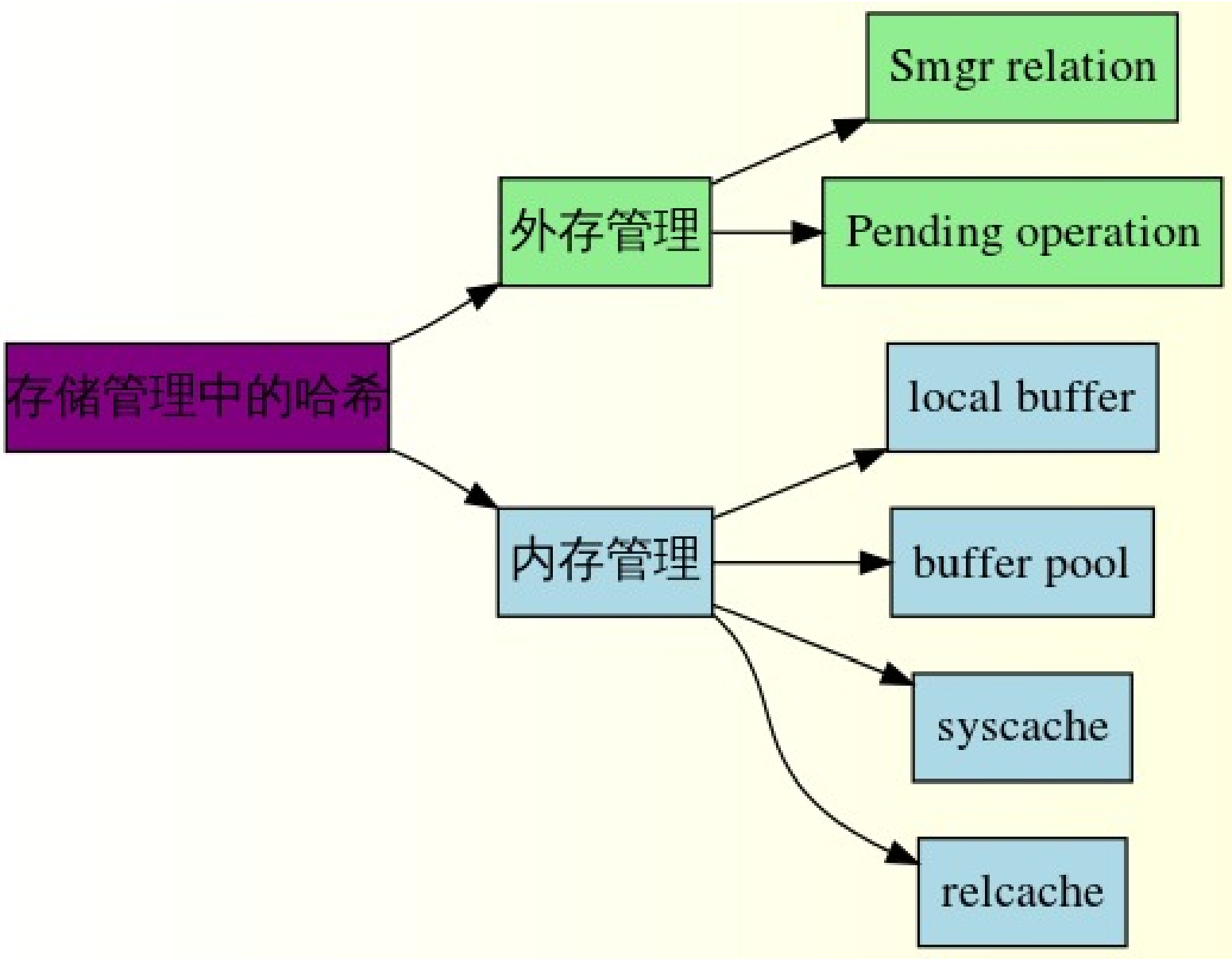
Postgresql Hash

马文韬

WHU

January 18, 2014

存储管理中的哈希


存储管理中的哈希
外存管理
内存管理
Smgr relation
Pending operation
local buffer
buffer pool
syscache
relcache

