作 业

1. Consider the table *loan* (loan-number, branch-name, amount) in the above-mentioned banking enterprise database. It is assumed that this table owns 100,000 rows and is stored in a DB file in DB2/Sybase/SQL Server DBS.

Some data types of attributes in the table supported by DB2/Sybase/SQL Server are listed as follows:

(Note: there are four data types describing the Integer attribute)

data type	bytes occupied	range
tinyint	1 bytes	0 ~ 255
smallint	2 bytes	-32,768 ~ 32,767
int	4 bytes	-2,147,483,648 ~ 2,147,483,647
bigint	8 bytes	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
char(n)	*n* bytes	Character string with fixed-length of *n*
float	4 bytes	floating-point number, range: -1.79E + 38 ~ 1.79E + 38
double	8 bytes	double precision floating-point number, range: -3.40E + 38 到 3.40E + 38

Given the following information about the columns in the table *loan*,

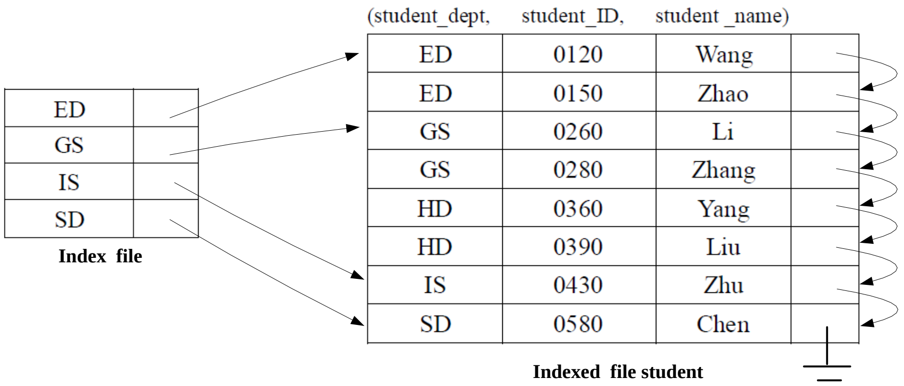
attribute	features
loan-number	Integer date type, with a range from 20,000,101 to 20,120,101
branch-name	character string, with the fixed-length of 8
amount	Numeric data type, and its value is between [0, 10,000,000]

(1) For the table *loan*, choose ***appropriate*** data types for its three columns, and give an estimation of the storage space (taking bytes as the units) of this table in the DB file.

(2) The index built on *loan* is supposed to be a multi-level index, in the form of a complete binary tree(完全二叉树), and the height of a tree's leaf node on average is 8. (Note: the height of the *root* node in the tree is 0).

If the size of the node in the index tree is on average 20 bytes, how many bytes will be occupied by the index tree?

2. Give the data file student(student_dept, student_ID, student_name) as shown below, which is organized as a sequential file, taking the attribute student_dept as the search key.



ED	
GS	
IS	
SD	

Index file

(student_dept,	student_ID,	student _name)	
ED	0120	Wang	
ED	0150	Zhao	
GS	0260	Li	
GS	0280	Zhang	
HD	0360	Yang	
HD	0390	Liu	
IS	0430	Zhu	
SD	0580	Chen	

Indexed file student

(1) Is the index a dense or sparse index, why?

(2) If a tuple (IS, 0430, Zhu) is deleted from the indexed file, depict the indexed file and the index file.

3. Consider the following tables in the database ***University***:

Student(*sid*, *name*, *department*, *classid*, *age*)
Department(*dname*, *building*, *budget*)

(1) For the table *Student*(*sid*, *name*, *department*, *age*), a primary/cluster index has been defined on *sid*. Then, are the tuples in the table organized as a heap file or a sequential file, and why?

(2) Consider the following SQL query. In addition to the existing primary indices on the primary keys of the tables, on which attributes in the tables the indices can be further defined to speed up the query?

select *department*, sum(*sid*)
from *Student* as A, *Department* as B
where *building*='T3' and *age*>20 and *A.department*= *B.dname*
group by *department*

(3) For the following query

update *Student*
set *age*=*age*+2
where *sid* between 211301 and 211318

i) Give a statement to define a nonclustering index (i.e. secondary index) on the attribute *age*.

ii) Does this index speed up or slow down the operation, and why?

(4) Suppose a multi-attribute index (or composite index) on attributes (department, classid, age) in relation *Student*.

