COMP2411

Assignment 2

Name:Zhu JinShun

ID:22101071d

Question A

1) Block anchor is the first record of each block in a data file. By using block anchors, a primary index optimizes the search process by providing a way to locate the blocks that potentially contain the desired records quickly. Since for primary index , it has one index entry for each block in the data file. Then each index entry has the key field value for first record in each block which is the block anchor of each block.

2) From comparing primary indexes with secondary indexes.

Primary indexes are unique as they are most commonly defined on the physical order of the records in an ordered data file while their index entry field is the field data that is order on so we can have at most one primary index on a file.
Secondary indexes allow for more flexible querying and a wider range of search criteria. It enables efficient access to records based on different columns, even if they are not part of the primary or clustering key. With secondary indexes defined by attributes of a file, it can have multiple indexes existing in one file if there are multiple attributes.

3) To choose the most efficient index we can compare the number of block accesses for each index

Each block can hold B = 600 bytes, and each record occupies 100 bytes, so each block can hold B/R = 600/100 = 6 records.
Number of records per block = 4,000 / 6 = 666.67 = 667 blocks

Clustering index:

Number of entries=10
Category Id record=4000/10=400 records
Index bfr = 400/(6+4) = 60 entries/ block
Number of index blocks=ceiling(10/60)=1 block
The estimated number of block accesses =1 + ceiling(400/6) = 68 block accesses.

Secondary index:

Index entry size=6+4=10bytes
Index bfr=floor(B/E)=floor(600/10)=60 entries/block

Index entries=200 entries
Index blocks =ceiling(200/60)=4 blocks
Number of records=4000/200=20 records
The estimated number of block accesses for the secondary index is ceiling(log24)+20+1=23 block accesses.

Bitmap index

(A bitmap index on both CategoryID and DirectoryID fields (1 byte = 8 bits; each bitmap is stored as a fixed-length record))

Two fields:
Category ID
Space needed =10*4000=40000bits=5000bytes
Director ID
Space needed=200*4000=800000bits=100000bytes
Index bfr=floor(600/[(200+10)/8]))=22 entries/block
Index blocks=ceiling(4000/22)=182 blocks
Total number of accesses=182+4=186 accesses
Comparing:
Clustering index=68 block accesses
Secondary index=23 block accesses
Bitmap index=186 block accesses

We can conclude that Secondary index < Clustering index< Bitmap index

So, Secondary index is the most efficient for answering this query.

Question B

1)

1st tuple:

Block 1 of relation R:
R(A, B): (7, x)
Block 1 of relation S:
S(A, C): (8, 1)
A=7 and A=8 do not match. No tuples are produced.
Block 2 of relation S:
S(A,C):(4,3)
A=7 and A=4 do not match. No tuples are produced.

Block 3 of relation S:
S(A,C):(2,6)
A=7 and A=2 do not match. No tuples are produced.
Block 4 of relation S:
S(A,C):(1,5)
A=7 and A=1 do not match. No tuples are produced.
Block 5 of relation S:
S(A,C):(3,7)
A=7 and A=3 do not match. No tuples are produced.
Block 6 of relation S:
S(A,C):(5,1)
A=7 and A=5 do not match. No tuples are produced.
Block 7 of relation S:
S(A,C):(7,8)
A=7 and A=7 match, so the tuple (7, x, 8) is produced.

The first tuple produced is (7, x, 8).

2nd tuple:

Block 2 of relation R:
R(A, B): (2, z)
Block 1 of relation S:
S(A, C): (8, 1)
A=2 and A=8 do not match. No tuples are produced.
Block 2 of relation S:
S(A,C):(4,3)
A=2 and A=4 do not match. No tuples are produced.
Block 3 of relation S:
S(A,C):(2,6)
A=2 and A=2 match, so the tuple (2, z, 6) is produced.

The second tuple produced is (2, z, 6).

