Quiz1

1. **According to the levels of data abstraction, database systems have several schemas, which are as follows:**

**Physical schema**, **Logical schema**, and **View schema**.

2. **List the names of data model used in database systems, and explain briefly how or when to use it.**

(i) **E-R model:** It is used in the conceptual database design, which describes the real world objects with entities, and relationships among these objects.

(ii) **Relational model:** It is used in the logical database design. It is the basis of the most DBMS products in the database world for more than 30 years, and it uses tables to represent both data and the relationships among those data.

(iii) **Object-oriented model:** It is heavily influenced by object-oriented programming language and can be understood as an attempt to add DBMS functionality to a programming language environment. It can be regarded as extending the E-R model with notions of encapsulation, methods and object identity, and support complex data structure. The model can be used in logical database design in most cases, and can also be used in conceptual design step.

(iv) **Object-relational model:** It extends the relational model with features of object-oriented model.

(v) **Semistructured-data model:** It is used in the logical database design steps. In this model, XML is widely used to represent semistructured data.

(vi) **Network model:** Data are represented by collections of records, and relationships among data are represented by links, which can be viewed as pointers. The records in the database are organized as collections of arbitrary graphs. It was only used in old database systems.

(vii) **Hierarchical model:** It is similar to the network model in the sense that data and relationships among data are represented by records and links, respectively. But it differs from the network model in that the records are organized as collections of trees rather than arbitrary graphs. It was used only in old database systems.

3. **What are the physical data independence and logical data independence in database system?**

**Physical Data Independence:** the ability to modify the physical schema without changing the logical schema, as well as application programs, that is, the modification in the physical schema does not affect the logical schema, there application program need not be changed.

**Logical data independence:** Protect application programs from changes in *logical* structure of data, i.e., the modification in the logical schema does not affect the view schema, there application program need not be changed.

Quiz2

**Problem 1. Relational algebra**

***Student*(*Sid*, *Name*, *Age*)**

**1 2 3**

***Project*(*ProjectName*, *Sid*, *Score*)**

**5 6 7**

$\bowtie$ **15,26,37**

$\times$ 15 16 17 25 26 27 …

**(1) Find the names of students who are in the project with project name 'MiniSQL'.**

$\Pi_{name}((student) \bowtie (\sigma_{projectName=\text{'MiniSQL'}}(project)))$

**(2) Find the Sid of students who have not been in any project team yet.**

$\Pi_{sid}(student) - \Pi_{sid}(project)$

**(3) Find the names of students who are the youngest.**

Method 1: $\Pi_{name}(student) - \Pi_{name}(\sigma_{student.age>st2.age}(student \times (\rho_{st2}(student))))$

Method 2: $\text{Temp} \leftarrow g_{\min(Age)}(student);$

$\Pi_{name}(\sigma_{age=minage}(student \times (\rho_{T(minage)}(\text{Temp}))))$

**Problem 2. Write SQL statement for the following queries.**

***Student*(*Sid*, *Name*, *Age*)**
***Project*(*ProjectName*, *Sid*, *Score*)**
**(1) Find the names of students who get score more than 85 in the project.**

```
SELECT Name
FROM Student S, Project P
WHERE S.Sid = P.Sid and Score > 85
```

**(2) Find the names of students who get the maximum score in each project.**

Method 1:
```
SELECT Name
FROM Student S, Project P
```

```
WHERE S.Sid = P.Sid and Score = (SELECT max(Score)
                                 FROM Project
                                 WHERE ProjectName = P.ProjectName)
```

Method 3:
```
SELECT Name
FROM Student S, Project P
WHERE S.Sid = P.Sid and (ProjectName, Score) in (SELECT ProjectName,
                                                        max(Score)
                                                 FROM Project
                                                 GROUP BY ProjectName)
```

Method 2:
```
SELECT Name
FROM Student
WHERE Sid in (SELECT Sid
              FROM Project P
              WHERE Score >= ALL (SELECT Score
                                  FROM Project
                                  WHERE        ProjectName        =
        P.ProjectName))
```

Quiz3

**Problem 1.** Consider a SQL table ***T*(*A int unique*, *B int*).** Assume there are no NULL values. As specified, attribute *A* is a key. Consider the following three SQL queries:

**Q1: Select *B* From *T***
**Where *B* >= some (Select *B* From *T*);**
**Q2: Select *B* From *T* as $T_1$**
**Where *B* > all (Select *B* From *T* as $T_2$ where $T_2$.A < > $T_1$.A);**
**Q3: Select max(*B*) From *T*;**

Which of the queries above are equivalent? Please show a smallest single instance of *T* you can come up with that demonstrates your answer.

