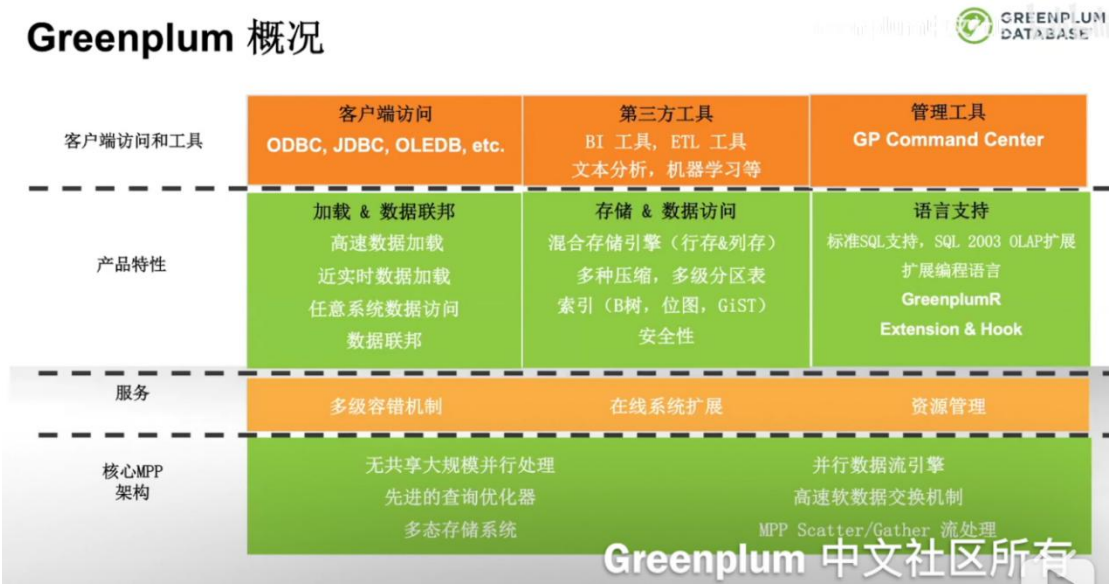
Greenplum 架构和核心引擎

学习地址

https://www.bilibili.com/video/av81898649?p=1

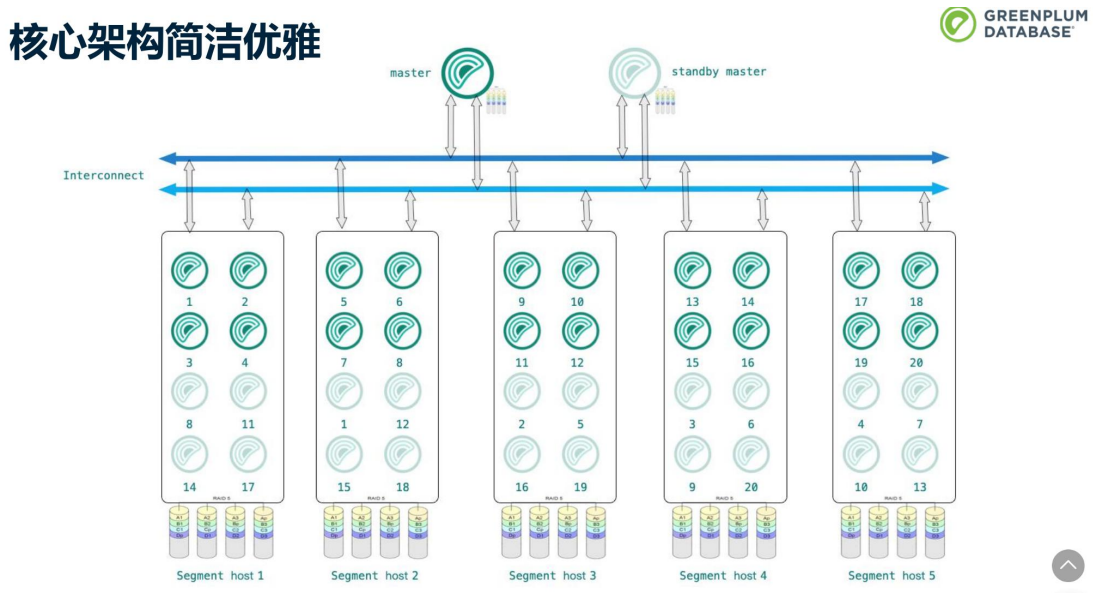
1 Greenplum 架构概述

1.1 概述简介



ODBC 与 JDBC 都是基于标准的 SQL 来执行的,支持很好的第三方工具

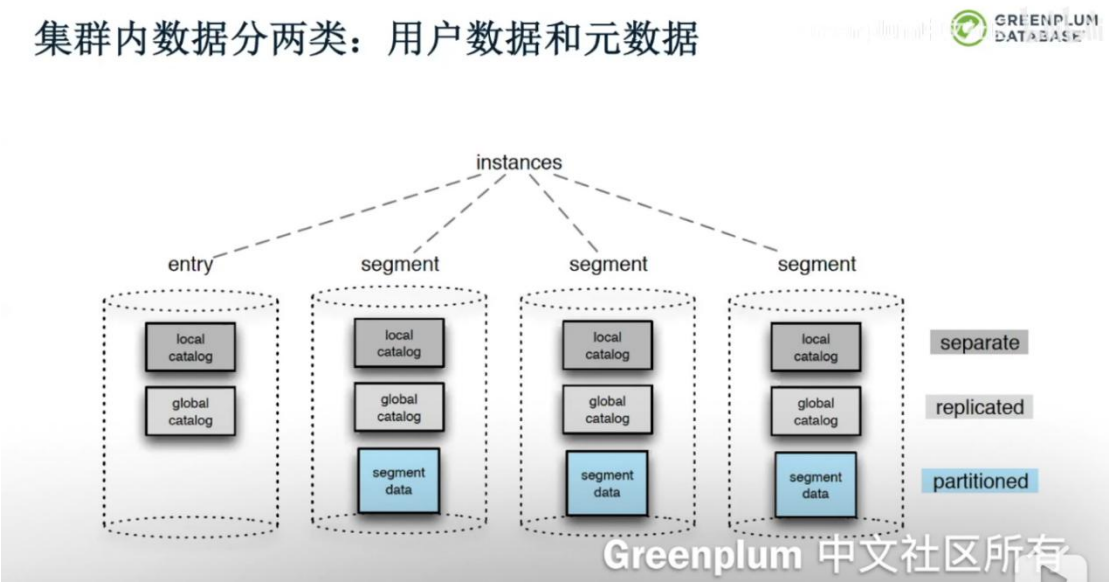
1.2 MPP 无共享静态拓扑



master 与 standby master 可以实现集群的高可用,通过共享高速的网络传送数据，除了网络是共享的其他的都是无共享的

1.3 集群内数据分两类

集群内数据分两类:用户数据与元数据

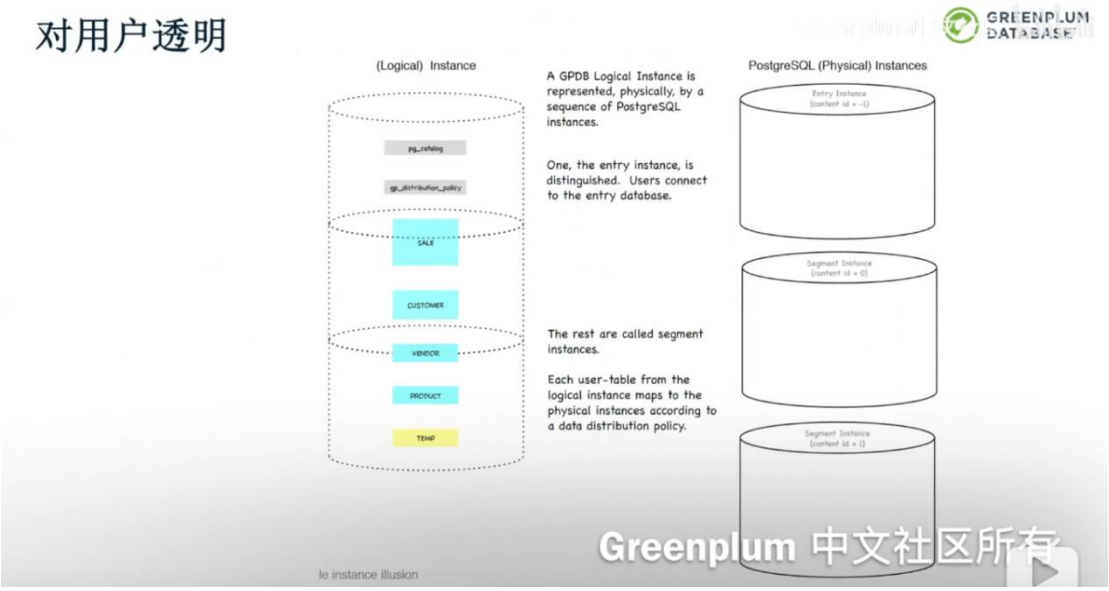


global 日志在所有的节点上都是一样的