3rd tuple:

Block 3 of relation R:
R(A, B): (9, y)
Block 1 of relation S:
S(A, C): (8, 1)
A=9 and A=8 do not match. No tuples are produced.
Block 2 of relation S:
S(A,C):(4,3)
A=9 and A=4 do not match. No tuples are produced.
Block 3 of relation S:
S(A,C):(2,6)

A=9 and A=2 do not match. No tuples are produced.
Block 4 of relation S:
S(A,C):(1,5)
A=9 and A=1 do not match. No tuples are produced.
Block 5 of relation S:
S(A,C):(3,7)
A=9 and A=3 do not match. No tuples are produced.
Block 6 of relation S:
S(A,C):(5,1)
A=7 and A=5 do not match. No tuples are produced.
Block 7 of relation S:
S(A,C):(7,8)
A=9 and A=7 do not match. No tuples are produced.
Block 7 of relation S:
S(A,C):(9,2)
A=9 and A=9 match. So the tuple (9, y, 2) is produced.

The third tuple produced is (9, y, 2).

4th tuple:

Block 4 of relation R:
R(A, B): (8, y)
Block 1 of relation S:
S(A, C): (8, 1)
A=8 and A=8 match, so the tuple (8, y,1) is produced.

The fourth tuple produced is (8, y, 1).

2)

For Relation R:
Record size = 20 bytes
Number of records = 100000
Total size of Relation R = Record size * Number of records = 20 bytes/record * 100000 records = 2000000 bytes
Number of blocks for Relation R = ceiling(Total size of Relation R / Block size) = 2000000 bytes / 1000 bytes/block = 2000 blocks

For Relation S:
Record size = 50 bytes
Number of records = 50000
Total size of Relation S = Record size * Number of records = 50 bytes/record * 50000 records = 2500000 bytes
Number of blocks for Relation S = ceiling (Total size of Relation S / Block size) = 2500000 bytes / 1000 bytes/block = 2500 blocks

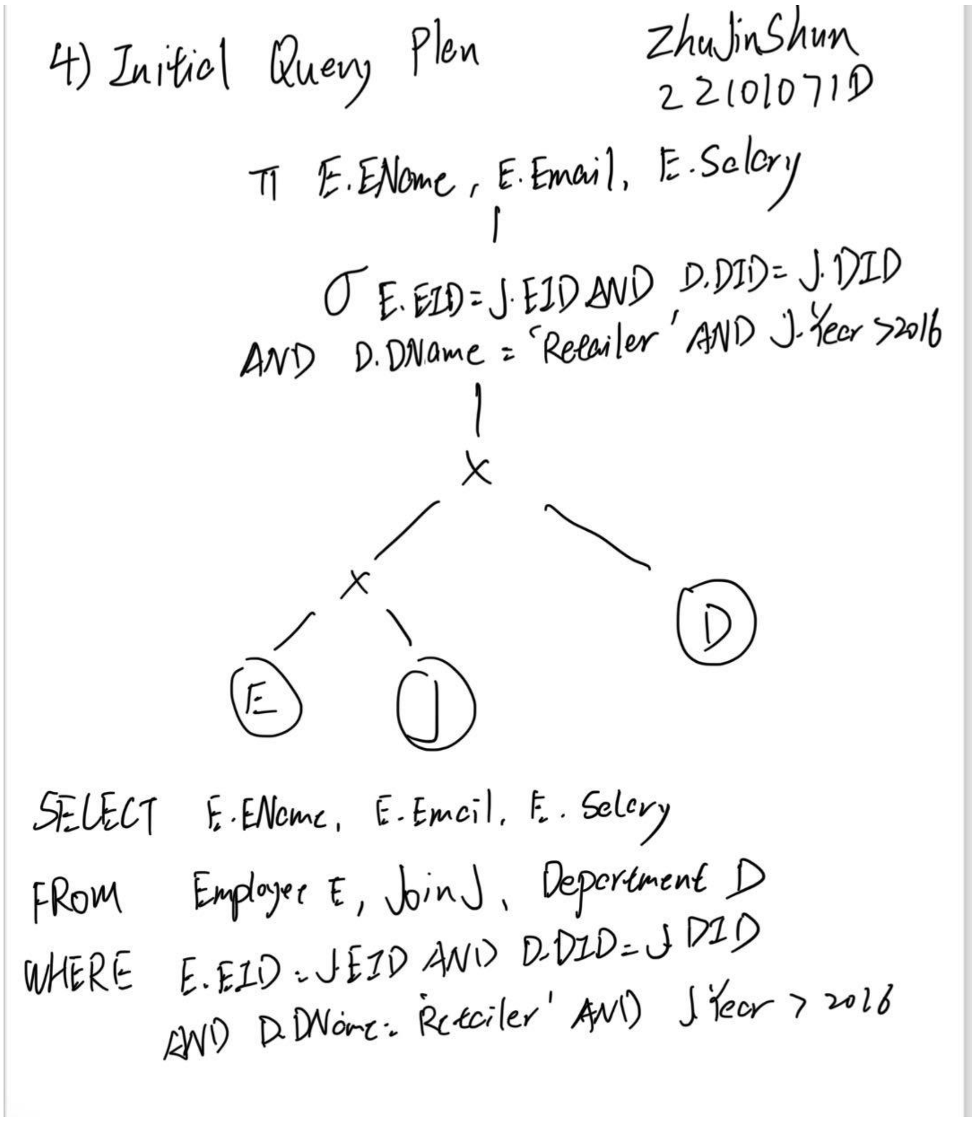
Block accesses needed for Relation R=2000
Block accesses needed for Relation S=2500