Case 1: Q2 and Q3 are equivalent, if the maximum of *B* is only once.
Case 2: None are equivalent, if the maximum of *B* is multiple times.

| A | B |
|---|---|
| 1 | 5 |
| 2 | 7 |
| 3 | 3 |

Case 1

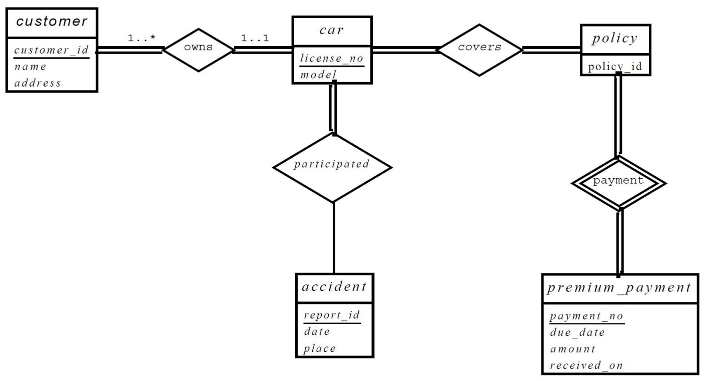
| A | B |
|---|---|
| 1 | 7 |
| 2 | 5 |
| 3 | 7 |

Case2

Case 1:
Q1: 5
7
3
Q2: 7
Q3: 7

Case2:
Q1: 5
7
3
Q2:
Q3: 7

7.1 Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.



**Problem 2.** Consider the following relational schemas describing ***books***, ***publishers***, ***readers***, and reader ratings of the books:

***Book*(*bid*, *title*, *author*, *price*)** // ***bid* is a primary key**

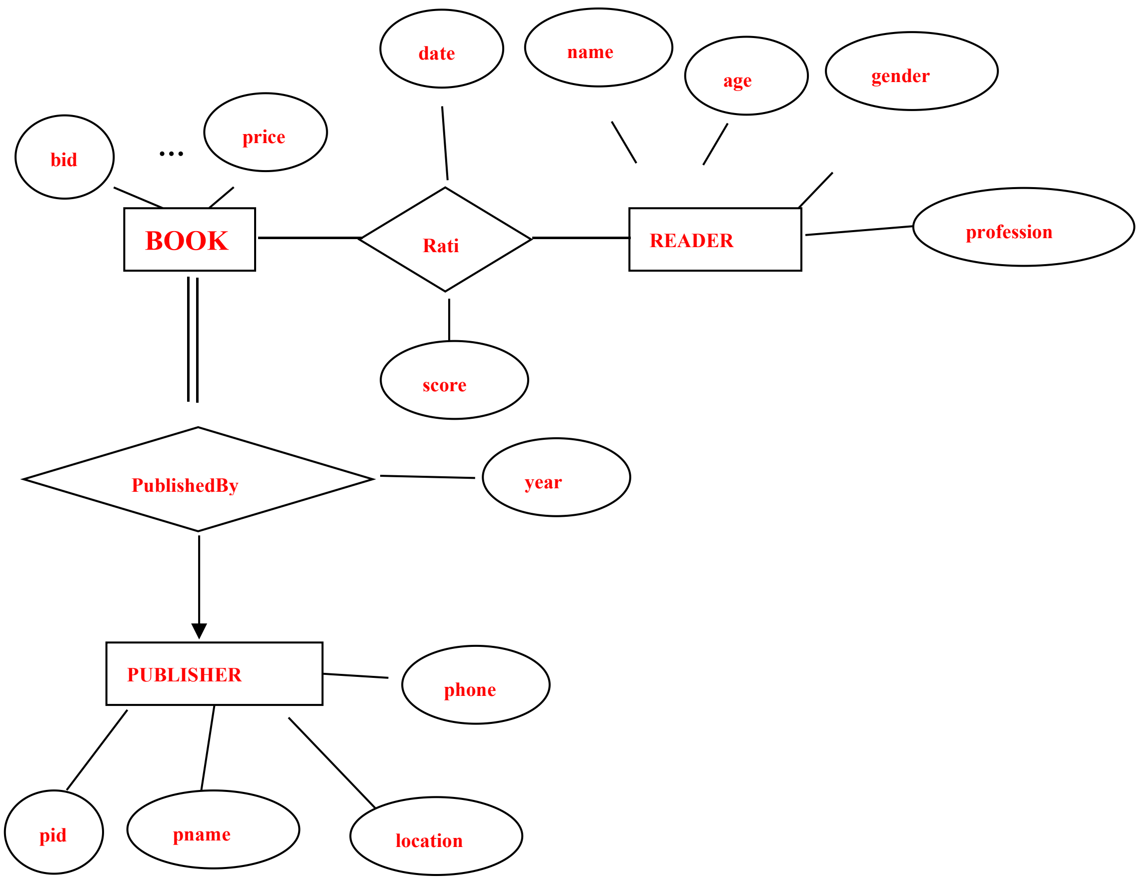
***Reader*(*name*, *age*, *gender*, *profession*)** // ***name* is a primary key**

