**9.5**

1. 每磁道的容量为 512\*50= 25600B = 25K；

每个表面则是25K\*2000= 50 000K；

该磁盘容量为 50 000K\*5\*2= 500 000K。

1. 每个盘面的磁道数就是柱面数，即2000。
2. 有效的块大小应该是扇区大小的整数倍，同时不能超过每个磁道的大小，故256和51200不是有效的块大小，而2048是有效的块大小。
3. 最大旋转延迟为：
4. 传输率为：

**9.6**

1.每块能存放的记录量为：1024/100 = 10.

2.总共需要的块数为：100 000/10 = 10 000.

总共需要的柱面数为：10 000/(25K/1024\*10) = 40.

故总共需要全部的10个盘面。

3.磁盘总共的块数量为500 000K/1024 = 500 000，又每块可以存储10个记录，故总共可存储的记录数为 500 000\*10 = 5 000 000.

4.每个磁道可以存储 25K/1024= 25块，故下一盘面的第一磁道的第一块应该存储第26页。所有磁头可以并行读写的话，所有盘面的第一磁道的第一块都可以并行读写，故下一盘面的第一磁道的第一块应该存储第二页。

5.该文件共有100 000/10 = 10 000个块，每个磁道可以存储25K/1024= 25个块，故总需10 000/25= 400个磁道，每个磁道的旋转延迟取0.011s，则总的传输延迟为400\*0.011= 4.4s。总的寻道次数为400/10= 40，每次花费0.01s，总花费0.4s。故总的时间为4.8s。

若为同步读取，则寻道次数和寻道时间不变，为0.4s，但由于可以同时读取10个磁道的数据，旋转延迟时间变为原来的十分之一，0.44s，故总的时间为0.84s。

1. 对于每个块：

寻道时间 0.01s；

旋转延迟 ；

传输时间 = 4.4\*10-4s.

总的时间为0.01644s。故总的随机读取时间为0.01644s\*10 000=164.4s

**9.12**

指的是在某些情况下， LRU方法会造成每次读数据都会读取文件的每一页的情况。这种情况下LRU方法无疑是极糟糕的方法。

**9.17**

一种是定长记录，一种是变长记录。定长记录可以通过维护一个比特位的数组来管理记录，每个比特位表示相对应的位置是否存在记录。而变长记录则通过维护一个槽目录来管理记录，其由<record offser, recorde length>对组成。

**8.4**

形式二<n, (j, k)>中n表示用于查找的关键字，j表示记录在第j页中，k表示在这页中的位置，形式三类似。

1.不可以，因为记录中age不唯一；

2.<11, (1,1)>, < 12, (1,2)>, < 18, (1,3)>, <19, (2,1)> , <19, (2,2)>；

3. <11, (1,1)>, < 12, (1,2)>, < 18, (1,3)>, <19, <(2,1), (2,2)>>；

4.11, 19 由于是聚簇形式，故顺序是重要的；

5. <11, (1,1)>, <19, (2,1)>顺序重要；

6. <11, (1,1)>, <19, <(2,1), (2,2)>>顺序重要；

7.不可以，因为gpa不是整数，并且当记录增长后不可避免存在重复；

8. <1.8, (1,1)>, <2.0, (1,2)> , <3.2, (2,1), <3.4, (1,3)>, <3.8, (2,2)>；

9. <1.8, (1,1)>, <2.0, (1,2)> , <3.2, (2,1), <3.4, (1,3)>, <3.8, (2,2)>；

10.不可以，因为文件并不是按gpa排序存储的；

11.不可以，因为文件并不是按gpa排序存储的；

12.不可以，因为文件并不是按gpa排序存储的；

**8.11**

***1.*** 选择B，因为这样可以只在索引上遍历一遍就可以完成查询。如果不支持仅在索引上的计算的话选择E，因为查找是要打印所有的记录，故都是要遍历所有记录的，建立索引没有任何帮助。

***2.*** 选择C，因为这样可以根据索引查找出第一个符合条件的记录，然后可以直接向后遍历直到不符合条件，这样就可以找出所有的需要查找的记录。并且这个方案无论系统是否支持仅在索引上的查找都可以。

**4.3**

1.

)}

2.

)}

3.

