Determination of the Atomic Weight of Magnesium CHEM 101

John Smith

January 20, 2014

Date Performed:	January 1, 2012
Partners:	James Smith
	Mary Smith
Instructor:	Professor Smith

1 Objective

To determine the atomic weight of magnesium via its reaction with oxygen and to study the stoichiometry of the reaction (as defined in 1.1):

$$2\,\mathrm{Mg} + \mathrm{O_2} \longrightarrow 2\,\mathrm{MgO}$$

1.1 Definitions

Stoichiometry The relationship between the relative quantities of substances taking part in a reaction or forming a compound, typically a ratio of whole integers.

Atomic mass The mass of an atom of a chemical element expressed in atomic mass units. It is approximately equivalent to the number of protons and neutrons in the atom (the mass number) or to the average number allowing for the relative abundances of different isotopes.

2 Experimental Data

Mass of empty crucible	7.28 g
Mass of crucible and magnesium before heating	8.59 g
Mass of crucible and magnesium oxide after heating	9.46 g
Balance used	#4
Magnesium from sample bottle	#1

3 Sample Calculation

Mass of magnesium metal	= 8.59 g - 7.28 g
	= 1.31 g
Mass of magnesium oxide	= 9.46 g - 7.28 g
	= 2.18 g
Mass of oxygen	= 2.18 g - 1.31
	= 0.87 g

Because of this reaction, the required ratio is the atomic weight of magnesium: 16.00 g of oxygen as experimental mass of Mg: experimental mass of oxygen or $\frac{x}{1.31} = \frac{16}{0.87}$ from which, $M_{\mathrm{Mg}} = 16.00 \times \frac{1.31}{0.87} = 24.1 = 24\,\mathrm{g/mol}$ (to two significant figures).

4 Results and Conclusions

The atomic weight of magnesium is concluded to be 24 g/mol, as determined by the stoichiometry of its chemical combination with oxygen. This result is in agreement with the accepted value.

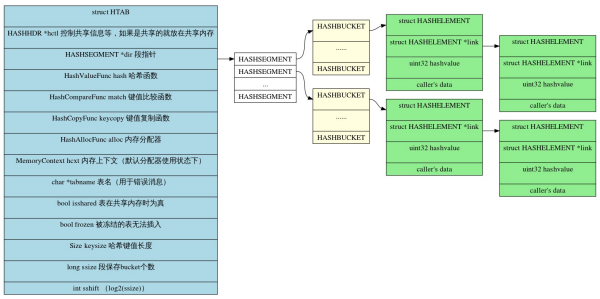

Figure 1: Figure caption.

5 Discussion of Experimental Uncertainty

The accepted value (periodic table) is 24.3 g/mol [**?**]. The percentage discrepancy between the accepted value and the result obtained here is 1.3%. Because only a single measurement was made, it is not possible to calculate an estimated standard deviation.

The most obvious source of experimental uncertainty is the limited precision of the balance. Other potential sources of experimental uncertainty are: the reaction might not be complete; if not enough time was allowed for total oxidation, less than complete oxidation of the magnesium might have, in part, reacted with nitrogen in the air (incorrect reaction); the magnesium oxide might have

absorbed water from the air, and thus weigh "too much." Because the result obtained is close to the accepted value it is possible that some of these experimental uncertainties have fortuitously cancelled one another.

6 Answers to Definitions

a. The *atomic weight of an element* is the relative weight of one of its atoms compared to C-12 with a weight of 12.0000000..., hydrogen with a weight of 1.008, to oxygen with a weight of 16.00. Atomic weight is also the average weight of all the atoms of that element as they occur in nature.

b. The *units of atomic weight* are two-fold, with an identical numerical value. They are g/mole of atoms (or just g/mol) or amu/atom.

c. *Percentage discrepancy* between an accepted (literature) value and an experimental value is $\frac{|\mathrm{experimentalresult}-\mathrm{acceptedresult}|}{\mathrm{acceptedresult}}$.