***Rating*(*name*, *bid*, *date*, *score*)** // **(*name*, *bid*) is a primary key**

***Publisher*(*pid*, *pname*, *location*, *phone*)** // ***pid* is a primary key**

***PublishedBy*(*bid*, *pid*, *year*)** // ***bid* is a primary key**

(1) Draw an E-R diagram from which these relational schemas could have been produced. Your diagram should be fully connected, and it should be as detailed as possible from the information you have.



(2) Please make necessary formalization of the relational schemas above, to get a minimum number of relation schemas.

*Book*(*bid*, *title*, *author*, *price*, *pid*, *year*) // *bid* is a primary key

*Reader*(*name*, *age*, *gender*, *profession*) // *name* is a primary key

*Rating*(*name*, *bid*, *date*, *score*) // (*name*, *bid*) is a primary key

*Publisher*(*pid*, *pname*, *location*, *phone*) // *pid* is a primary key

(3) Write SQL data definition statements for the relation schemas from the issue/step (2), and

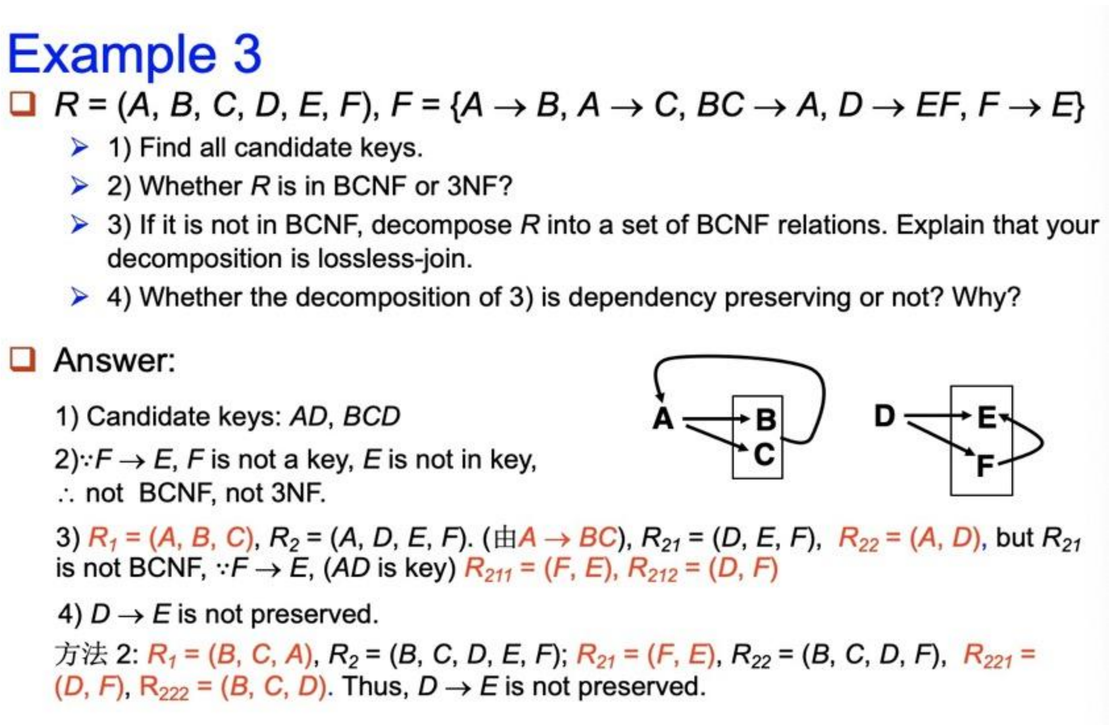
give necessary integrity constraints on them.

```
Create table Publisher(pid         char(10) primary key,
                       pname       varchar(30),
                       location    varchar(50),
                       phone       varchar(20));

Create table Book(bid        char(10) primary key,
                  title      varchar(50),
                  author     varchar(50),
                  price      real,
                  pid        char(10),
                  year       date,
                  Foreign key(pid) references publisher);

Create table Reader(name           varchar(15) primary key,
                    age            int,
                    gender         char(1) not null,
                    profession     varchar(30),
                    check (gender in ('M', 'F')));

Create table Rating(name      varchar(15),
                    bid       char(10),
                    date      date,
                    score     real,
                    primary key (name, bid),
                    foreign key (name) references reader,
                    foreign key (bid) references book);
```