Can this index speed up the following queries，why?

```
(i)     select   serialID
        from     Student
        where    department='CS' and classid=2019211

(ii)    select    sid, name
        from      Student
        where     classid=2019211 and age between 17 and 23

(iii)   select    sid, name
        from      Student
        where     age between 17 and 23
```

4. Given the relation *student*(*Sno*, *Sdept*, *Sname*, Ssocre) as shown below. A primary key is bulit on the attribute Sno. In addition to the index on Sno, a nonclustering index is also created on the attribute Sdept. Given a figure to illustrate the physical structures of the indexed file and clustering/nonclustering files, assuming that the indexes are organized as B^+ trees.

Sno	*Sdept*	*Sname*	*Ssocre*
S1	Automation	Bai	88
S2	Automation	Wang	90
S3	Computer	Yu	60
S4	Computer	Li	67
S5	Economy	Xin	67
S6	Finance	Liu	67
S7	History	An	70

例题及答案

1. Consider the table *loan* (loan-number, branch-name, amount) in the above-mentioned banking enterprise database. It is assumed that this table owns 100,000 rows and is stored in a DB file in DB2/Sybase/SQL Server DBS.

Some data types of attributes in the table supported by DB2/Sybase/SQL Server are listed as follows:

(Note: there are four data types describing the Integer attribute)

data type	bytes occupied	range
tinyint	1 bytes	0 ~ 255
smallint	2 bytes	-32,768 ~ 32,767
int	4 bytes	-2,147,483,648 ~ 2,147,483,647
bigint	8 bytes	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
char(n)	*n* bytes	Character string with fixed-length of *n*
float	4 bytes	floating-point number, range: -1.79E + 38 ~ 1.79E + 38
double	8 bytes	double precision floating-point number, range: -3.40E + 38 到 3.40E + 38

Given the following information about the columns in the table *loan*,

attribute	features
loan-number	Integer date type, with a range from 20,000,101 to 20,120,101
branch-name	character string, with the fixed-length of 8
amount	Numeric data type, and its value is between [0, 10,000,000]

(2) For the table *loan*, choose ***appropriate*** data types for its three columns, and give an estimation of the storage space (taking bytes as the units) of this table in the DB file.

Answers:

为表中的 3 个属性选择如下数据类型

attribute	data type	bytes occupied
loan-number	int	4
branch-name	char(8)	8
amount	float	4

表中每一行需要 16 个 bytes 的存储空间，共有 100,000 行，因此所需总空间为：

16 *100,000 bytes = 1,600,000 bytes ≈1.6M

正确选择属性数据类型 2 分，文件存储空间结果 2 分。

(3) If we often need to retrieve the table loan by branch-name, use a SQL statement to define an index to speed up queries.

Answer:

Create index name-index **on** *loan* (branch-name)

(3) The index built on *loan* is supposed to be a multi-level index, in the form of a complete binary tree(完全二叉树), and the height of a tree's leaf node on average is 8. (Note: the height of the *root* node in the tree is 0).

If the size of the node in the index tree is on average 20 bytes, how many bytes will be occupied by the index tree?

Answer:

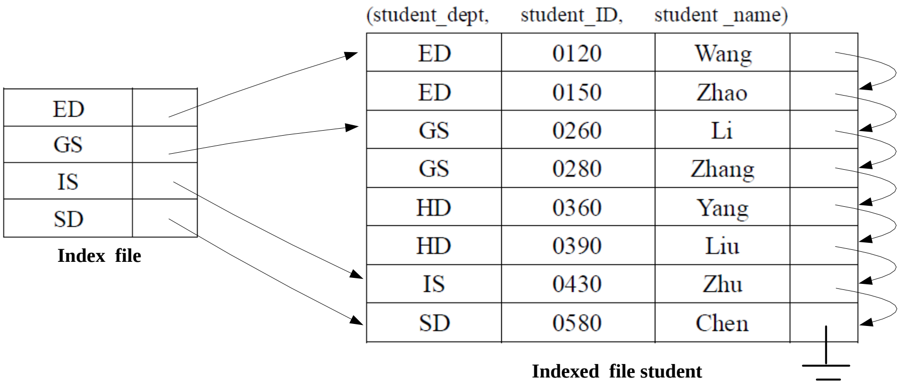
索引树中的根结点、非叶结点、叶结点总数为：

$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 =$ （1 + 2 +4 +8 + 16 + 32 + 64 + 128 + 256） = 511

索引树所占空间为：

511 * 20 bytes≈10K

2. Give the data file student(student_dept, student_ID, student_name) as shown below, which is organized as a sequential file, taking the attribute *student_dept* as the search key.