local 日志一般都是一些统计信息等系统表
segment data 保存的用户数据信息

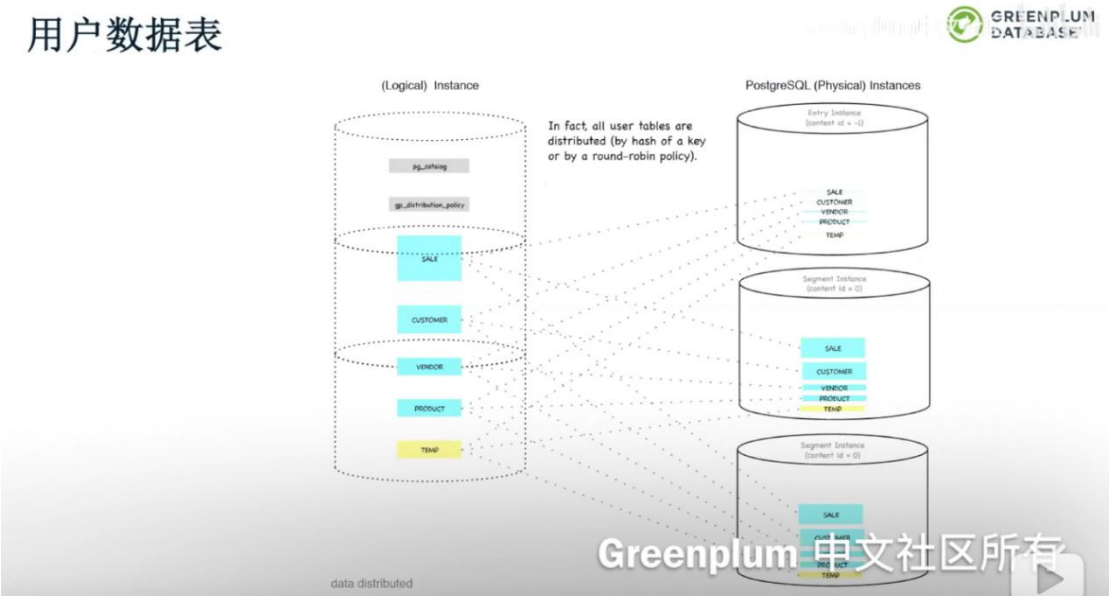
1.4 对用户透明

对用户可以看到以下的 instance



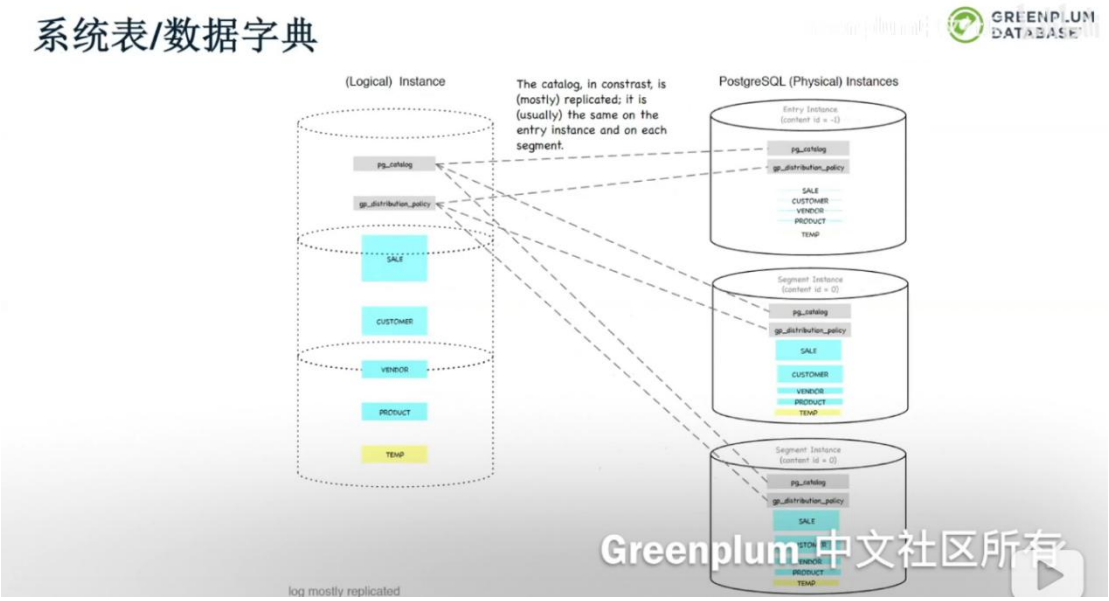
1.5 用户数据表

在以下的图标中可以看出用户数据已经打散到每个节点上,每个节点上有一部分，master 有元信息



1.6 系统表/数据字典

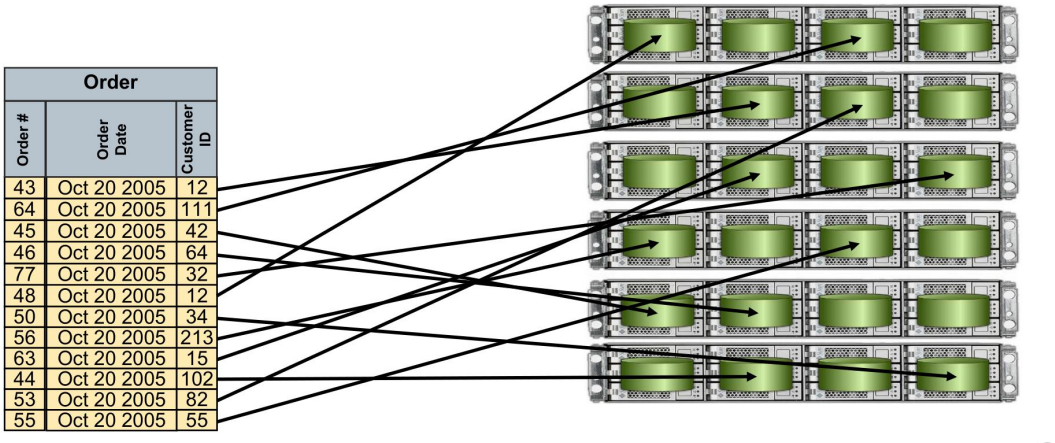
对于系统表/数据字典全部复制到每个节点上



1.7 数据分布:并行化处理的根基

集群按照算法均匀的把数据分不到不同的分区中，便于查快的查询

最重要的策略和目标是均匀分布。



Order		
Order #	Order Date	Customer ID
43	Oct 20 2005	12
64	Oct 20 2005	111
45	Oct 20 2005	42
46	Oct 20 2005	64
77	Oct 20 2005	32
48	Oct 20 2005	12
50	Oct 20 2005	34
56	Oct 20 2005	213
63	Oct 20 2005	15
44	Oct 20 2005	102
53	Oct 20 2005	82
55	Oct 20 2005	55

1.8 多态储存:根据数据温度选择最佳的储存方式

一般的数据都是有热度的，一般的越新的数据价值越高，越老的数据价值越低

1.8.1 行储存

1、访问多列时速度快
2、支持高效更新和删除
3、AO 行储存主要为插入而优化

1.8.2 列储存

1、列储存更适合压缩
2、查询列子集时速度快
3、不同列可以使用不同的压缩方式:gzip(1-9),quicklz,delta,RLE,zstd

1.8.3 外部表

1、历史数据和不常访问的数据储存在 HDFS 或者其他外部系统中
2、无缝查询所有数据
3、Text,CSV,Bianry,Avro,Parquest,ORC 格式