Hash table基本结构图

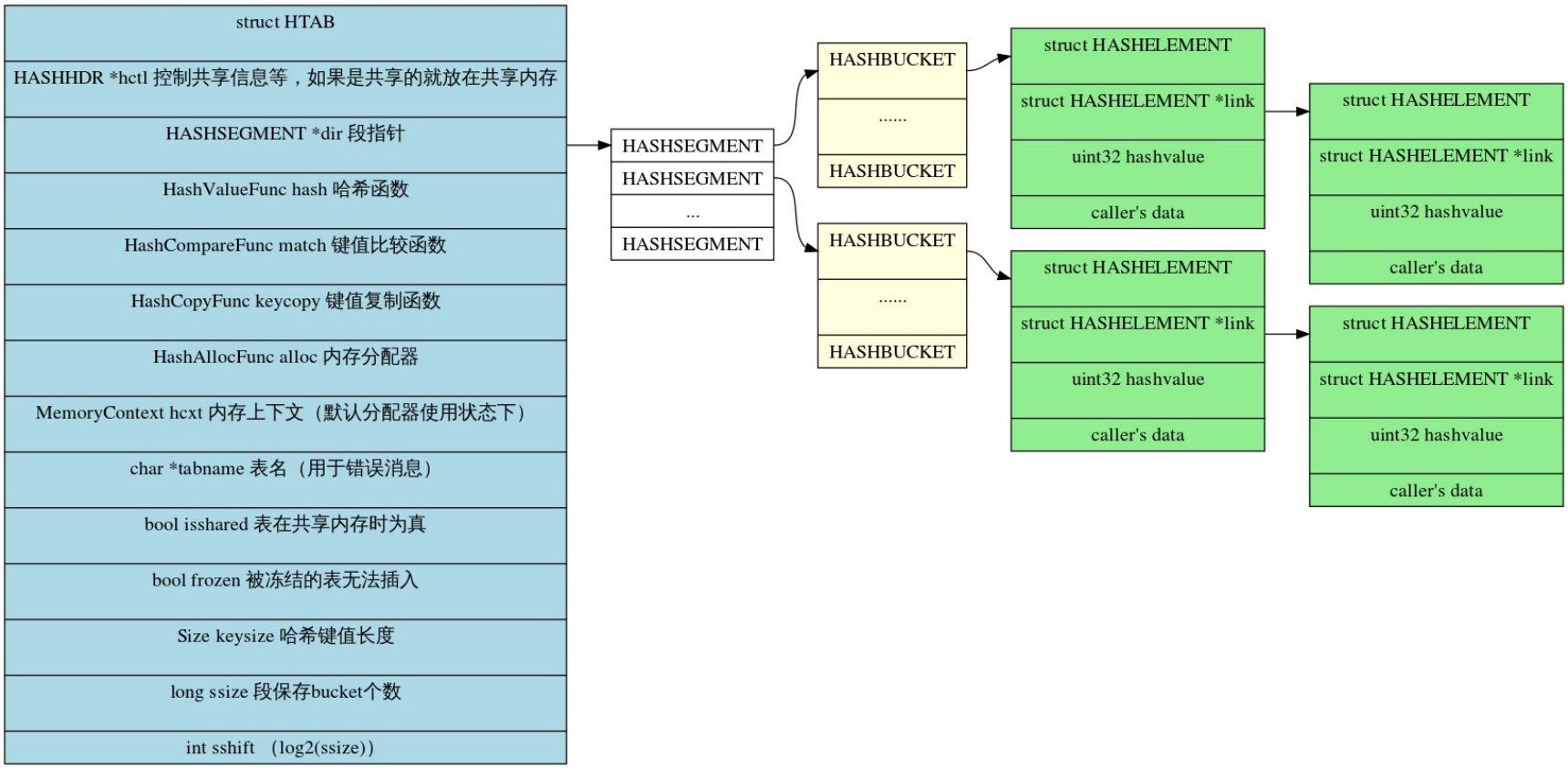


Figure: hash table 结构图

Hash table基本结构及函数

Table: 哈希结构体中函数指针的默认函数及其功能

函数指针	相关函数
HashValueFunc hash	string_hash 计算string哈希值 tag_hash 计算tag哈希值 oid_hash 计算oid的哈希值 bitmap_hash计算bitmap哈希
HashCompareFunc match	string_compare，如果不自定义，默认 bitmap_match 和bitmap_hash一起用
HashCopyFunc keycopy	strlcpy 可以自定义，默认
HashAllocFunc alloc	DynaHashAlloc 调用上下文 中定义的内存分配方法