Total estimated accesses for R * S using page-oriented nested-loop join= 2000*2500+2000=5002000 block accesses

3)

No, because Total number of block accesses using S*R=2500*2000+2500=5002500 block accesses.
5002500>5002000, as S* R results in more block accesses than R*S, so changing the join order will not improve the efficiency of R*S

4)

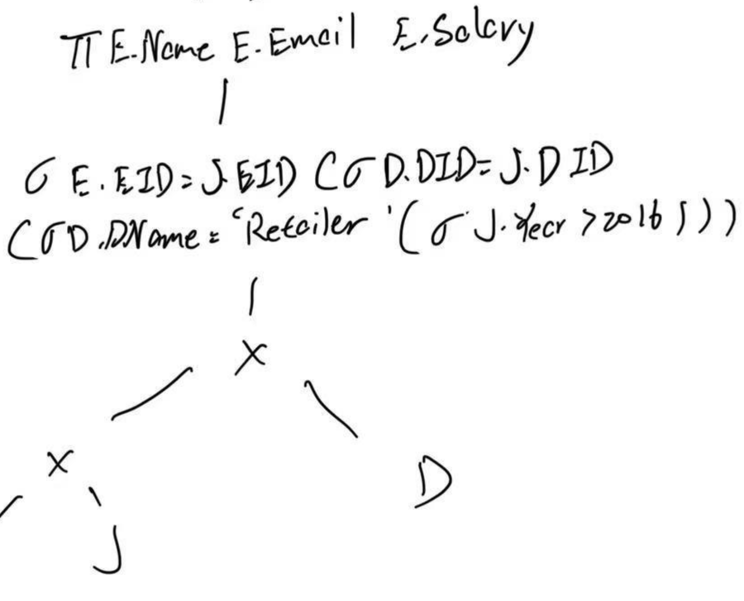

4) Initial Query Plen
ZhuJinShun
22101071D
π E.EName, E.Email, E.Selcry
σ E.EID=J.EID AND D.DID=J.DID AND D.DName='Retailer' AND J.Year>2016
×
×
E
J
D

```
SELECT E.EName, E.Emcil, E. Selery
FROM   Employee E, Join J, Depertment D
WHERE  E.EID=J.EID AND D.DID=J.DID
       AND D.DName='Retciler' AND J.Year > 2016
```