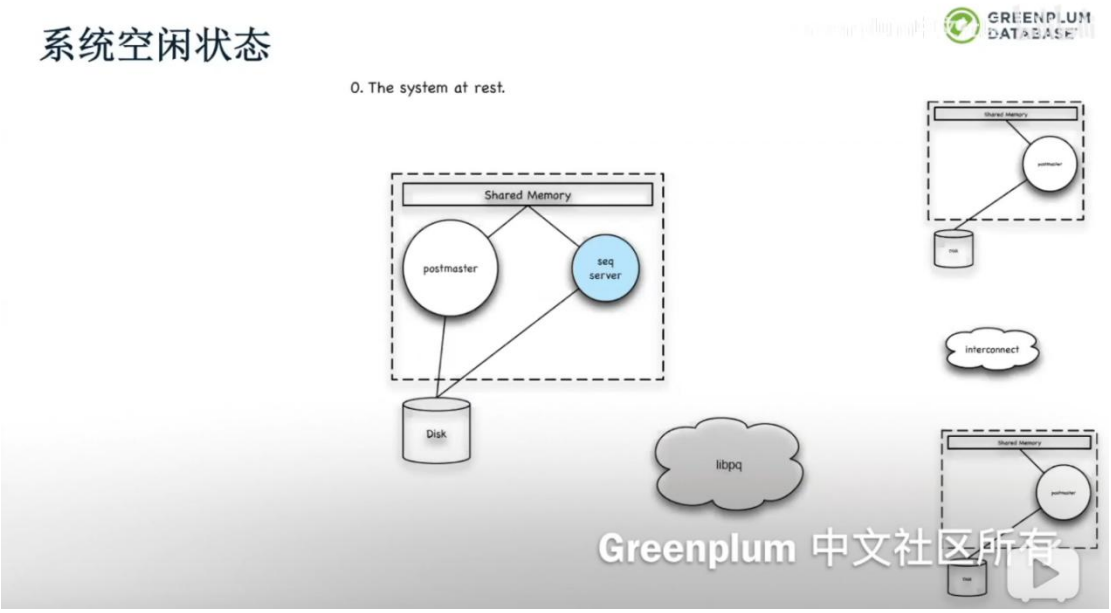
2 Greenplum SQL 的执行过程

0、The system at rest
1、Client connects via the entry postmaster
2、Entry postmaster forks a new backend -- the QD
3、QD connects to segment via the segment postmasters
4、Segment postmasters fork initial gang of QEs

5、Client submits a query to the QD
6、QD plans query and submits plans to QEs
7、QD and QEs setup interconnect routes according to plan
8、QD and QEs execute their slices sending tuples up the slice tree
9、QEs return status to QD
10、QD returns result set and status to the client

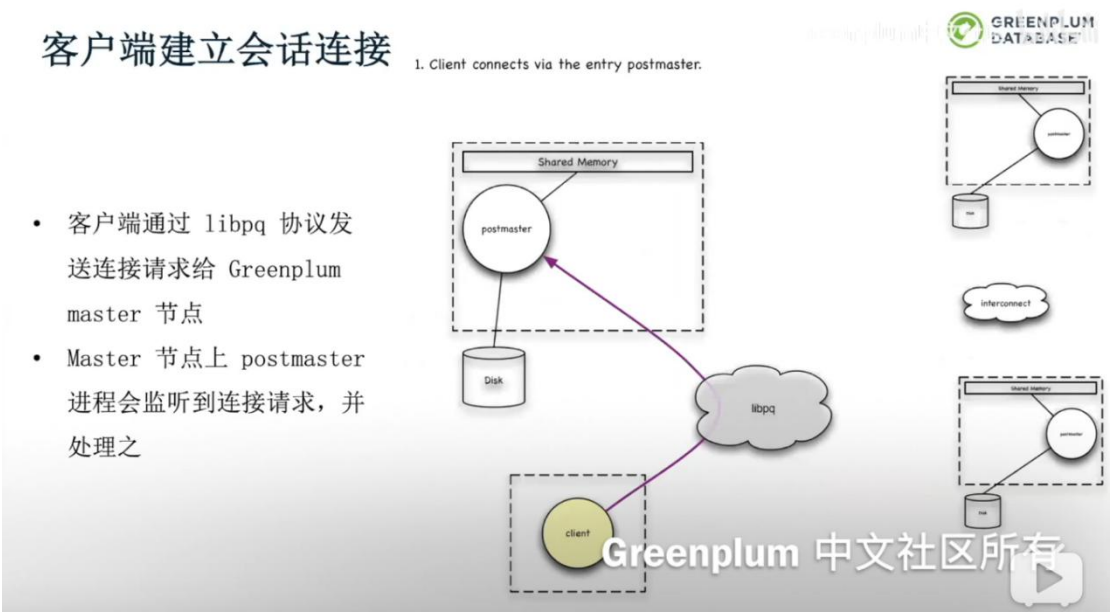
2.1 系统空闲状态

1、1 个 master，2 个 segment
11、postmaster 是数据库主进程，监听用户的请求
12、此时系统空闲，没有任何运行查询
13、Master 上的 seq server 为序列号生成器



2.2 客户端建立会话链接

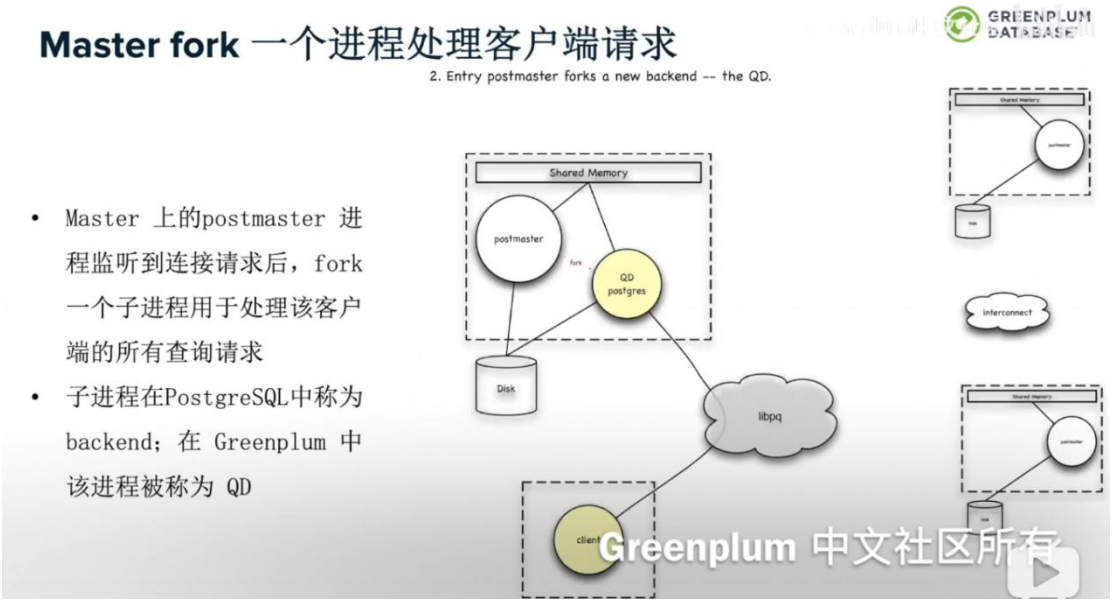
1、客户端通过 libpq 协议发送链接请求给 Greenplum master 节点
2、Master 节点上 postmaster 进程会监听到链接请求，并处理