哈希表相关函数及其操作

```
void *hash_search(HTAB *hashp, void *keyPtr, HASHACTION action, bool *foundPtr);
```

Table: hash_search相关操作action标记

HASH_FIND	根据key 查找哈希表
HASH_ENTER	查找哈希表， 如果entry没出现， 那么创建一个
HASH_ENTER_NULL	查找哈希表， 如果超出内存， 返回NULL
HASH_REMOVE	移除有特定key的entry

smgr relation中的哈希

RelFileNode

- Oid spcNode; 表空间
- Oid dbNode ; 数据库
- Oid relNode; 关系

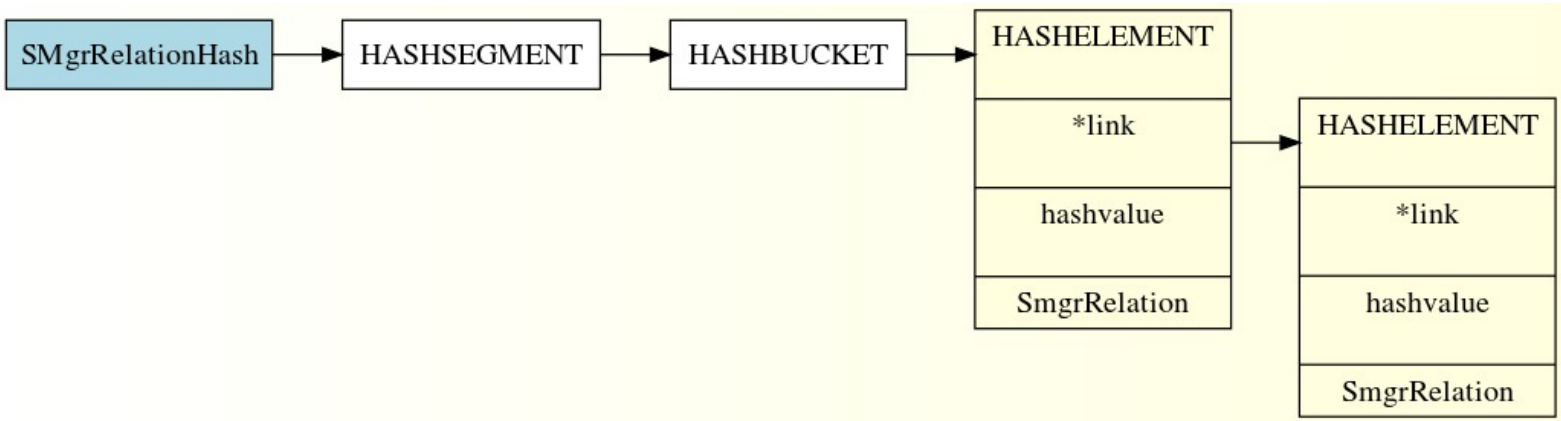


Figure: smgr中的哈希

smgr relation中的hash

Table: smgr relation中的哈希函数

函数名	参数	调用函数	使用场景
hash_search	SMgrRelationHash, (void *)&rnode, HASH_ENTER，&found	SMgrRelation smgropen(RelFileNode rnode)	RelFileNode作为主键 查找并返回SMgrRelation对象， 找不到就创建一个
hash_search	SMgrRelationHash, &(reln->smgr_rnode), HASH_REMOVE,NULL	void smgrclose(SMgrRelation reln)	smgr关系的RelFileNode 作为主键查找并删除 SMgrRelation对象
hash_search	SMgrRelationHash, (void *)&rnode HASH_FIND,NULL	void smgrclosenode(RelFileNode rnode)	RelFileNode作为主键查找返回 相应SMgrRelation对象， 由smgrclose完成删除 避免创建无用关系

Pending operation中的哈希

PendingOperationEntry

- PendingOperationTag;
- bool canceled;
- CycleCtr cycle_ctr;

Key:PendingOperationTag

- RelFileNode rnode;
- ForkNumber forkNum;
- BlockNumber segno;

