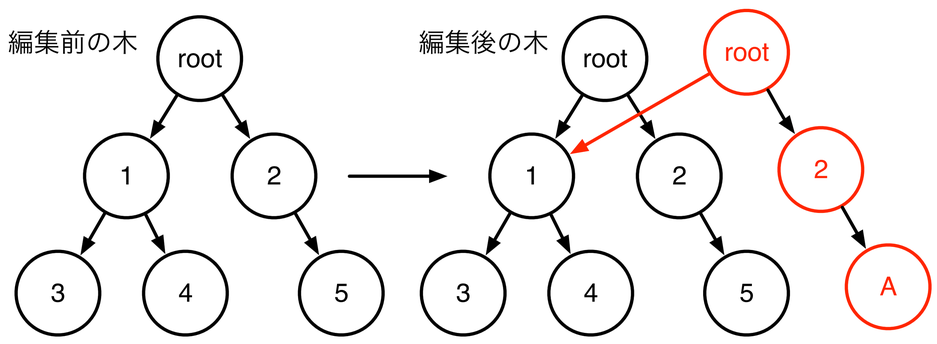
・ プログラムからデータを分離して扱うデータベースには、プログラム中のデータ構造とRDBの表構造とのインピーダンスミスマッチという問題がある
・ 本研究室ではその問題を解決するため、プログラム内部に木構造を格納できるデータベースJungleを開発している
・Jungleは読み込みは高速に行える反面、書き込みの手間は木の形、大きさに依存しており最悪の場合O(n)となってしまう
・本研究では、Jungleの木の編集機能の改善を行った
・性能測定では既存のJungleとくらべて大幅な速度向上を確認した

## 非破壊赤黒木の実装

・Jungleは過去の版の木を全て保持している。
・Jungleは過去に木に対する検索もサポートしているため、全ての版にIndexを持つ必要がある
・従来のJungleのIndexは破壊的赤黒木で実装されており、木に更新が入るたびに新しいIndexをO(n)で作り直していた
・非破壊赤黒木を実装することで、Indexの更新をO(log n)で行えるようになった
・非破壊赤黒木は、編集時変更されない部分は過去の版と共有されるのでメモリ効率も改良された

## 非破壊的木構造

・一度作成したデータは変更しない
・新しい木構造を作成することでデータの編集を行う



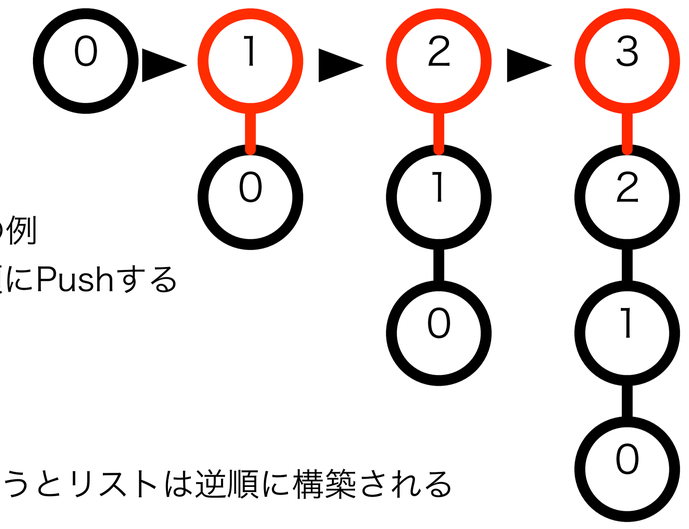

通常の木構造と異なり並列に読み書きが可能である
ロックが必要になるのは新しいルートノードを登録する時のみのため、通常の木構造よりロックが少ない
一度構築した木を変更しないため、複数の版の木が同時に存在できる

## 線形の木の取り扱い

・Jungleは木の編集時、木の複製を行う
・木の変更の手間は木構造の形によって異なり、特に線形の場合変更の手間がO(n)になってしまう
・線形の木をO(1)で変更するためにPush/Pop操作と差分木を実装した

## Push/Pop

・線形の木のルートの上にノードを付け加える操作(Push)
・線形の木のルートを取り除き、その子供をルートにする操作(Pop)
の２つの操作のこと。どちらも実行オーダーはO(1)で線形の木の長さを変更できる。

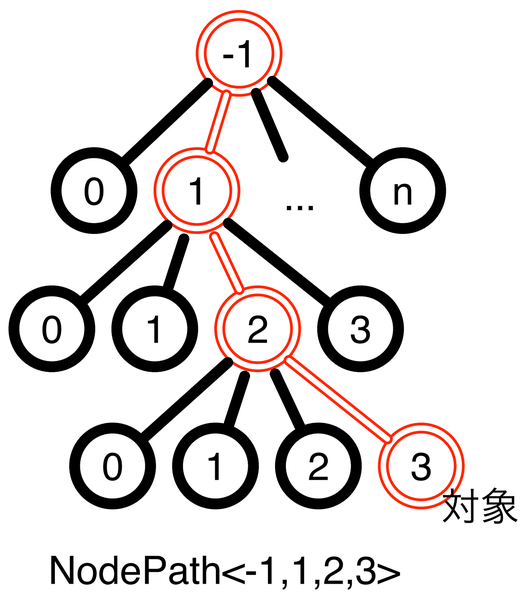



Push操作の例
0,1,2,3の順にPushする

・Pushを連続で行うとリストは逆順に構築される

## Jungle Treeでのノードの指定

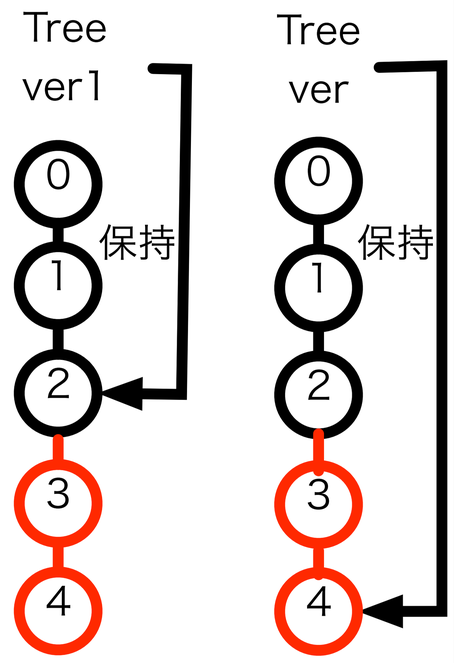
・Jungleでは木のノードの位置をNodePathを使って表す
・NodePathはルートから対象のノードまでの経路を数字で指し示す
・ルートノードは例外として-1と表記される
・NodePathは、木に編集を加える際などに使用する




NodePath<-1,1,2,3>

## 差分木

・Logなどデータの並びが正順でないといけないデータもあるため、正順の木をO(1)で構築できる差分木の実装を行った
・差分木は、版毎に自身の最後尾のノードを持つ
・編集によって木の形は変更されるが、各版が持つ末尾のノードを超えないようにアクセスすることで、線形木の非破壊性を維持することができる




## 本研究でJungleに追加した構成要素

・非破壊赤黒木
・Indexの差分Update
・差分木
・Red Black Jungle Tree