2.3 Master fork 一个进程处理客户端请求

1、Master 上的 postmaster 进程监听到链接请求后，fork 一个子进程用于处理该客户端的所有查询请求

2、子进程在 PostgrsSQL 中称为 backend;在 Greenplum 中该进程称为 QD

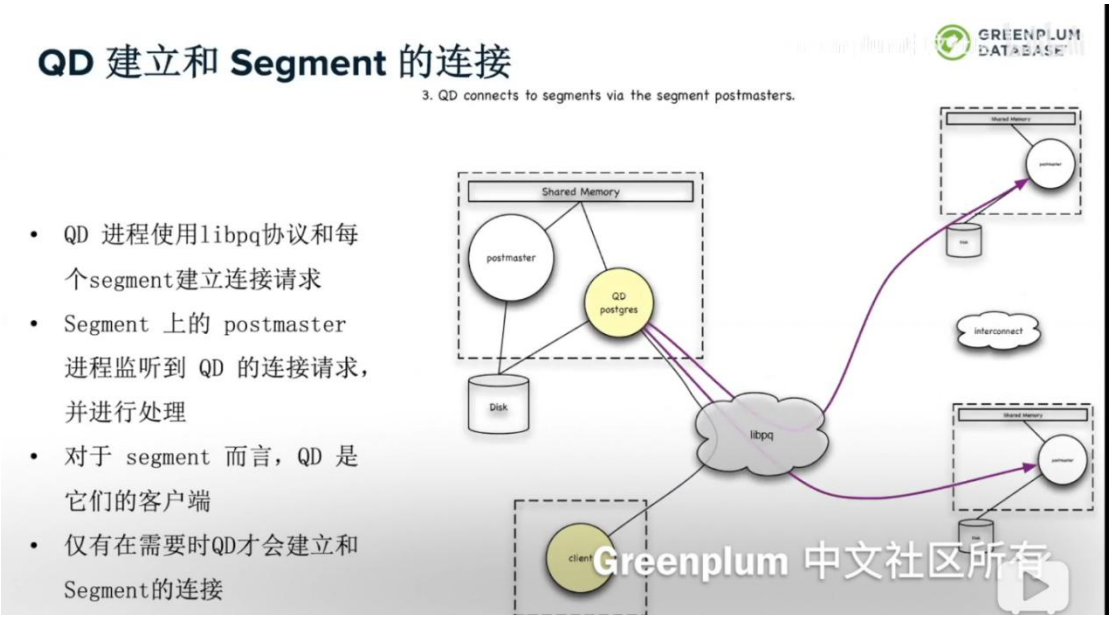


2.4 QD 建立和 Segment 的链接

1、QD 进程使用 libpq 协议和每个 segment 建立链接请求

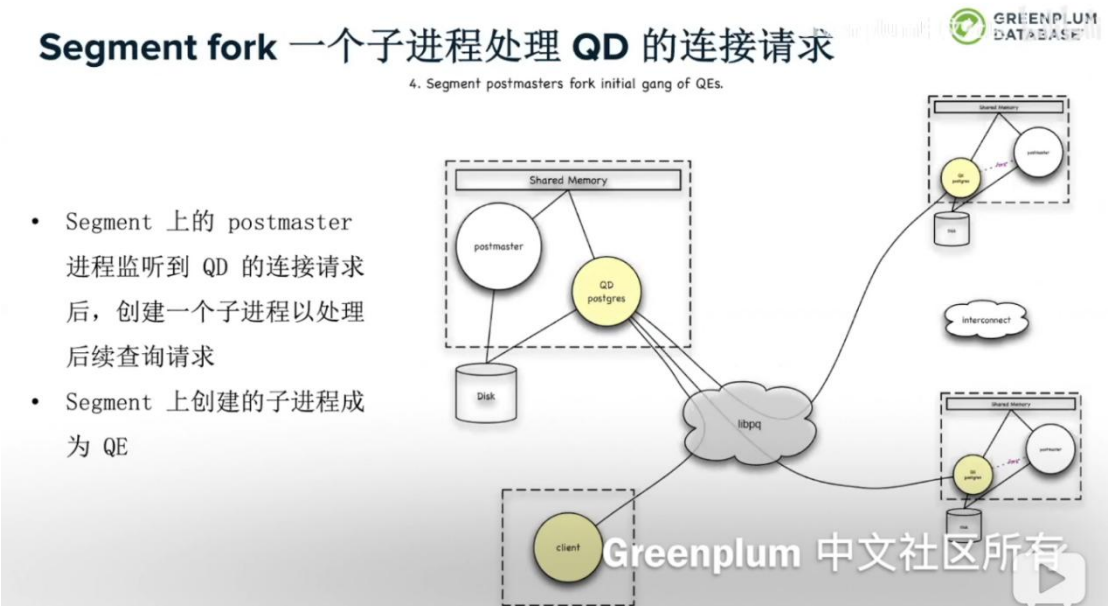
2、Segment 上的 postmaster 进程监听到 QD 的链接请求并进行处理