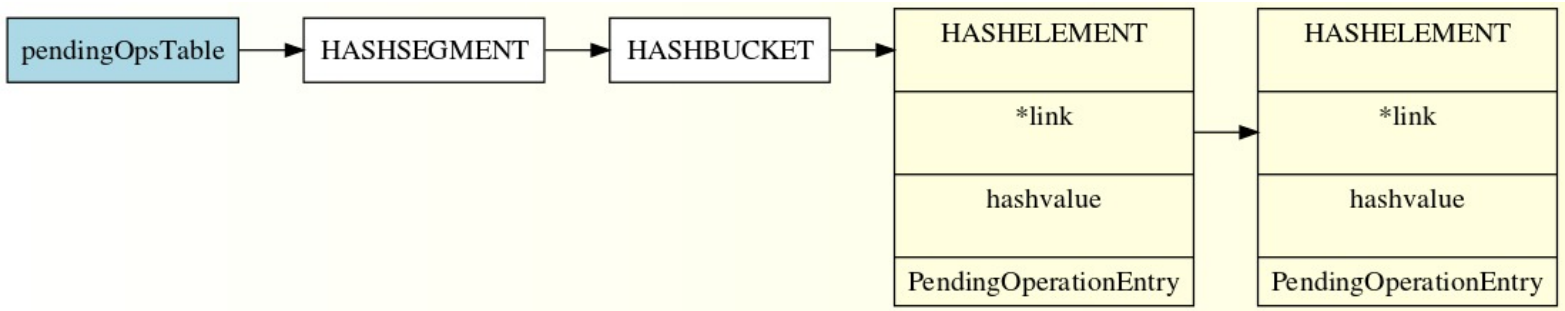


Figure: pending operation中的哈希

pending operation中的hash

Table: pending operation中的哈希函数

函数名	参数	调用函数	使用场景
hash_search	pendingOpsTable, &entry->tag, HASH_REMOVE, NULL	void mdsync(void)	写操作同步到磁盘时 删除已经无效的操作 ,PendingOperationTag 作为键查找相应操作并移除
hash_search	pendingOpsTable, &key,HASH_ENTER, &found	void RememberFsyncRequest (RelFileNode rnode, ForkNumber forknum, BlockNumber segno)	PendingOperationTag作为 主键将fsync的请求 添加到哈希表中