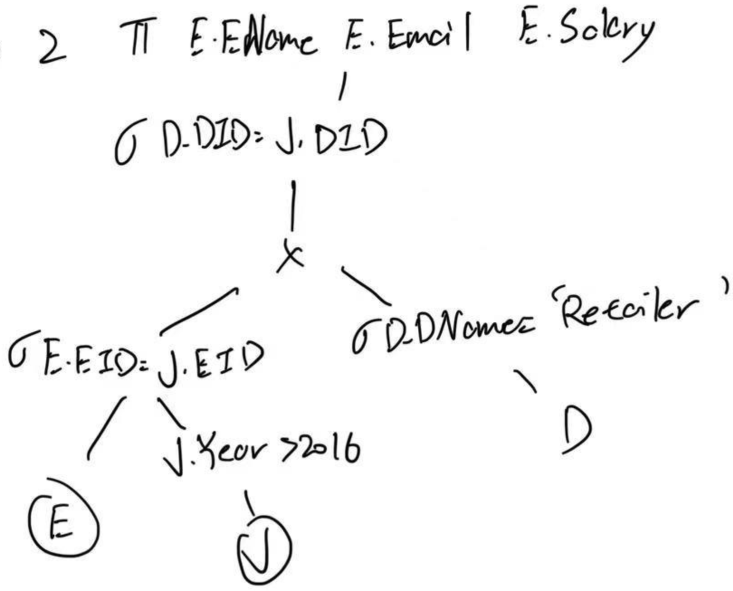
5)

5) Most efficient query tree

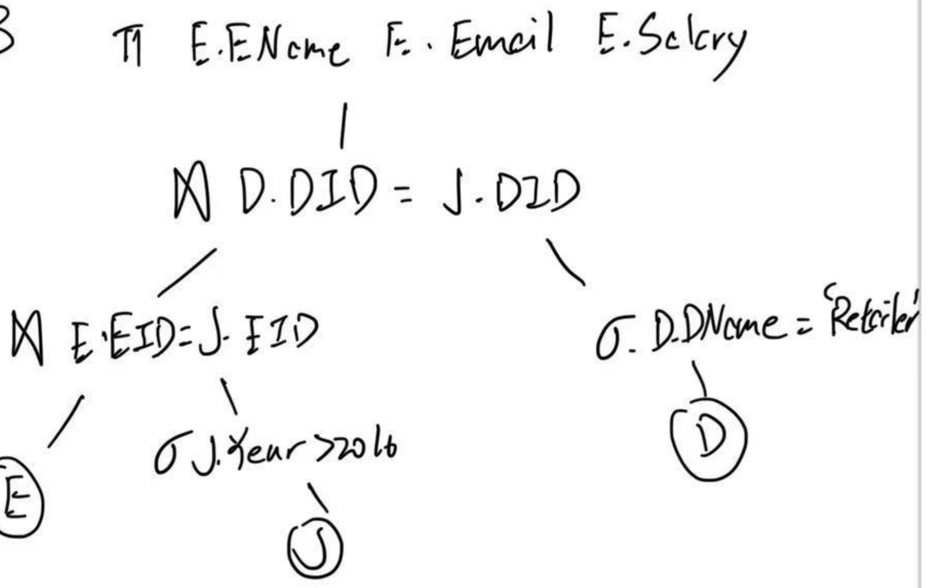
Step 1


π E.Name E.Email E.Salary
σ E.EID = J.EID (σ D.DID = J.DID (σ D.DName = 'Retailer' (σ J.Year > 2016)))
×
×
E
J
D