3、对于 segment 而言，QD 是他们的客户端
4、仅有在需要时 QD 才会建立和 Segment 的链接



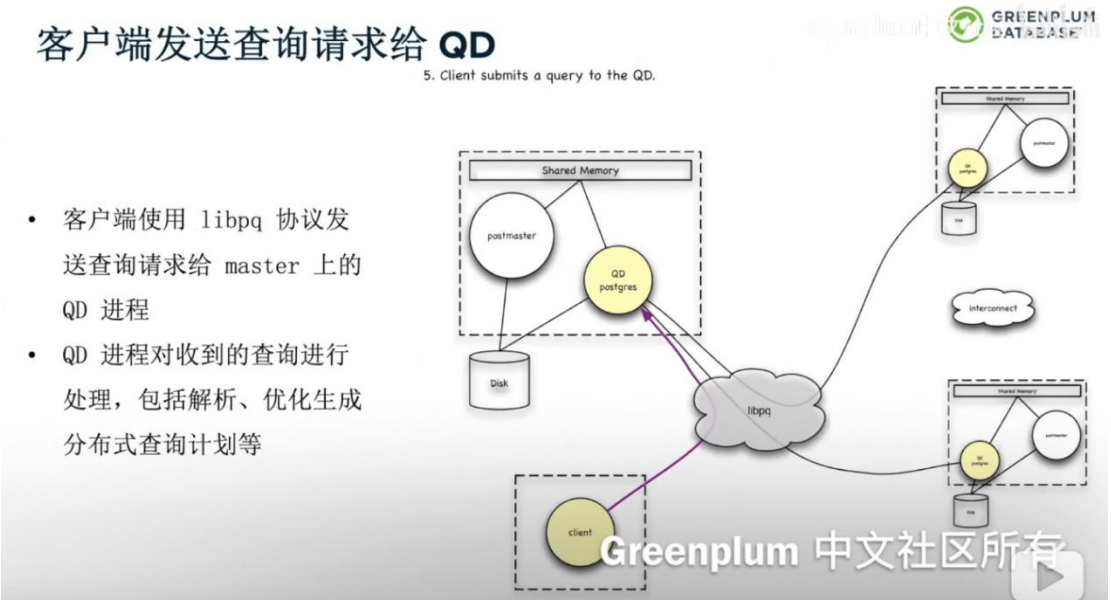
2.5 segment fork 一个子进程处理 QD 的链接请求

1、Segment 上的 postmaster 进程监听到 QD 的链接请求后，创建一个子进程以处理后续查询的请求
2、Segment 上创建的子进程称为 QE



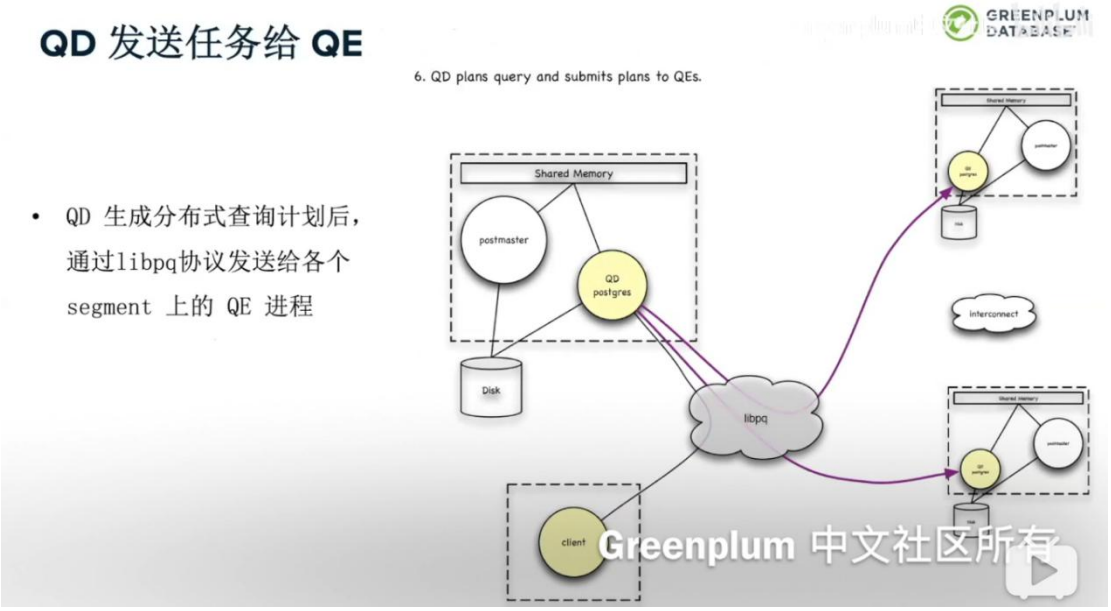
2.6 客户端发送查询请求给 QD

1、客户端使用 libpq 协议发送查询请求给 master 上得得 QD 进程