SysCache 中的Hash

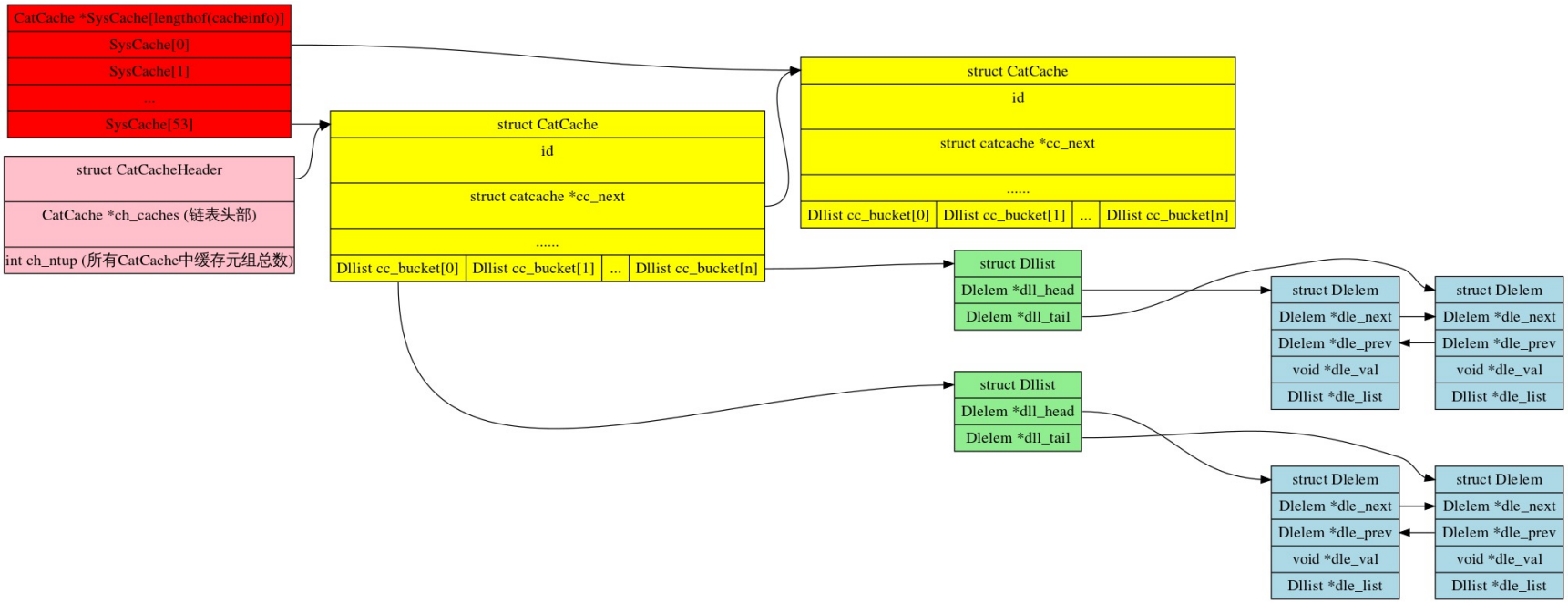


Figure: hash桶结构图

SysCache中的Hash

Table: 精确匹配基本流程及相关函数

初始化关键字信息	根据四个关键字初始化cur_skey
计算哈希值和索引	CatalogCacheComputeHashValue 由关键字计算hashValue HASH_INDEX宏利用hashValue 和cc_buckets计算索引
在索引对应的桶中查找	HeapKeyTest检查key是否匹配 将匹配的元组移动到链表头

Table: 部分匹配基本流程及相关函数

初始化关键字信息	根据部分关键字初始化cur_skey
计算哈希值和索引	CatalogCacheComputeHashValue 根据关键字个数计算lhashValue
在CatCache的cc_lists 指向的CatCList链表中查找	HeapKey Test检查key是否匹配 将匹配的CatClist放到cc_lists 链表的头部 不存在CatCList，扫描物理表并构建