## Example 3

- $R = (A, B, C, D, E, F)$, $F = \{A \rightarrow B, A \rightarrow C, BC \rightarrow A, D \rightarrow EF, F \rightarrow E\}$
  - 1) Find all candidate keys.
  - 2) Whether $R$ is in BCNF or 3NF?
  - 3) If it is not in BCNF, decompose $R$ into a set of BCNF relations. Explain that your decomposition is lossless-join.
  - 4) Whether the decomposition of 3) is dependency preserving or not? Why?

- Answer:

$$\begin{tikzcd} A \arrow[r] \arrow[dr] & \boxed{B} \arrow[l, bend right, "BC"'] \\ & C \end{tikzcd} \qquad \begin{tikzcd} D \arrow[r] \arrow[dr] & E \\ & F \arrow[u, bend right] \end{tikzcd}$$

1) Candidate keys: $AD$, $BCD$

2)$\because F \rightarrow E$, $F$ is not a key, $E$ is not in key,
$\therefore$ not BCNF, not 3NF.

3) $R_1 = (A, B, C)$, $R_2 = (A, D, E, F)$. (由$A \rightarrow BC$), $R_{21} = (D, E, F)$, $R_{22} = (A, D)$, but $R_{21}$ is not BCNF, $\because F \rightarrow E$, ($AD$ is key) $R_{211} = (F, E)$, $R_{212} = (D, F)$

4) $D \rightarrow E$ is not preserved.

方法 2: $R_1 = (B, C, A)$, $R_2 = (B, C, D, E, F)$; $R_{21} = (F, E)$, $R_{22} = (B, C, D, F)$, $R_{221} = (D, F)$, $R_{222} = (B, C, D)$. Thus, $D \rightarrow E$ is not preserved.

判断是否是 3NF 的条件： 对于 R 上的每个函数依赖 X->A （X 是关系 R 属性的一个子集，
A 是 R 的一个属性） ,以下条件中的一个成立：

1 X ∈ A

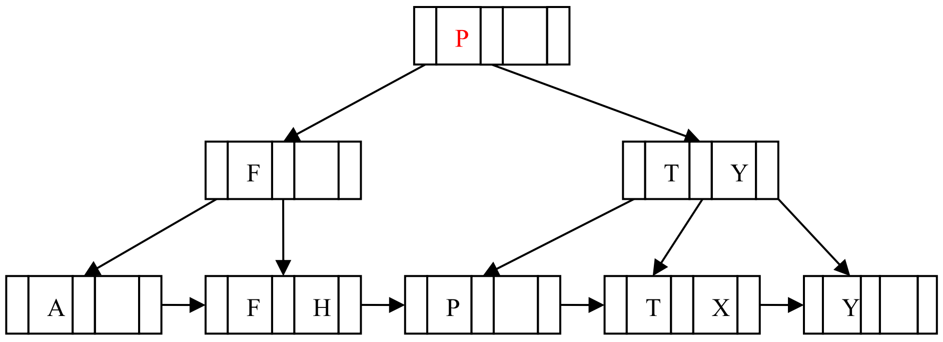
2 X 是超码
3 A 是 R 的码的一部分
判断是否是 BCNF 的条件：对于 R 上的每个函数依赖 X->A（X 是关系 R 属性的一个子
集，A 是 R 的一个属性） ,以下条件中的一个成立：