7 概述

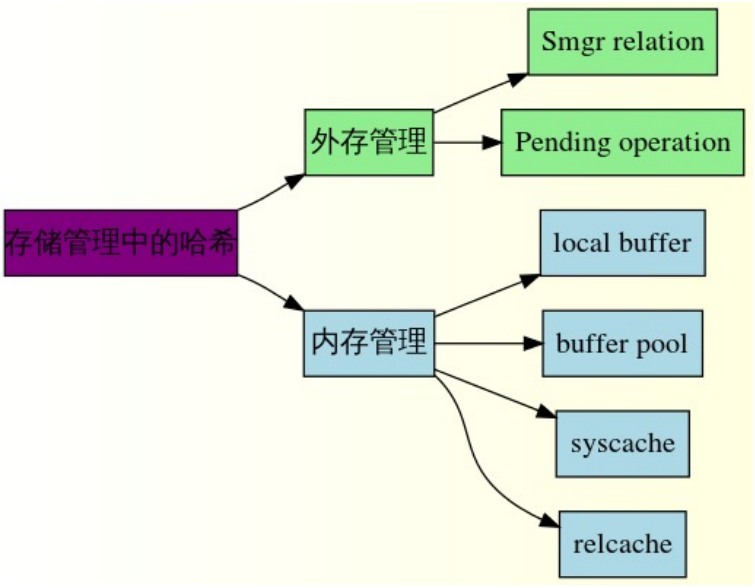

Figure 2: 存储管理中的哈希图

8 Hash table基本结构及其操作

asdfadsgsagasdg

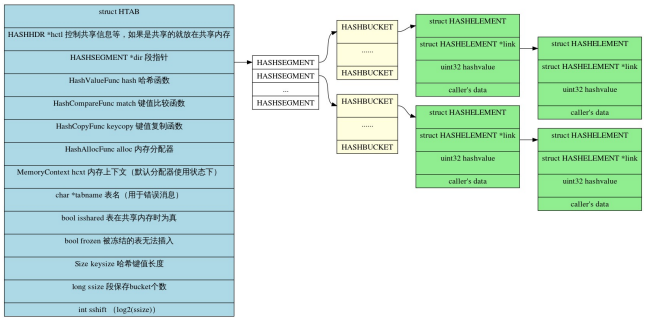

Figure 3: hash table 结构图

Table 1: 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

asdfasd

9 Smgr relation中的Hash

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

Table 2: 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

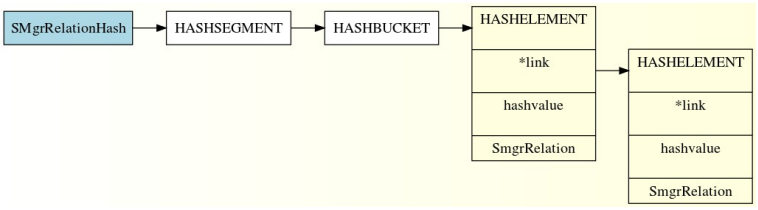

Figure 4: smgr中的哈希

Table 3: 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

10 Pending operation中的Hash

- PendingOperationTag;
- bool canceled;
- CycleCtr cycle_ctr;
- RelFileNode rnode;
- ForkNumber forkNum;
- BlockNumber segno;

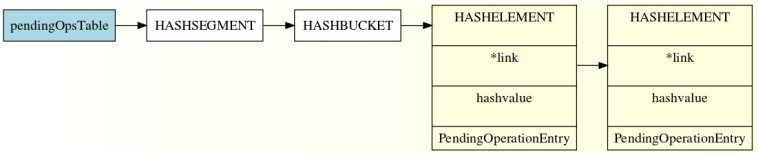

Figure 5: pending operation中的哈希

11 SysCache中的Hash

12 RelCache中的Hash

Relation 的Oid作为key进行查询。

13 Buffer pool中的Hash

- 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;

Table 4: 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

Table 5: 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 6: 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

Table 7: 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

Table 8: 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

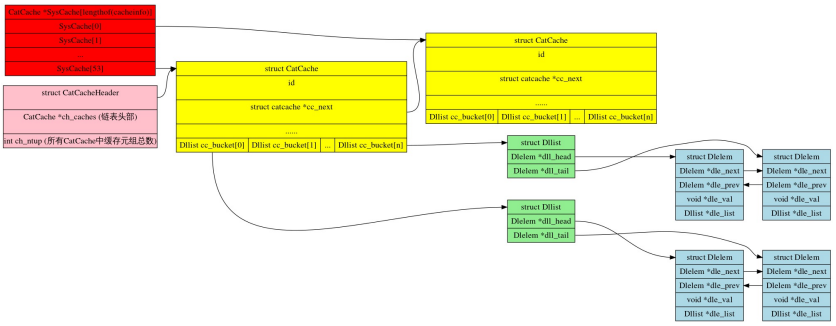

Figure 6: hash桶结构图

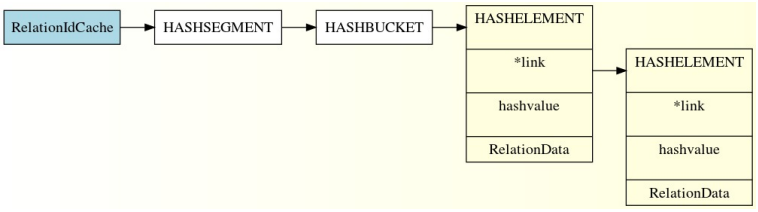

Figure 7: relcache中的哈希

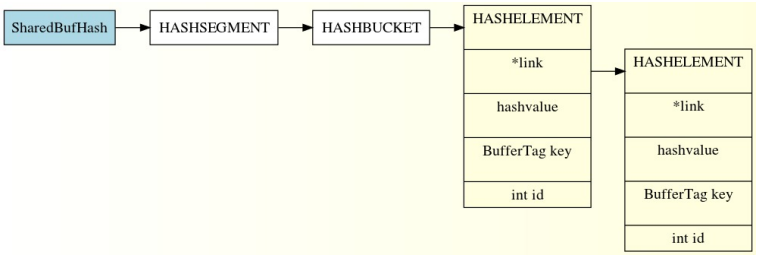

Figure 8:

14 Local buffer中的Hash

和buffer pool一样，使用BufferTag作为键进行查找。

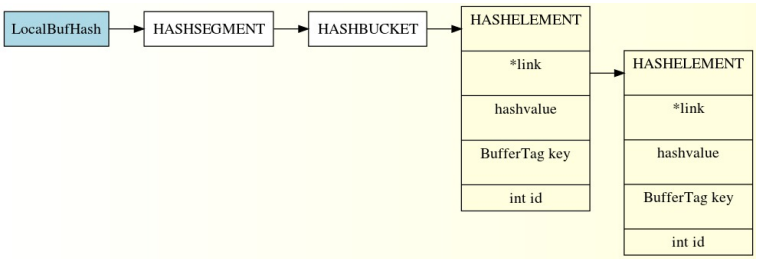

Figure 9: local buffer中的哈希

Table 9: 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