RelCache中的哈希

Key

Relation 的Oid作为key进行查询。

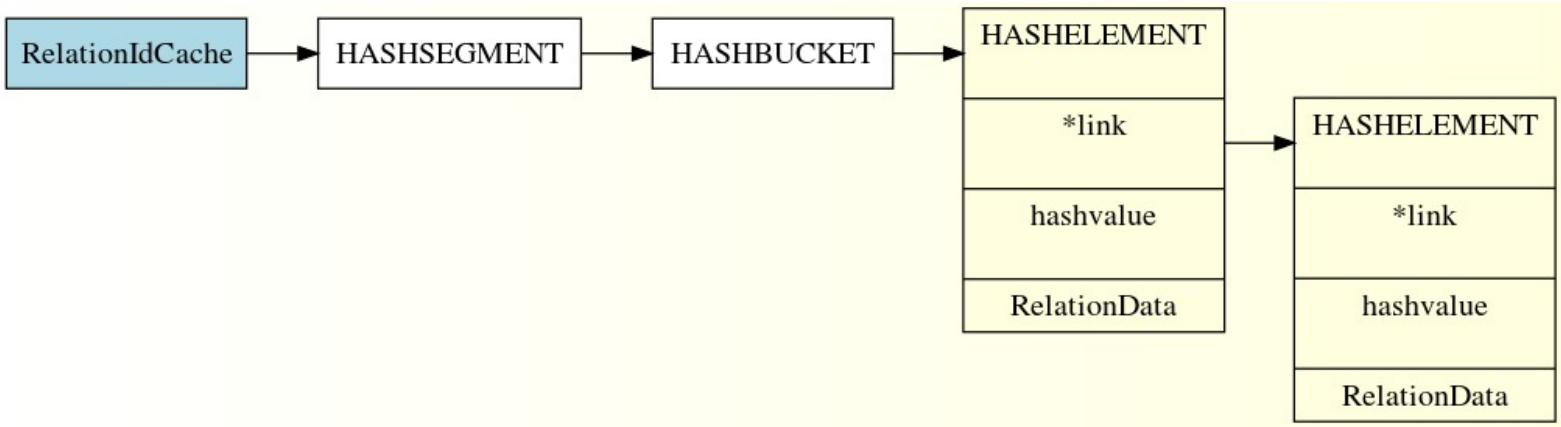


Figure: relcache中的哈希

RelCache中的hash

Table: RelCache中的hash

函数名	参数	调用宏	使用场景
hash_search	RelationIdCache, &(RELATION->rd_id), HASH_ENTER, &found	RelationCacheInsert (RELATION)	以关系的Oid作为主键将新的关系插入到relcache哈希表中
hash_search	RelationIdCache, &(ID),HASH_FIND, NULL	RelationIdCacheLookup (ID,RELATION)	用关系Oid作为主键在relcache中查找相应对象
hash_search	RelationIdCache, &(RELATION->rd_id), HASH_REMOVE, NULL	RelationCacheDelete (RELATION)	用关系Oid作为主键查找并删除relcache相应对象

Buffer pool中的哈希

BufferTag

- rnode(表空间OID，数据库OID和表OID组成);
- forkNum 枚举类型，标记缓冲区中文件块类型;
- blockNum 块号;

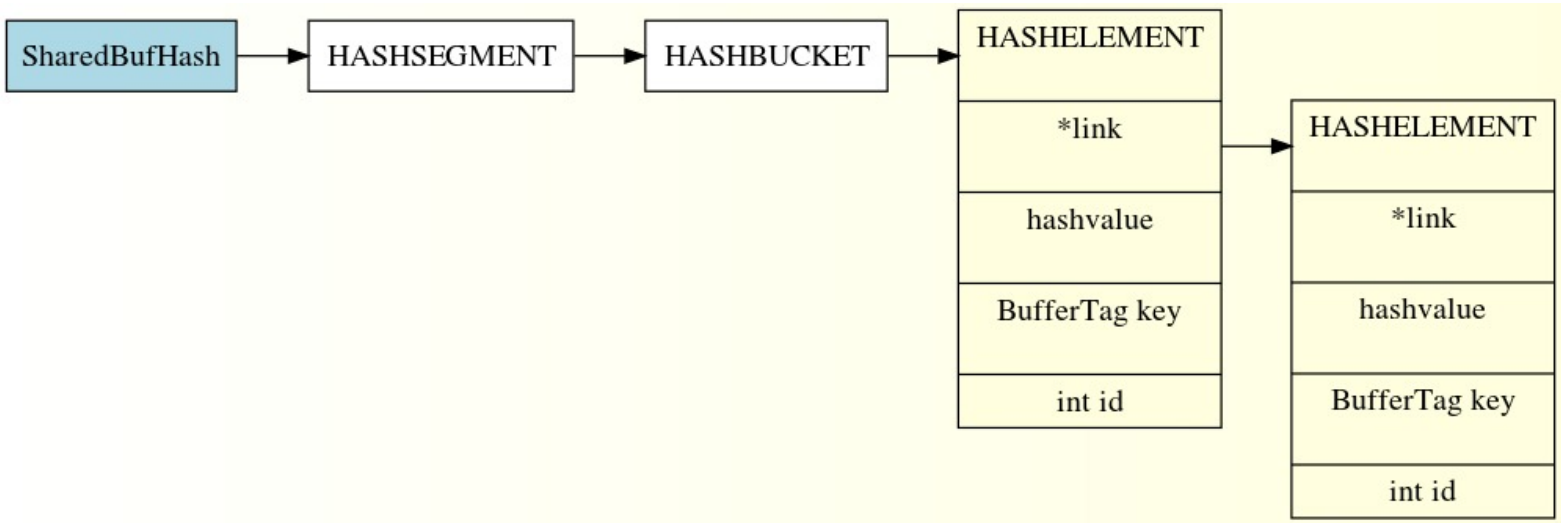


Figure:

buffer pool 中的hash

Table: buffer pool中的hash

函数名	参数	调用函数	使用场景
hash_search_with_hash_value	SharedBufHash, tagPtr, hashcode, HASH_FIND, NULL	int BufTableLookup (BufferTag *tagPtr, uint32 hashcode)	根据BufferTag在ShareBufHash中查询, 返回buffer ID
hash_search_with_hash_value	SharedBufHash, tagPtr, hashcode, HASH_REMOVE, NULL	void BufTableDelete (BufferTag *tagPtr,uint32 hashcode)	根据BufferTag删除ShareBufHash中的entry
hash_search_with_hash_value	SharedBufHash, tagPtr, hashcode, HASH_ENTER, &found	int BufTableInsert(BufferTag *tagPtr, uint32 hashcode, int buf_id	根据BufferTag和buffer ID插入entry, 如果有冲突entry, 返回冲突entry的buffer ID