Step 2

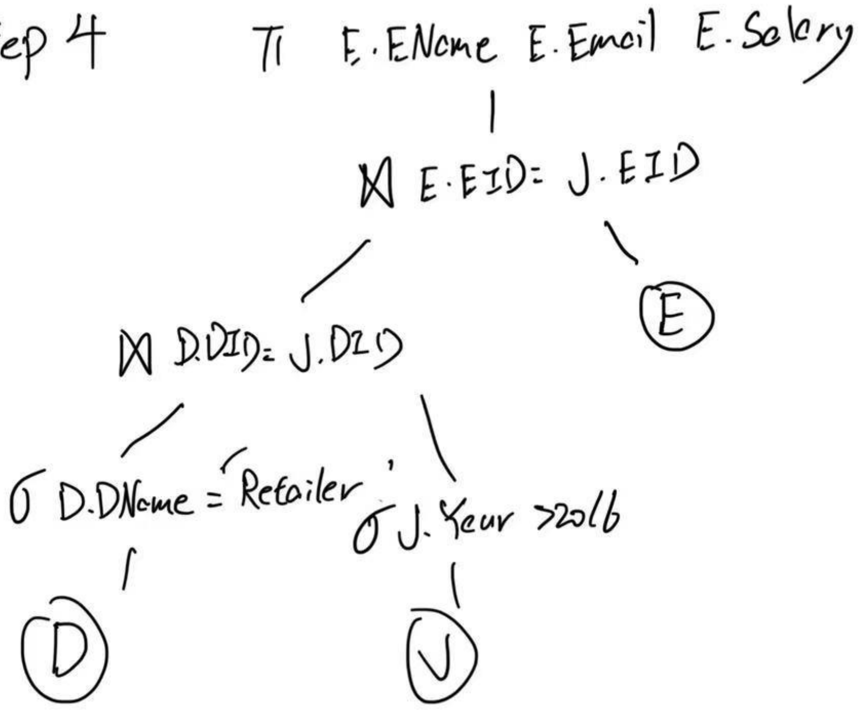


Step 3

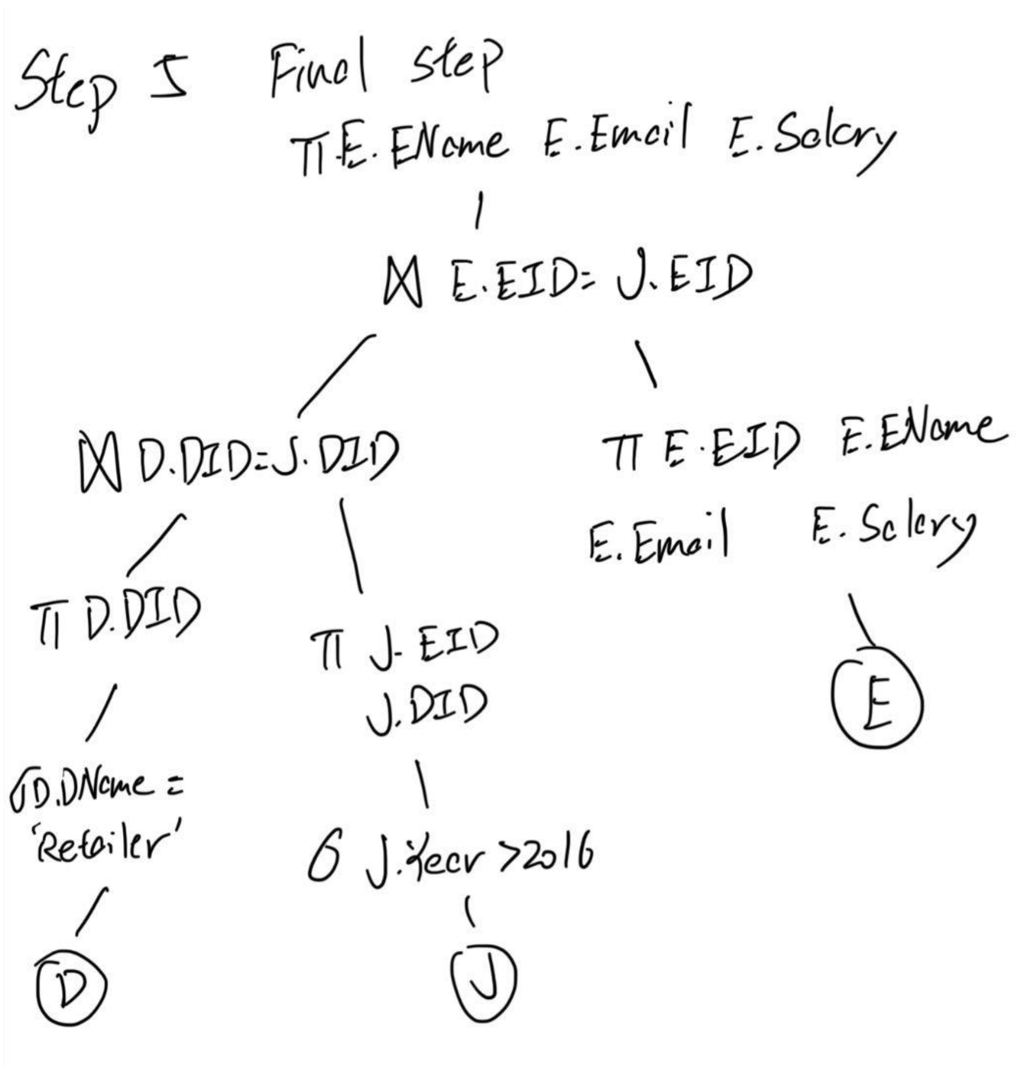


π E.EName E.Email E.Salary
⋈ D.DID = J.DID
⋈ E.EID=J.EID
σ. D.DName = 'Retailer'
E
σ J.Year >2016
J
D

Step 4



π E.EName E.Email E.Salary
⋈ E.EID= J.EID
⋈ D.DID= J.DID
E
σ D.DName = 'Retailer'
σ J.Year >2016
D
J



Question C

1)

Basic 2PL:

Protocol Description:
In Basic 2PL, a transaction is divided into two phases: the growing phase and the shrinking phase. During the growing phase, a transaction can acquire locks on data items (read or write). Once a transaction releases a lock during the shrinking phase, it cannot acquire any new locks.

Allows Schedule S: Basic 2PL protocol does allow schedule S.

Explanation:
Schedule S is compliant with the Basic 2PL protocol due to its adherence to the fundamental principles of the protocol. In schedule S, transaction T1 follows the protocol by first acquiring a write lock on X (W1(X)), then proceeding to read Y (R1(Y)), and finally

committing (C1). Similarly, transaction T2 adheres to the protocol by acquiring a read lock on Y (R2(Y)), reading X (R2(X)), and ultimately committing (C2). Notably, all lock acquisitions and releases occur in a valid sequence without any conflicting actions. As a result, schedule S is indeed permitted under the Basic 2PL protocol.

Conservative 2PL:

Protocol Description:
In Conservative 2PL, before a transaction begins, it first acquires all the locks it will need throughout its execution. This means the transaction takes a conservative approach by obtaining all the required locks upfront, ensuring that no conflicts occur during the execution.

Allows Schedule S: Conservative 2PL protocol does not allow schedule S.

Explanation:
