

Use Rust To Make A TSDB

rust 入门基础 (六)

Lecturer: Rose

Date: 2022.06.14

Welcome to follow the GitHub repo 欢迎关注我们的代码仓库

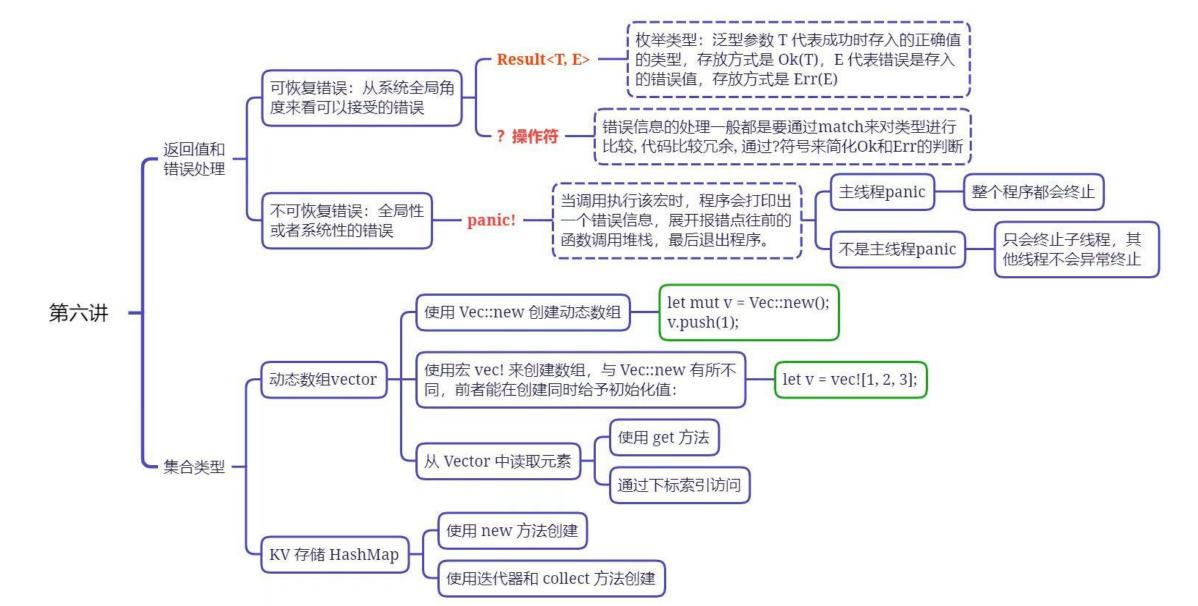
https://github.com/cnosdb/cnosdb







rust 入门基础 回顾



rust 入门基础



指针是个通用概念,它表示内存地址这种类型,其引用或"指向"其他数据。 Rust中的指针是"第一类公民"(first-class values),可以将它们移动或复制, 存储到数据结构中并从函数中返回

Rust提供了多种类型的指针

- 引用(Reference), 共享引用&, 可变引用&mut
- 原生指针,又叫裸指针(Raw Pointer), *const和*mut
- 智能指针 (Smart Pointer) , Box, Rc

rust 入门基础



Rust 可以划分为 Safe Rust 和 Unsafe Rust。

• 引用 主要应用于 Safe Rust。

在Safe Rust 中,编译器会对引用进行借用检查,以保证内存安全和类型安全。

• 原生指针 主要用于 Unsafe Rust。

原生指针不在 Safe Rust 的控制范围内,需要编程人员自己保证安全。

rust 入门基础 智能指针—Box<T>

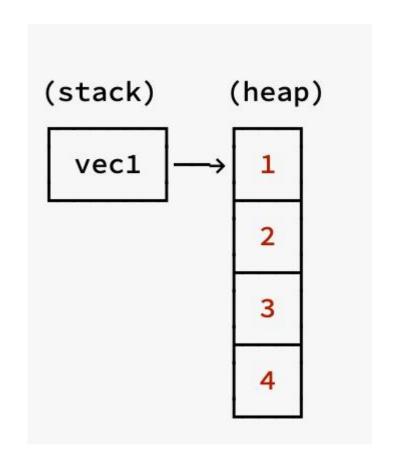


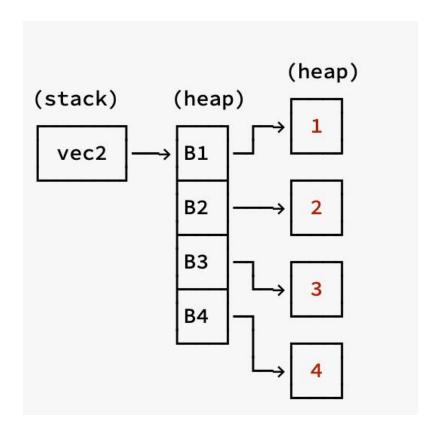
Box<T>

- Rust 中的值默认被分配到 栈内存,通过 Box<T> 在堆内存中分配。
- Box<T> 是指向类型为T的堆内存分配值的智能指针。
- 当 Box<T> 超出作用域范围时,将调用其析构函数,销毁内部对象,并自动 释放内存。
- 可以通过解引用操作符 * 来获取Box<T>中的 T
- Box<T> 的行为像引用,并可以自动释放内存,所以称为智能指针。

rust 入门基础









rust 入门基础 Deref 和 Drop



智能指针往往是基于结构体实现,它与我们自定义的结构体最大的区别在于它实现了 Deref 和 Drop 特征

Deref 可以让智能指针像引用那样工作,这样你就可以写出同时支持智能指针和引用的代码,例如 *T

Drop 允许你指定智能指针超出作用域后自动执行的代码,回收内存资源例如做一些数据清除等收尾工作



Q&A

Welcome to follow the GitHub repo 欢迎关注我们的代码仓库 https://github.com/cnosdb/cnosdb