local buffer中的哈希

Key

和buffer pool一样，使用BufferTag作为键进行查找。

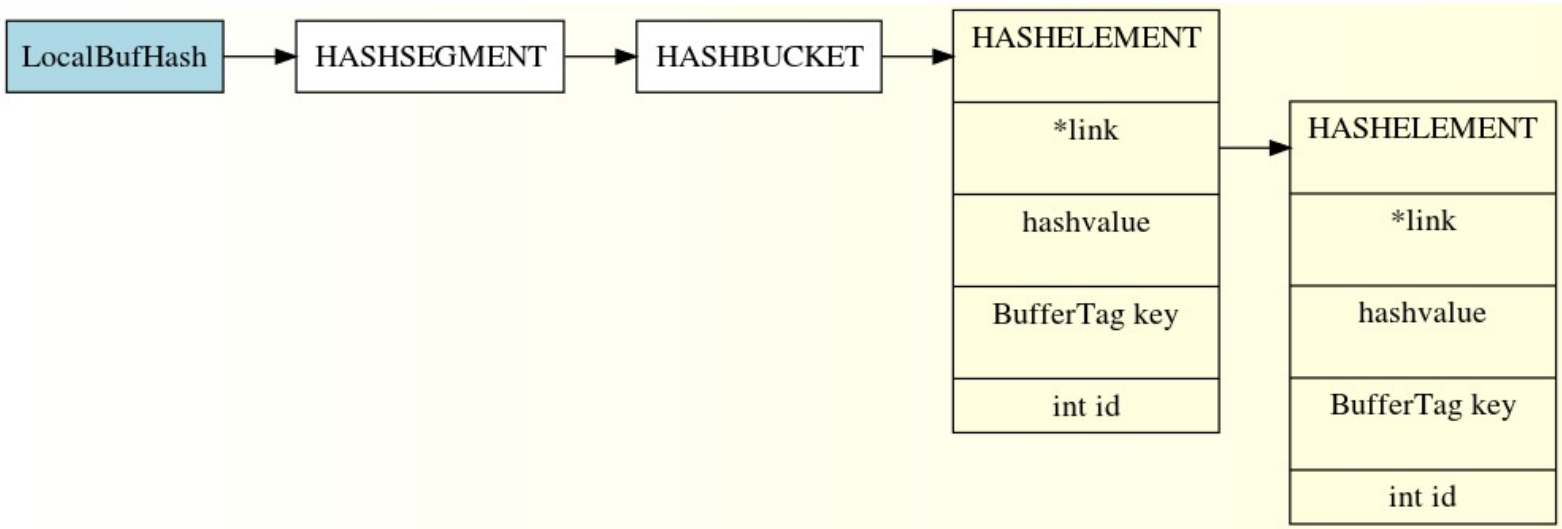


Figure: local buffer中的哈希

local buffer hash

Table: local buffer中的哈希函数

函数名	参数	调用函数	使用场景
hash_search	LocalBufHash,&newTag, HASH_FIND,NULL	void LocalPrefetchBuffer (SMgrRelation smgr, ForkNumber forkNum, BlockNumber blockNum)	异步读取一个关系块 用smgr等参数创建一个tag, 查找LocalBufHash中相应块
hash_search	LocalBufHash,&newTag, HASH_FIND,NULL	void LocalBufferAlloc (SMgrRelation smgr, ForkNumber forkNum, BlockNumber blockNum, bool *foundPtr)	用smgr等参数创建一个tag, 为给定关系的给定页面创建 local buffer
hash_search	LocalBufHash, &bufHdr->tag, HASH_REMOVE,NULL	void LocalBufferAlloc (SMgrRelation smgr, ForkNumber forkNum, BlockNumber blockNum, bool *foundPtr)	更新LocalBufHash, 移除旧的entry
hash_search	LocalBufHash, &bufHdr->tag, HASH_ENTER,NULL	void LocalBufferAlloc (SMgrRelation smgr, ForkNumber forkNum, BlockNumber blockNum, bool *foundPtr)	更新LocalBufHash, 创建新的entry