*{ T | ∃T*1 *∈ Catalog*( *∃X ∈ P arts*(*X.color* = ‘*red ∧ X.pid* = *T*1*.pid*)*∧T.sid* = *T*1*.sid*)*∨∃T*2 *∈ Suppliers*(*T*2*.address* = 221*P ackerStreet ∧ T.sid* = *T*2*.sid*)*}*

4. *ρ*(*R*1*, πsid*((*πpidσc o l o r*=  *r ed P arts*)  *Catalog*))

*ρ*(*R*2*, πsid*((*πpidσc o l o r*=  *g r een P arts*)  *Catalog*))

*R*1 *∩ R*2

*{T | ∃T*1 *∈ Catalog*(*∃X ∈ P arts*(*X.color* = ‘*red ∧ X.pid* = *T*1*.pid*)

*∧∃T*2 *∈ Catalog*(*∃Y ∈ P arts*(*Y.color* =  *green ∧ Y.pid* = *T*2*.pid*)

*∧T*2*.sid* = *T*1*.sid*) *∧ T.sid* = *T*1*.sid*)*}*

5. (*πsid , pidCatalog*)*/*(*πpidParts*)

*{T | ∃T*1 *∈ Catalog*( *∀X ∈ Parts*( *∃T*2 *∈ Catalog*(*T*2*.pid* = *X.pid ∧ T*2*.sid* = *T*1*.sid*)) *∧ T.sid* = *T*1*.sid*) *}*

*6.* (*πsid , pidCatalog*)*/*(*πpidσcolor*=  *r ed Parts*)

*{T | ∃T*1 *∈ Catalog*( *∀X ∈ Parts*(*X.color* = ‘*red ∨∃T*2 *∈ Catalog*(*T*2*.pid* = *X.pid ∧ T*2*.sid* = *T*1*.sid*))*∧T.sid* = *T*1*.sid*) *}*

*7.* (*πsid , pidCatalog*)*/*(*πpidσc o l o r*=  *r ed ∨ c o l o r*=  *g r een Parts*)

*{T | ∃T*1 *∈ Catalog*( *∀X ∈ Parts*((*X.color* = ‘*red*  
*∧X.color* = ‘*green*) *∨ ∃T*2 *∈ Catalog*  
(*T*2*.pid* = *X.pid ∧ T*2*.sid* = *T*1*.sid*)) *∧ T.sid* = *T*1*.sid*) *}*

*8. ρ*(*R*1*,* ((*πsid , pidCatalog*)*/*(*πpidσc o l o r*=  *r ed Parts*)))  
*ρ*(*R*2*,* ((*πsid , pidCatalog*)*/*(*πpidσc o l o r*=  *g r een Parts*)))  
*R*1 *∪ R*2

*{ T | ∃T*1 *∈ Catalog*(( *∀X ∈ P arts*  
(*X.color* = ‘*red ∨ ∃Y ∈ Catalog*(*Y.pid* = *X.pid ∧ Y.sid* = *T*1*.sid*))  
*∨∀Z ∈ P arts*(*Z.color* = ‘*green ∨ ∃P ∈ Catalog*  
(*P.pid* = *Z.pid ∧ P.sid* = *T*1*.sid*))) *∧ T.sid* = *T*1*.sid*)*}*

*9. ρ*(*R*1*, Catalog*)  
*ρ*(*R*2*, Catalog*)  
*πR*1 *. sid , R*2 *. sid*(*σR*1 *. pid*= *R*2 *. pid ∧ R*1 *. sid* = *R*2 *. sid ∧ R*1 *. c o st> R*2 *. c o st*(*R*1 *× R*2))

*{T | ∃T*1 *∈ Catalog*(*∃T*2 *∈ Catalog*  
(*T*2*.pid* = *T*1*.pid ∧ T*2*.sid* = *T*1*.sid*  
*∧T*2*.cost < T*1*.cost ∧ T.sid*2 = *T*2*.sid*)  
*∧T.sid*1 = *T*1*.sid*)*}*

*10. ρ*(*R*1*, Catalog*)  
*ρ*(*R*2*, Catalog*)  
*πR*1 *. pidσR*1 *. pid*= *R*2 *. pid ∧ R*1 *. sid* = *R*2 *. sid*(*R*1 *× R*2)

*{T | ∃T*1 *∈ Catalog*( *∃T*2 *∈ Catalog*  
(*T*2*.pid* = *T*1*.pid ∧ T*2*.sid* = *T*1*.sid*)  
*∧T.pid* = *T*1*.pid*) *}*