ED
GS
IS
SD

Index file

(student_dept,	student_ID,	student _name)
ED	0120	Wang
ED	0150	Zhao
GS	0260	Li
GS	0280	Zhang
HD	0360	Yang
HD	0390	Liu
IS	0430	Zhu
SD	0580	Chen

Indexed file student

(1) Is the index a dense or sparse index, why?

Answer:

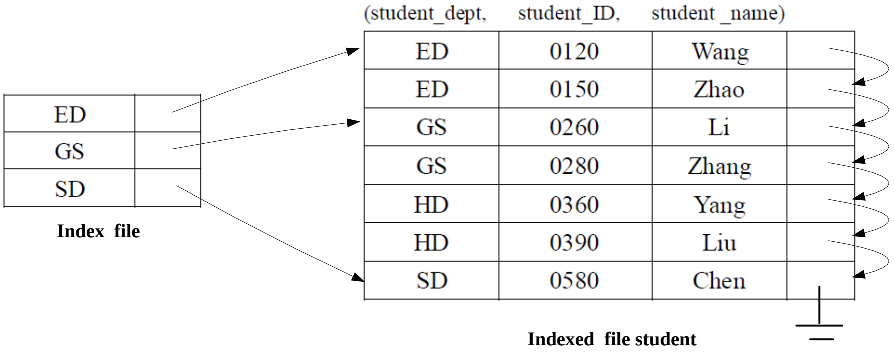
(1) Sparse index.

Because the index records are created only for the search key values ED, GS, IS, SD of student_dept rather than created for every search key value in student_dept.

(2) If a tuple (IS, 0430, Zhu) is deleted from the indexed file, depict the indexed file and the index file.

Answer:

The indexed file and the index file after deleting the tuple (IS, 0430, Zhu) are shown as follows:



ED	
GS	
SD	

Index file

(student_dept,	student_ID,	student _name)	
ED	0120	Wang	
ED	0150	Zhao	
GS	0260	Li	
GS	0280	Zhang	
HD	0360	Yang	
HD	0390	Liu	
SD	0580	Chen	

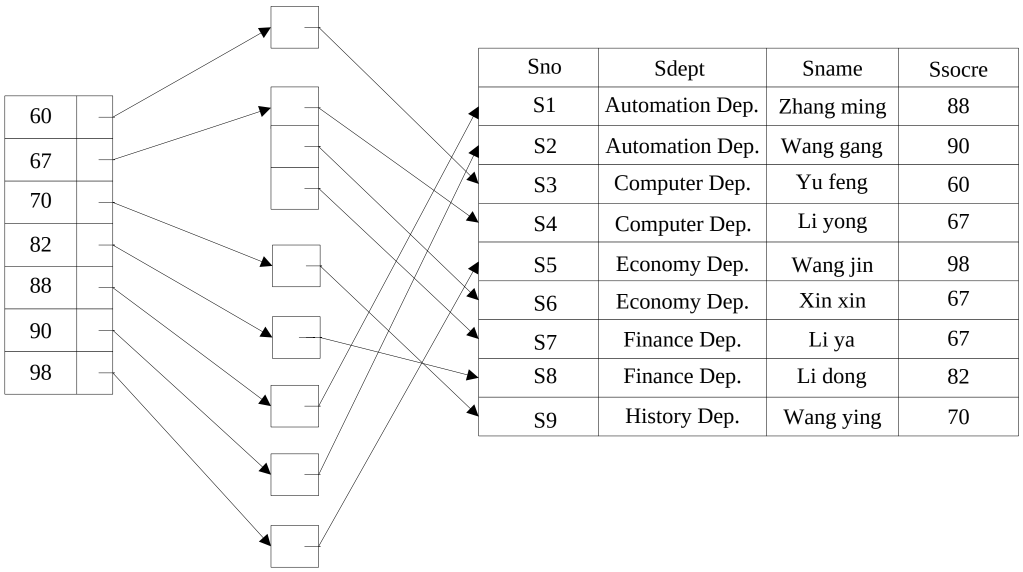
Indexed file student

3. Given the relation student(Sno, Sdept, Sname, Ssocre) as shown below, which is organized as a sequential file.

Taking the attribute Sscore as the search key, define a *dense* and secondary index for the indexed file *student*. The index file and index entries in the index file should be drawn.