1 X ∈ A

2 X 是超码

Quiz4

**Problem 1.** For the following B+ tree (n=3):

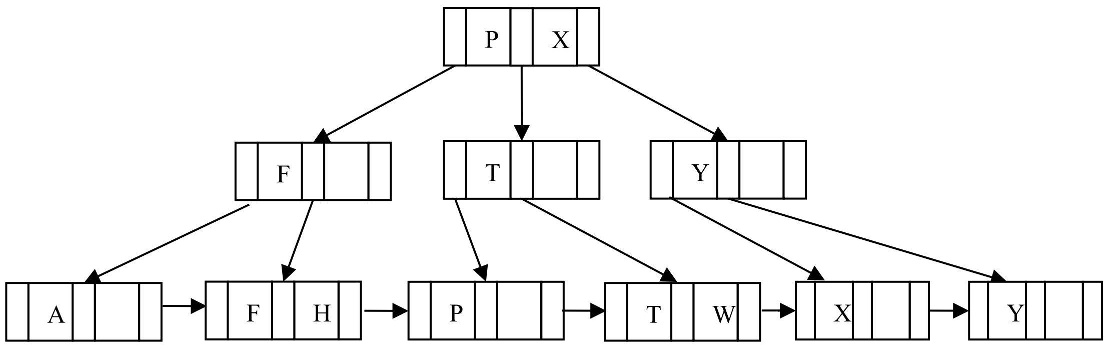


(1) Draw the B+ tree after insert an index item with key 'W' to the given tree.

1.找到子节点，插入

2.如果需要拆分，依次向上，绝不向下。

Solution:



(2) Draw the B+ tree after delete an index item with key 'A' from the original tree.

1.删除该节点,若不满足B+树性质则2

2.往上找父节点p,与p的右边兄弟节点合并

3.判断当前节点父节点符不符合B+树性质

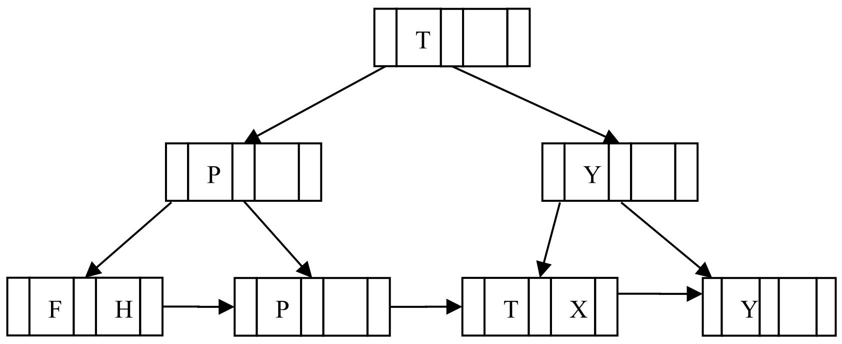
若符合，结束

若不符合，

可拆分，拆分

不可拆分，重复2

Solution:



(3) Assume that the B+ tree contains 1000 index items, please estimate the height of the B+ tree.

$$\left\lceil log_3 \frac{1000}{2} \right\rceil + 1 \leq \text{height} \leq \left\lceil log_{\left\lceil \frac{3}{2} \right\rceil} 1000 \right\rceil + 1$$

$$7 \leq \text{height} \leq 11$$

(4) Assume that the B+ tree contains 1000 index items, please estimate the size (i.e. the number of nodes) of the B+ tree.

二叉树 1000 + 500 +250+ 125+ 62+31 +15+7+3+1
1750+125+62+57
1994

B+树 500 + 167+56+19+7+3+1
753

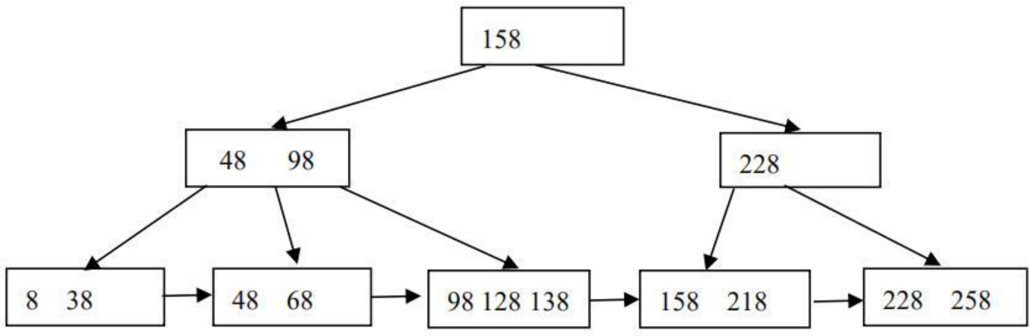
$$\text{size} \geq \left\lceil \frac{1000}{2} \right\rceil + \left\lceil \left\lceil \frac{1000}{2} \right\rceil / 3 \right\rceil + \left\lceil \left\lceil \left\lceil \frac{1000}{2} \right\rceil / 3 \right\rceil / 3 \right\rceil + \cdots + 1 = 755$$