*11. ρ*(*R*1*, πsidσsnam e*=  *Y o sem iteS h am Suppliers*)  
*ρ*(*R*2*, R*1  *Catalog*)  
*ρ*(*R*3*, R*2)  
*ρ*(*R*4(1 *→ sid,* 2 *→ pid,* 3 *→ cost*)*, σR*3 *. c o st< R*2 *. c o st*(*R*3 *× R*2))  
*πpid*(*R*2 *− πsid , pid , c o stR*4)

*{ T | ∃T*1 *∈ Catalog*( *∃X ∈ Suppliers*  
(*X.sname* =  *Y osemiteSham ∧ X.sid* = *T*1*.sid*) *∧ ¬*(*∃S ∈ Suppliers*  
(*S.sname* =  *Y osemiteSham ∧ ∃Z ∈ Catalog*  
(*Z.sid* = *S.sid ∧ Z.cost > T*1*.cost*))) *∧ T.pid* = *T*1*.pid*)

12.

**5.4**

1. SELECT E.ename, E.age

FROM Emp E, Works W1, Works W2, Dept D1, Dept D2

WHERE E.eid = W1.eid AND W1.did = D1.did AND D1.dname = ‘Hardware’ AND

E.eid = W2.eid AND W2.did = D2.did AND D2.dname = ‘Software’

1. SELECT W.did, COUNT (W.eid)

FROM Works W

GROUP BY W.did

HAVING 2000 *<* ( SELECT SUM (W1.pct time)

FROM Works W1

WHERE W1.did = W.did )

1. SELECT E.ename

FROM Emp E

WHERE E.salary *>* ALL (SELECT D.budget

FROM Dept D, Works W

WHERE E.eid = W.eid AND D.did = W.did)

1. SELECT DISTINCT D.managerid

FROM Dept D

WHERE 1000000 *<* ALL (SELECT D2.budget

FROM Dept D2

WHERE D2.managerid = D.managerid )

1. SELECT E.ename

FROM Emp E

WHERE E.eid IN (SELECT D.managerid

FROM Dept D

WHERE D.budget = (SELECT MAX (D2.budget)

FROM Dept D2))

1. SELECT D.managerid

FROM Dept D

WHERE 5000000 *<* (SELECT SUM (D2.budget)

FROM Dept D2WHERE D2.managerid = D.managerid )

1. SELECT DISTINCT tempD.managerid

FROM (SELECT DISTINCT D.managerid, SUM (D.budget) AS tempBudget

FROM Dept D

GROUP BY D.managerid ) AS tempD

WHERE tempD.tempBudget = (SELECT MAX (tempD.tempBudget)

FROM tempD)

1. SELECT E.ename

FROM Emp E, Dept D

WHERE E.eid = D.managerid GROUP BY E.Eid, E.ename

HAVING EVERY (D.budget *>* 1000000) AND ANY (D.budget *<* 5000000)

**5.8**

1. CREATE TABLE Student ( snum INTEGER,

sname CHAR(20) ,

major CHAR(20) ,

level CHAR(20) ,

age INTEGER,

PRIMARY KEY (snum))

CREATE TABLE Faculty ( fid INTEGER,

fname CHAR(20) ,

deptid INTEGER,

PRIMARY KEY (fnum))

CREATE TABLE Class ( name CHAR(20) ,

meets atTIME,

room CHAR(10) ,

fid INTEGER,

PRIMARY KEY (name),

FOREIGN KEY (fid) REFERENCES Faculty)

CREATE TABLE Enrolled ( snum INTEGER,

cname CHAR(20) ,

PRIMARY KEY (snum, cname),

FOREIGN KEY (snum) REFERENCES Student,

FOREIGN KEY (cname) REFERENCES Class)

2.

a. CREATE TABLE Enrolled ( snum INTEGER,

cname CHAR(20) ,

PRIMARYKEY (snum, cname),

FOREIGNKEY (snum) REFERENCES Student),

FOREIGN KEY (cname) REFERENCES Class,),

CHECK (( SELECT COUNT (E.snum)

FROM Enrolled E

GROUP BY E.cname) >= 5),

CHECK (( SELECT COUNT (E.snum)

FROM Enrolled E

GROUP BY E.cname) <= 30))

b.不用修改，因为教室已经跟课程相关，故每有一间新教室就会有一个与之相对应的课程，不会出现没有课程的教室。

c. CREATE ASSERTION TeachTwo

CHECK ( ( SELECT COUNT (\*)

FROM Facult F, Class C

WHERE F.fid = C.fid

GROUP BY C.fid

HAVING COUNT (\*) < 2) = 0)

d.

