MemoryContext

概述

在 PostgreSQL 的旧版本中，常常需要处理大量以指针传值的查询，因而存在着内存泄露的问题，直到查询结束时才能将内存收回。尤其是在处理 TOAST 数据时，需要使用大量的内存，因而使得内存泄露的问题更加明显

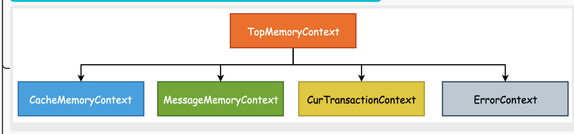
为此，PostgreSQL 在 7.1 版本开始实现了内存上下文管理机制

内存上下文机制本质上就是对内存进行分类和分层

- 一条 SQL 从用户发送给 postgres 进程，到返回结果这个过程中，需要多次地进行内存分配
- 我们们需要为用户发来的命令，例如 "select * from t"，开辟一个内存空间并存储它，PG 使用 MessageContext 来存储
- 对于不经常改变的 Catalog Relation 可以放入缓存中，不必每次都从磁盘中读取，那么 Cache 所需的内存就可以由 CacheMemoryContext 进行管理
- 当执行一个事务时，一定会伴随着内存分配，比如元组的扫描、索引的扫描或者元组的排序等等，这些内存可能需要在事务结束后才释放，因此可由 CurTransactionContext、ExecutorState 来管理

在数据库运行过程中，会不断地申请各种各样的内存，PostgreSQL 将其分门别类整理好，在内存释放时就将更加从容和方便。即系统中的内存分配操作在各种语义的内存上下文中进行，所有在内存上下文中分配的内存空间都通过内存上下文进行记录

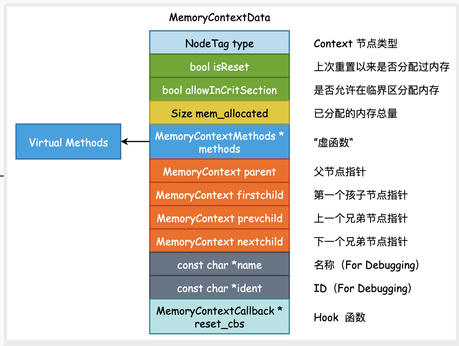
因此可以很轻松地通过释放内存上下文来释放其中所有的内存，而不用费心地去释放其中的每一块内存



PostgreSQL 中的每一个子进程都拥有多个私有的内存上下文，这些上下文将会组成一个树形结构，更准确地说，是一棵多叉树，根节点为 TopMemoryContext

MemoryContextData

MemoryContextData 可以是一个抽象类，包含了内存上下文之间的关联关系，以及对内存上下文进行操作的一系列函数，可以有多种实现，但目前只有 AllocSetContext 这一种实现

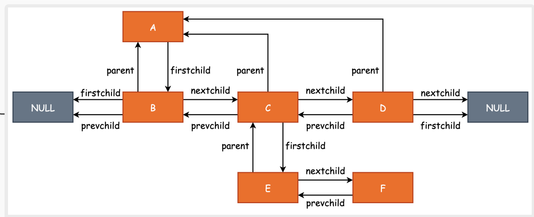


isReset

- 表示当前内存上下文从上一次重置到当前是否还没有内存分配，初始值为 true，即重置以来还没有进行内存分配。当进行了内存分配时，该值将会被更新为 false
- 那么最终在重置内存上下文时，如果发现该字段为 true，则表示该内存上下文还没有进行过内存分配，就可以不进行实际的重置工作，从而提高效率

parent、firstchild、prevchild 和 nextchild

构成了内存上下文的树形结构，并且每一个节点都保存了其父指针和兄弟指针，也就是说，只要能拿到树中的任意一个节点即可对整棵树进行遍历

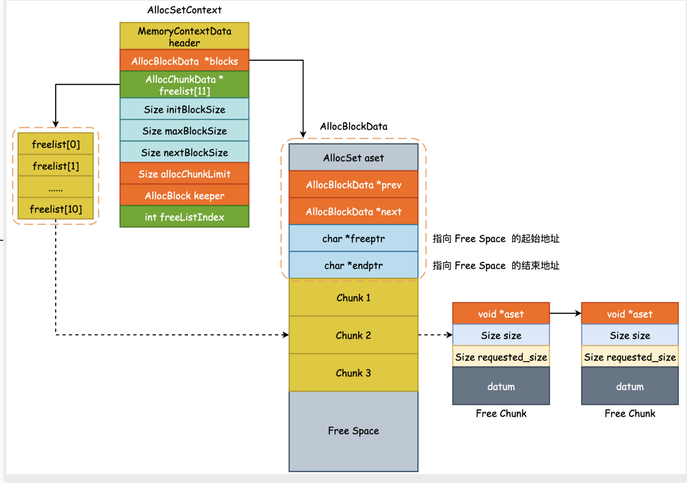


因此，MemoryContextData 最重要的作用便是管理各个内存上下文之间的关联关系，在清除一个内存上下文时，将会遍历该节点的所有子节点并对其进行释放

AllocSetContext

Region-Based Memory Management

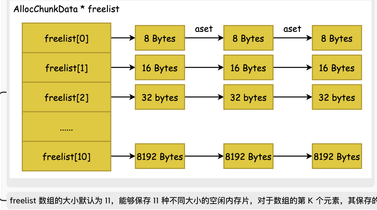
- PostgreSQL 将内存分为内存块（Block）和内存片（Chunk），其中内存块是通过 malloc() 这一系统调用取得的
- 一个内存块中将会有一个或者多个内存片，内存片才是 PostgreSQL 的最小存储单元
- 简单的理解就是 PostgreSQL 首先向操作系统要一块比较大的内存（Block），然后在对这一块大内存进行切割（Chunk），把切割之后的内存返回给调用方
- 使用 malloc 申请较大的内存块，然后将该内存块切割成一个一个的小的内存片，将内存片返回给调用方。当调用方使用完毕返还时，并不会直接返回给操作系统，而是添加至 Free List 这一空闲链表的指定区域内，以用于下一次的内存分配



AllocBlockData 就是我们上面所说的大的内存块，由 malloc() 调用申请。而 AllocChunkData 则是小的内存片，一个 Block 中将会有一个或者多个内存片，空闲内存片之间使用单向链表这一数据结构保存

freelist

由 aset 指针组成的空闲内存片链表（freelist）相当重要，这些空闲内存片将用于再分配，并且有着多种不同大小的内存片以供分配



freelist 数组的大小默认为 11，能够保存 11 种不同大小的空闲内存片，对于数组的第 K 个元素，其保存的内存片大小为 2^(K+2) 字节

K 从 1 开始取值，也就是说，freelist 数组中最小的内存片大小为 8 Bytes，最大的内存片为 8192 bytes（默认情况下），相同大小的内存片由链表链接

值得注意的是，所有 freelist 中的内存片的大小都为 2 的指数，当需要一个大小为 size 的内存时，将会根据向上取整的规则取出 freelist 中的空闲内存片

内存分配概览