2、QD 进程对接收到的查询进行处理,包括解析，优化生成分布式查询计划等



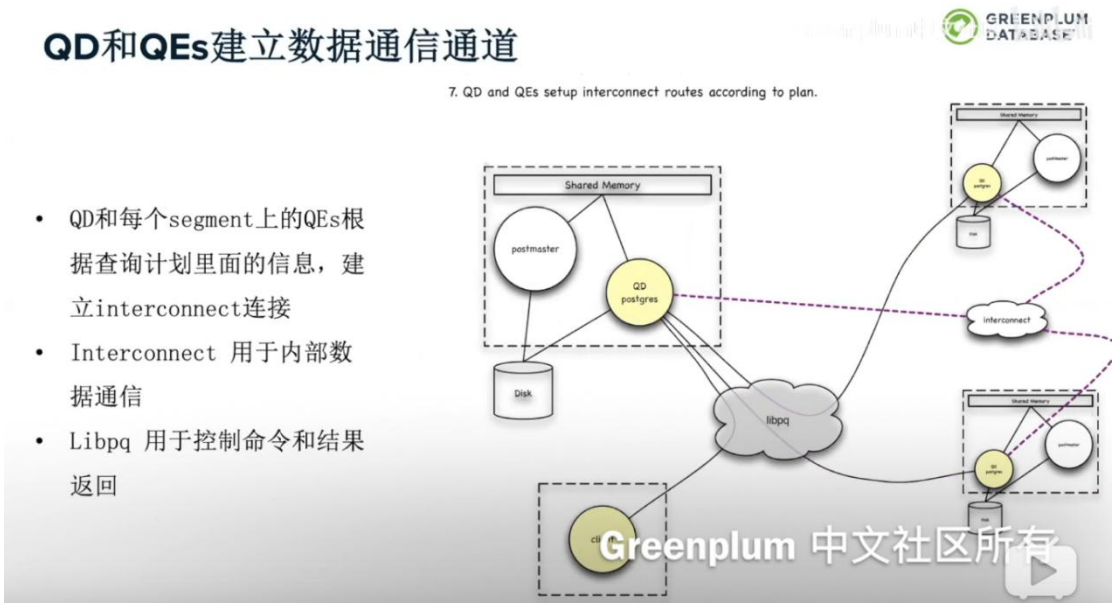
2.7 QD 发送任务给 QE

1、QD 生成分布式查询计划后通过 libpq 协议发送给各个 segment 上的 QE 进程



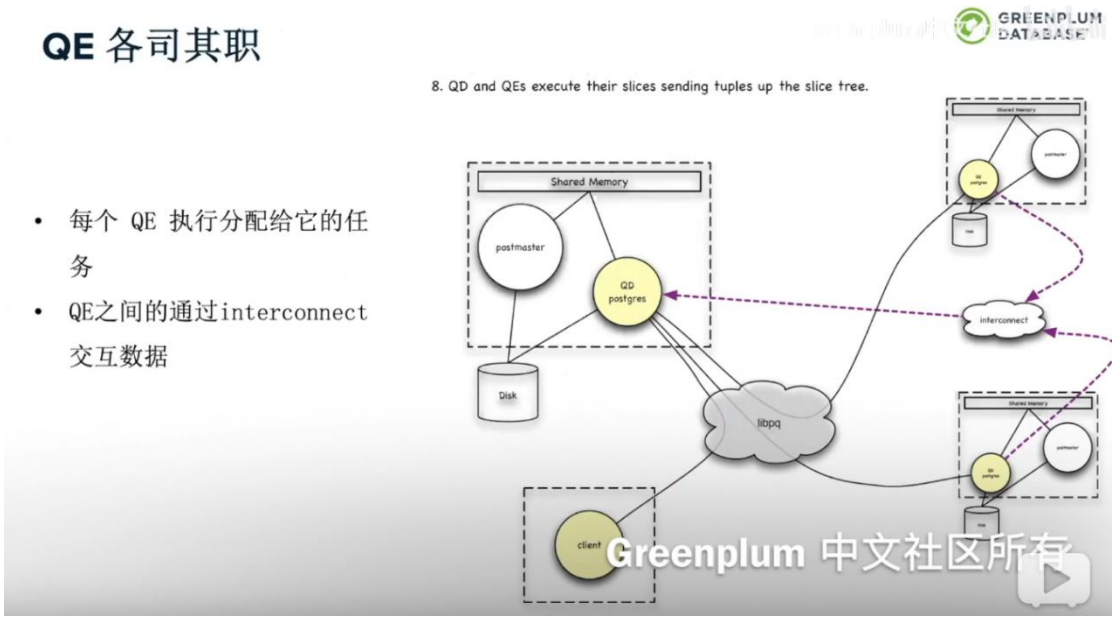
2.8 QD 与 QEs 建立数据通信通道

1、QD 和每个 segment 上的 QEs 根据查询计划里面的信息，简历 interconnect 链接
2、Interconnect 用于内部数据通信
3、Libpq 用于控制命令和结果返回