$$\text{size} \leq 1000 + \lceil 1000/2 \rceil + \lceil \lceil 1000/2 \rceil / 2 \rceil + \cdots + 1 = 2001$$

$$755 \leq \text{size} \leq 2001$$

## Problem 5: B+ -Tree (12 points, 4 points per part)

For the following B+-tree (n=4 )：



(1) Draw the B+ -tree after inserting four entries 28,168, 58,18 sequentially.
(2) Draw the B+ -tree after deleting four entries 8, 38, 98,128 from the original B+-tree sequentially.
(3) Assuming (a) there are 3 blocks in main memory buffer for B+-tree operation, at first these blocks are empty. (b) Least Recently Used (LRU strategy) is used for buffer replacement. (c) each node of B+-tree occupies a block.
Please count the number of B+-tree blocks transferred to buffer in order to complete the operation in 1).

**Problem 2.** Consider the following relational schema and SQL query:

**product(pid: char(10), name: char(20), producer: char(20), price: integer)**
**customer(cid: integer, name: char(20), age: integer; city: char(20))**
**order(cid: integer, pid: char(10))**

**select customer.name, product.name**
**from customer, order, product**
**where customer.cid= order.cid and product.pid = order.pid**
**customer.city ='Hangzhou' and product.price>=200;**

(1) Identify a relational algebra tree (or a relational algebra expression if you prefer) that reflects the order of operations that a decent query optimizer would choose.

$$\prod_{\text{customer.name, product.name}} (\sigma_{\text{city='Hangzhou'}}(\text{customer}) \bowtie \text{Order} \bowtie \sigma_{\text{price}\geq 200}(\text{product}))$$

(2) What indexes might be of help in processing this query? Explain briefly.
Since the pid (in product and order) and cid (in customer and order) is crux of the join, it's helpful to create B+tree indexes in these attributes. (Different answer is OK if it makes sense.)

**12.3** Let relations $r_1(A, B, C)$ and $r_2(C, D, E)$ have the following properties: $r_1$ has 20,000 tuples, $r_2$ has 45,000 tuples, 25 tuples of $r_1$ fit on one block, and 30 tuples of $r_2$ fit on one block. Estimate the number of block transfers and seeks required, using each of the following join strategies for

$r_1 \bowtie r_2$:

a. Nested-loop join.

b. Block nested-loop join.

c. Merge join.

d. Hash join.

**Answer:**

$r_1$ needs 800 blocks, and $r_2$ needs 1500 blocks. Let us assume $M$ pages of memory. If $M > 800$, the join can easily be done in 1500 + 800 disk accesses, using even plain nested-loop join. So we consider only the case where $M \leq 800$ pages.

a. Nested-loop join:
Using $r_1$ as the outer relation we need $20000 * 1500 + 800 = 30,000,800$ disk accesses, if $r_2$ is the outer relation we need $45000 * 800 + 1500 = 36,001,500$ disk accesses.

b. Block nested-loop join:
If $r_1$ is the outer relation, we need $\lceil \frac{800}{M-1} \rceil * 1500 + 800$ disk accesses, if $r_2$ is the outer relation we need $\lceil \frac{1500}{M-1} \rceil * 800 + 1500$ disk accesses.

c. Merge-join:
Assuming that $r_1$ and $r_2$ are not initially sorted on the join key, the total sorting cost inclusive of the output is $B_s = 1500(2\lceil log_{M-1}(1500/M)\rceil +$

$2) + 800(2\lceil log_{M-1}(800/M)\rceil + 2)$ disk accesses. Assuming all tuples with the same value for the join attributes fit in memory, the total cost is $B_s + 1500 + 800$ disk accesses.

d. Hash-join:
We assume no overflow occurs. Since $r_1$ is smaller, we use it as the build relation and $r_2$ as the probe relation. If $M > 800/M$, i.e. no need for recursive partitioning, then the cost is $3(1500+800) = 6900$ disk accesses, else the cost is $2(1500 + 800)\lceil log_{M-1}(800) - 1\rceil + 1500 + 800$ disk accesses.