Sno	Sdept	Sname	Ssocre
S1	Automation Dep.	Zhang ming	88
S2	Automation Dep.	Wang gang	90
S3	Computer Dep.	Yu feng	60
S4	Computer Dep.	Li yong	67
S5	Economy Dep.	Wang jin	98
S6	Economy Dep.	Xin xin	67
S7	Finance Dep.	Li ya	67
S8	Finance Dep.	Li dong	82
S9	History Dep.	Wang ying	70

Answer:



Sno	Sdept	Sname	Ssocre
S1	Automation Dep.	Zhang ming	88
S2	Automation Dep.	Wang gang	90
S3	Computer Dep.	Yu feng	60
S4	Computer Dep.	Li yong	67
S5	Economy Dep.	Wang jin	98
S6	Economy Dep.	Xin xin	67
S7	Finance Dep.	Li ya	67
S8	Finance Dep.	Li dong	82
S9	History Dep.	Wang ying	70

the index file *stu_index*　　　　　　the indexed file *student*

4. Consider the following tables in the database ***University***:

Student(*sid*, *name*, *department*, *age*)

Department(*dname*, *building*, *budget*)

(1) For the table *Student*(*sid*, *name*, *department*, *age*), a primary/cluster index has been defined on *sid*. Then, are the tuples in the table organized as a heap file or a sequential file, and why?

答案：

组织为顺序文件。

该表已经定义了主键和主索引，因此表中数据将组织成顺序文件，按照主键 *sid* 的顺序在文件中依次存储表中各个元组。

(2) Consider the following SQL query. In addition to the existing primary indices on the primary keys of the tables, on which attributes in the tables the indices can be further defined to speed up the query?

```
select   department, sum(sid)
from     Student as A, Department as B
where    building='T3' and age>20 and A.department= B.dname
group by department
```

答案：

可以分别在：

i) *Student* 表的 *age*,

ii) *Student* 表的 *department*,

iii) *Department* 表的 *building*

，这 3 个属性上建立索引。

(3) For the following query

```
update Student
set    age=age+2
where sid between 211301 and 211318
```

, a nonclustering index has been defined on the attribute *age*.

Does this index speed up or slow down the operation, and why?

答案:

age 上的 nonclustering index 将 slow down *update* 操作。

因为，当通过 update 操作修改现有元组的 age 值后，DBMS 重新调整定义在 *age* 上的非聚集索引，降低了 update 操作的执行速度。

(4) Suppose a multi-attribute index (or composite index) on attributes (department, classid, age) in relation *Student*.

Can this index speed up the following queries，why?

```
(i)   select   serialID
      from     Student
      where    department='CS' and classid=2019211
```

where 条件使用了索引的两个属性，且满足左前缀要求，索引起作用，可加速查询

```
(ii)  select   sid, name
      from     Student
      where    classid=2019211 and age between 17 and 23
```

where 条件使用了索引的 2 个属性 date，不满足左前缀要求，索引不起作用，无法加速查询

```
(iii) select   sid, name
      from     Student
      where    age between 17 and 23
```

where 条件使用了索引的 1 个属性 date，不满足左前缀要求，索引不起作用，无法加速查询

5.Consider the following tables in the database ***University***:

instructor(*ID*, *name*, *dept-name*, *salary*)

department(*dept-name*, *building*, *budget*)

(1) Consider the following SQL query. In addition to the existing primary indices on the primary

keys of the tables, on which attributes in the tables the indices can be further defined to speed up the query?

```
select    A.dept-name, avg(salary)
from      instructor as A, course as B
where     budget >5000 and salary>1000 and A.dept-name= B.dept-name
group by   A.dept-name
```

(2) For the following *insert* operation

```
insert into department(dept-name, building, budget)
            values(Language, Building_1, 50000)
```

a clustering index has been defined on the primary key *dept-name*.

Does this index speed up or slow down the operation, and why?

(3) For the following query

```
update instructor
set   salary=salary + 1000
where   dept-name=Comp.Sci
```

i) Give a SQL statement to define a nonclustering index (or secondary index) on the attribute *salary*.

ii) Does this index speed up or slow down the operation, and why?