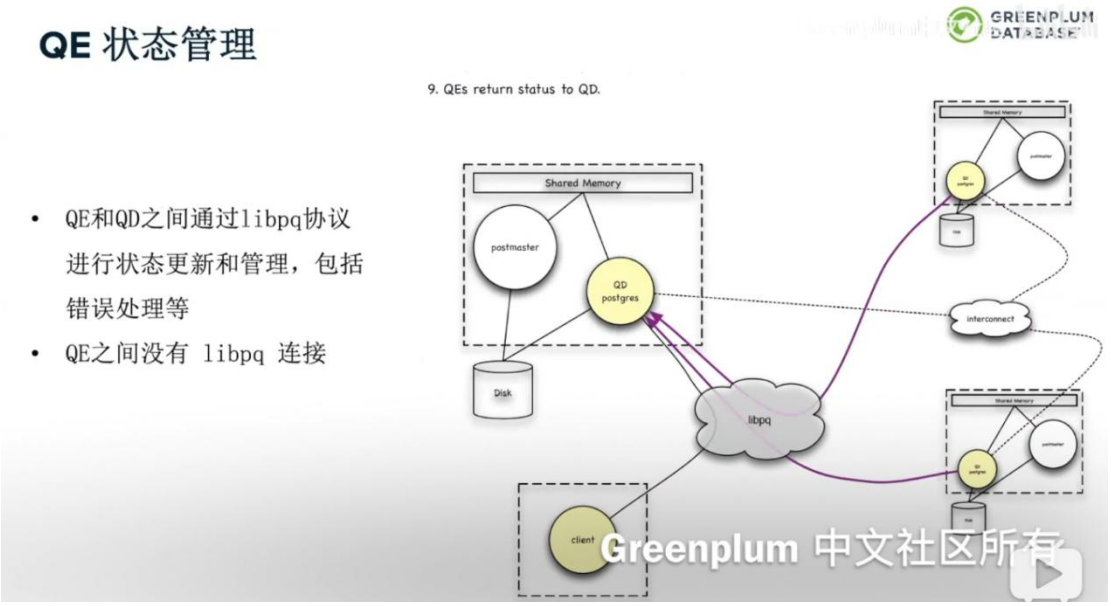
2.9 QE 各司其职

1、每个 QE 执行分配给它的任务
2、QE 之间的通过 interconnect 交互数据



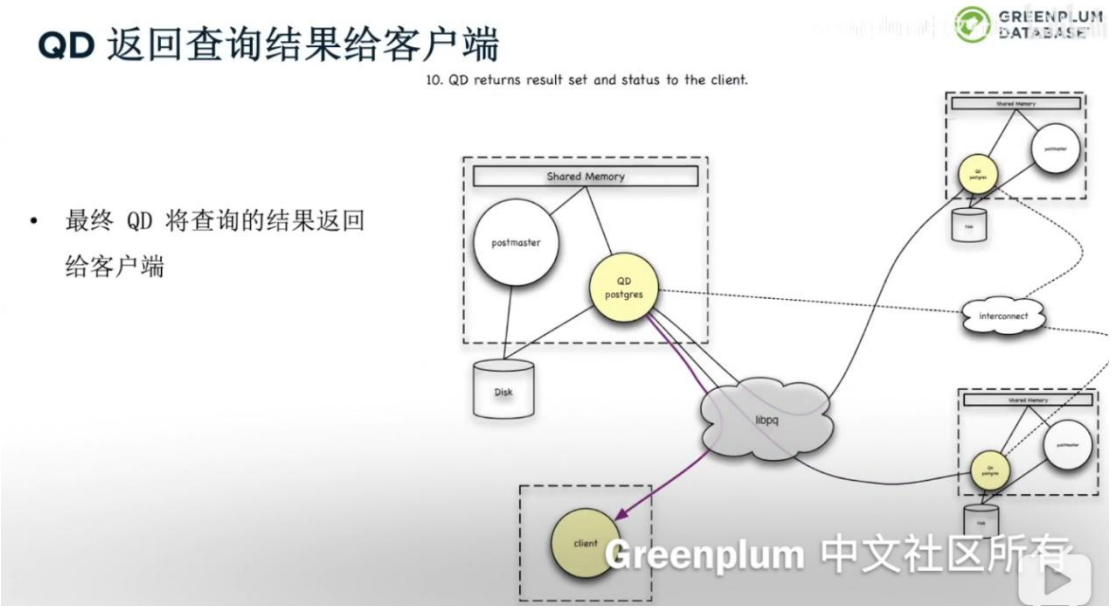
2.10 QE 状态管理

1、QE 和 QD 之间通过 libpq 协议进行状态的跟新和管理，包括错误处理等
2、QE 之间没有 libpq 链接



2.11 QD 返回查询结果给客户端

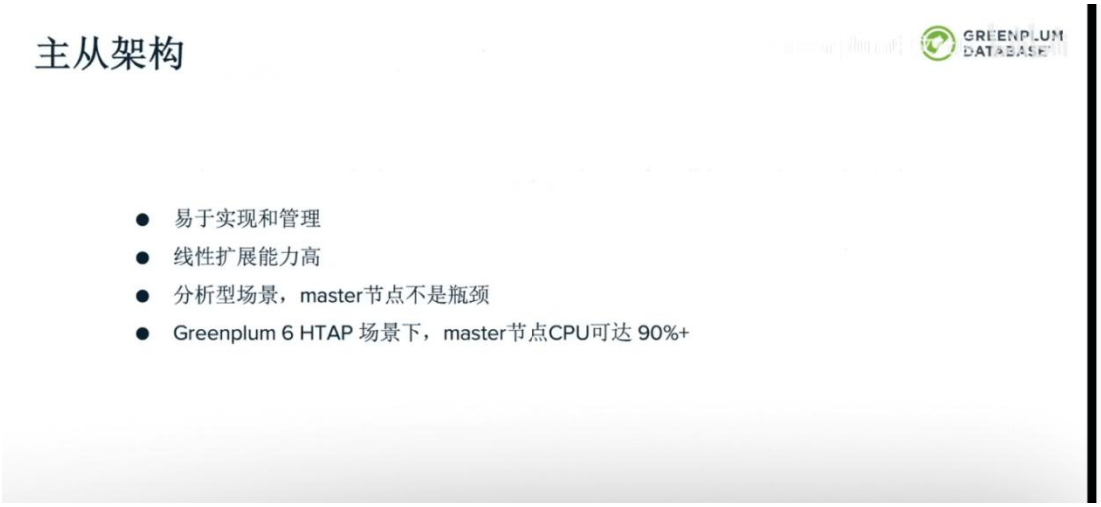
1、最终 QD 将查询的结果返回给客户端



3 Greenplum 主要设计思考

3.1 继承自 PostgreSQL 的设计

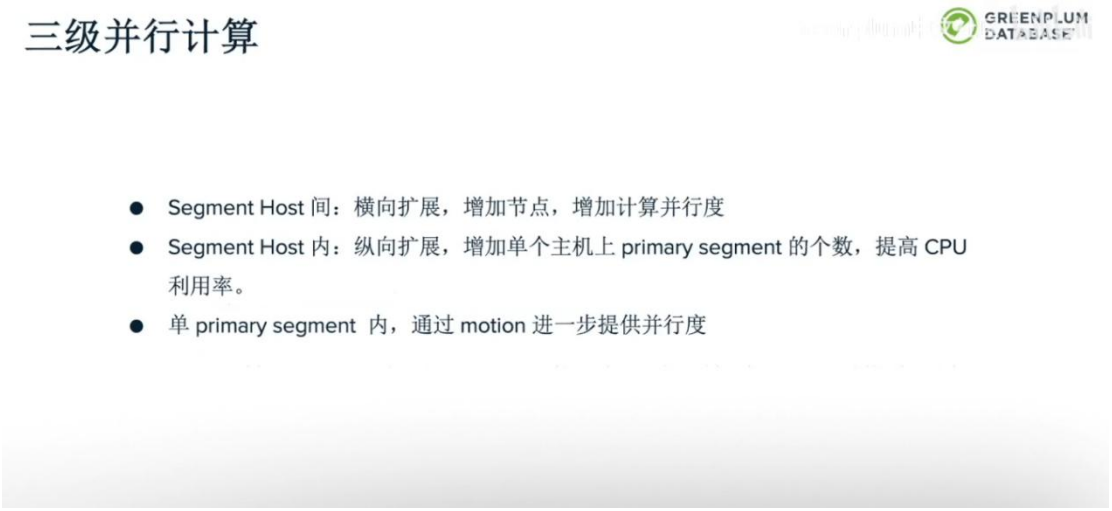
3.2 主从架构



3.3 数据储存

3.4 数据通信

3.5 三级并行计算



3.6 流水线执行

3.7 网络

网络

GREENPLUM DATABASE

- TCP：状态信息和控制信息，基于PostgreSQL标准的 libpq 协议
- UDP：数据，自研的 interconnect 协议，效率高，扩展性好

3.8 磁盘

磁盘

GREENPLUM DATABASE

- 多块磁盘（12-24块很常见）提高IO吞吐
- RAID 5，RAID 10
- 2个副本 vs. 3个副本