**12.13** Design a variant of the hybrid merge-join algorithm for the case where both relations are not physically sorted, but both have a sorted secondary index on the join attributes.

**Answer:** We merge the leaf entries of the first sorted secondary index with the leaf entries of the second sorted secondary index. The result file contains pairs of addresses, the first address in each pair pointing to a tuple in the first relation, and the second address pointing to a tuple inthe second relation. This result file is first sorted on the first relation's addresses. The relation is then scanned in physical storage order, and addresses in the result file are replaced by the actual tuple values. Then the result file is sorted on the second relation's addresses, allowing a scan of the second relation in physical storage order to complete the join.

**12.14** Estimate the number of block transfers and seeks required by your solution to Exercise 12.13 for $r_1 \bowtie r_2$, where $r_1$ and $r_2$ are as defined in Practice Exercise 12.3.

**Answer:** $r_1$ occupies 800 blocks, and $r_2$ occupies 1500 blocks. Let there be $n$ pointers per index leaf block (we assume that both the indices have leaf blocks and pointers of equal sizes). Let us assume $M$ pages of memory, $M < 800$. $r_1$'s index will need $B_1 = \lceil \frac{20000}{n} \rceil$ leaf blocks, and $r_2$'s index will need $B_2 = \lceil \frac{45000}{n} \rceil$ leaf blocks. Therefore the merge join will need $B_3 = B_1 + B_2$ accesses, without output. The number of output tuples is estimated as $n_o = \lceil \frac{20000 * 45000}{max(V(C,r_1), V(C,r_2))} \rceil$. Each output tuple will need two pointers, so the number of blocks of join output will be $B_{o1} = \lceil \frac{n_o}{n/2} \rceil$. Hence the join needs $B_j = B_3 + B_{o1}$ disk block accesses.

Now we have to replace the pointers by actual tuples. For the first sorting, $B_{s1} = B_{o1}(2\lceil log_{M-1}(B_{o1}/M) \rceil + 2)$ disk accesses are needed, including the writing of output to disk. The number of blocks of $r_1$ which have to be accessed in order to replace the pointers with tuple values is $min(800, n_o)$. Let $n_1$ pairs of the form ($r_1$ *tuple, pointer to* $r_2$) fit in one disk block. Therefore the intermediate result after replacing the $r_1$ pointers will occupy $B_{o2} = \lceil (n_o/n_1) \rceil$ blocks. Hence the first pass of replacing the $r_1$-pointers will cost $B_f = B_{s1} + B_{o1} + min(800, n_o) + B_{o2}$ disk accesses.

The second pass for replacing the $r_2$-pointers has a similar analysis. Let $n_2$ tuples of the final join fit in one block. Then the second pass of replacing the $r_2$-pointers will cost $B_s = B_{s2} + B_{o2} + min(1500, n_o)$ disk accesses, where $B_{s2} = B_{o2}(2\lceil log_{M-1}(B_{o2}/M) \rceil + 2)$.

Hence the total number of disk accesses for the join is $B_j + B_f + B_s$, and the number of pages of output is $\lceil n_o/n_2 \rceil$.

## Problem 7: Concurrency Control

Consider following three transactions:

T1: read(A)
    Read(B)
    Write(B)

T2: read(B)
    Read(A)
    Write(A)

T3: read(A)
    Write(A)

1) For the following schedule, please draw the precedence graph, and explain whether it is conflict serializable.

| T1 | T2 | T3 |
|---|---|---|
| Read(A) | | |
| Read(B) | | |
| | | Read(A) |
| | | Write(A) |
| | Read(B) | |
| | Read(A) | |
| Write(B) | | |
| | Write(A) | |

2) For the following schedule, please explain whether it is cascadeless.

| T1 | T2 | T3 |
|---|---|---|
| Read(A) | | |
| Read(B) | | |
| Write(B) | | |
| | Read(B) | |
| | Read(A) | |
| | Write(A) | |
| | | Read(A) |
| Abort | | |

3) Please explain whether the two-phase locking protocol can be used to implement the schedule in 1).