答案:

（1）可以分别在:

(1) *instructor* 表的 *dept-name*

(2) *instructor* 表的 *salary*

(3) *department* 表的 *budget*

这 3 个属性上建立索引。

（2）*dept-name* 上的 clustering index 将 slow down insert 操作。(

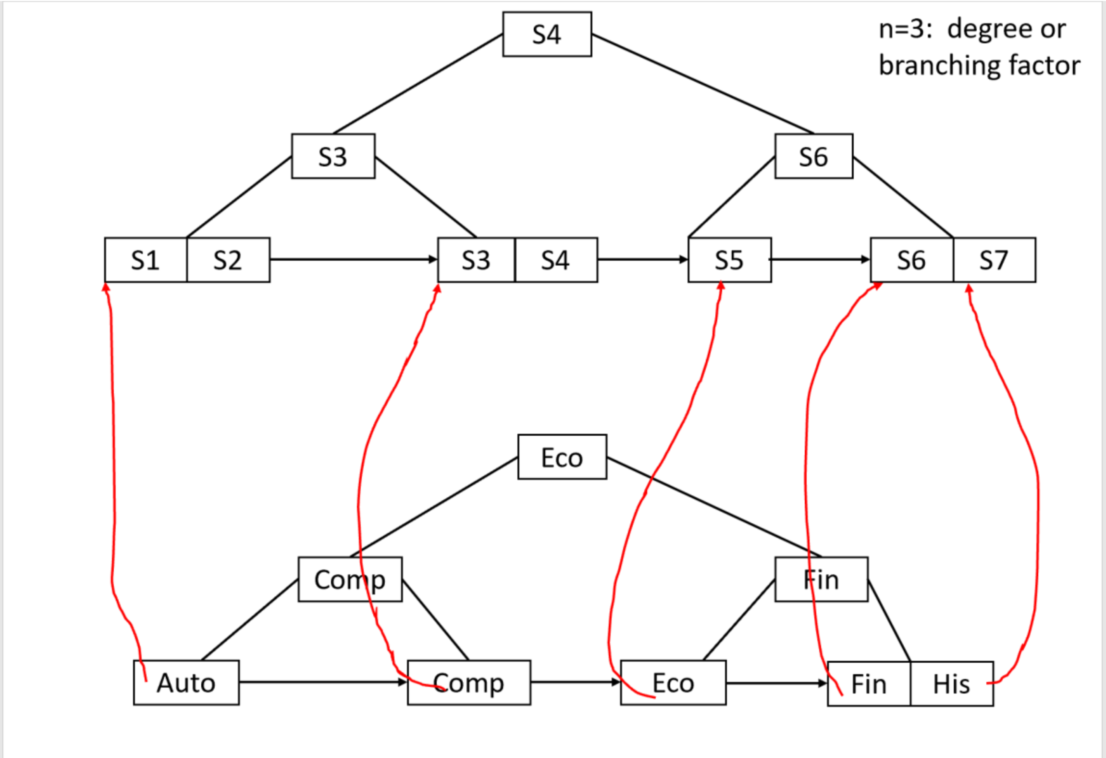
因为，索引无法加快插入操作速度。反而是，当插入新元组后，DBMS 重新调整定义在 *dept-name* 上的聚集索引，反而降低了插入操作的执行速度。

（3）*salary* 上的 nonclustering index 将 slow down *update* 操作。

因为，当通过 update 操作修改现有元组的 *salary* 值后，DBMS 重新调整定义在 *salary* 上的非聚集索引，降低了 update 操作的执行速度。【索引并没有建立 where 查询条件属性 dept-name 上，而是建立在 set 子句中的 salary 属性上。】

6. Given the relation *student*(*Sno*, *Sdept*, *Sname*, Ssocre) as shown below. A primary key is bulit on the attribute Sno. In addition to the index on Sno, a nonclustering index is also created on the attribute Sdept. Given a figure to illustrate the physical structures of the indexed file and clustering/nonclustering files, assuming that the indexes are organized as B$^+$ trees.

Sno	*Sdept*	*Sname*	*Ssocre*
S1	Automation	Bai	88
S2	Automation	Wang	90
S3	Computer	Yu	60
S4	Computer	Li	67
S5	Economy	Xin	67
S6	Finance	Liu	67
S7	History	An	70



n=3: degree or branching factor

答案不唯一

聚集索引文件，平衡二叉树，3 层，7 个叶节点；

非聚集索引，平衡二叉树，3 层，5 个叶节点；

对每个非叶节点： 左子树中后代结点的 search key 值<非叶节点的 search key 值≤右子树中后代结点的 search key 值；

或者：左子树中后代结点的 search key 值≤非叶节点的 search key 值< 右子树中后代结点的 search key 值