CREATE ASSERTION NoTeachThree

CHECK ( ( SELECT COUNT (\*)

FROM Facult F, Class C

WHERE F.fid = C.fid AND F.deptid 33

GROUP BY C.fid

HAVING COUNT (\*) > 3) = 0)

e.

CREATE ASSERTION InMath101

CHECK (( SELECT COUNT (\*)

FROM Student S

WHERE S.snum NOT IN ( SELECT E.snum

FROM Enrolled E

WHERE E.cname = ’Math101’)) = 0)

f.

CREATE TABLE Class ( name CHAR(20) ,

meets at TIME,

room CHAR(10) ,

fid INTEGER,

PRIMARYKEY (name),

FOREIGNKEY (fid) REFERENCES Faculty),

CHECK ( (SELECT MIN (meets at)

FROM Class) <>

(SELECT MAX (meets at)

FROM Class)))

g.

CREATE TABLE Class ( name CHAR(20) ,

meets at TIME,

room CHAR(10) ,

fid INTEGER,

PRIMARY KEY (name),

FOREIGN KEY (fid) REFERENCES Faculty),

CHECK ((SELECT COUNT (\*)

FROM ( SELECT C.room, C.meets

FROM Class C

GROUP BY C.room, C.meets

HAVING COUNT (\*) > 1)) = 0))

h.

CREATE TABLE Faculty ( fid INTEGER,

fname CHAR(20) ,

deptid INTEGER,

PRIMARY KEY (fnum),

CHECK ( (SELECT MAX (\*)

FROM ( SELECT COUNT (\*)

FROM Faculty F

GROUP BY F.deptid))

< 2 \*

(SELECT MIN (\*)

FROM ( SELECT COUNT (\*)

FROM Faculty F

GROUP BY F.deptid))))

i.

CREATE TABLE Faculty (fid INTEGER,

fname CHAR(20) ,

deptid INTEGER,

PRIMARY KEY (fnum),

CHECK ( ( SELECT COUNT (\*)

FROM Faculty F

GROUP BY F.deptid

HAVING COUNT (\*) > 10) = 0))

j.

不能实现这个功能，因为SQL没有可以限制每次更新数据的范围的语句。

k.

CREATETABLE Student ( snum INTEGER,

sname CHAR(20) ,

major CHAR(20) ,

level CHAR(20) ,

age INTEGER,

PRIMARY KEY (snum),

CHECK ( ( SELECT COUNT (\*)

FROM Student S

WHERE S.major = ’CS’) >

(SELECT COUNT (\*)

FROM Student S

WHERE S.major = ’Math’)))

l.

CREATE ASSERTION MoreCSMajors

CHECK ( (SELECT COUNT (E.cname)

FROM Enrolled E, Student S

WHERE S.snum = E.snum AND S.major = ’CS’) >

(SELECT COUNT (E.cname)

FROM Enrolled E, Student S

WHERE S.snum = E.snum AND S.major = ’Math’))

m.

CREATE ASSERTION MoreEnrolledThanMath

CHECK ( (SELECT COUNT (E.snum)

FROM Enrolled E, Faculty F, Class C

WHERE E.cname = C.name

AND C.fid = F.fid AND F.deptid = 33) >

(SELECT COUNT (E.snum)

FROM Student S

WHERE S.major = ’Math’))

n.

CREATE TABLE Student ( snum INTEGER,

sname CHAR(20) ,

major CHAR(20) ,

level CHAR(20),

age INTEGER,

PRIMARY KEY (snum),

CHECK ((SELECT COUNT (S.snum)

FROM Student S

WHERE S.major = ’CS’) > 0 ))

o.

CREATE ASSERTION NotSameRoom

CHECK ( (SELECT COUNT (\*)

FROM Faculty F1, Faculty F2, Class C1, Class C2

WHERE F1.fid = C1.fid

AND F2.fid = C2.fid

AND C1.room = C2.room

AND F1.deptid F2.deptid) = 0)