In schedule S, transaction T1 performs a write operation (W1(X)) and transaction T2 performs a read operation (R2(Y)). According to Conservative 2PL, a transaction must acquire all locks it will need upfront. In this case, T1 needs a lock on X, and T2 needs a lock on Y. However, T2 cannot acquire the lock on Y because T1 already holds a lock on Y. This creates a potential deadlock situation, as both transactions are waiting for a lock that the other transaction possesses. Therefore, schedule S violates the Conservative 2PL protocol.

Strict 2PL:

Protocol Description:
In Strict 2PL, a transaction holds all its locks until it commits or aborts. It ensures that no other transaction can read or write a data item that is locked by an uncommitted transaction.

Allows Schedule S: Strict 2PL protocol does not allow schedule S.

Explanation:
Schedule S is deemed incompatible with the Strict 2PL protocol due to its violation of the key rule of maintaining locks until the completion of a transaction. In schedule S, transaction T1 initially acquires a write lock on X (W1(X)), proceeds to read Y (R1(Y)), and subsequently commits (C1). However, T1 inappropriately releases the lock on X prior to completing its execution. Additionally, transaction T2 acquires a read lock on Y (R2(Y)) and then reads X (R2(X)). As T1 fails to uphold the prerequisite of retaining the lock on X until completion, it infringes upon the strictness mandate imposed by the Strict 2PL protocol. Consequently, schedule S is considered unsuitable under the constraints of

Strict 2PL.

2)

What type of anomaly occurs in this schedule S?

Answer:

The type of anomaly that occurs in this schedule S is a "Dirty Read". Because both T1 and T2 are accessing and modifying the same data item X. T1 reads X, modifies it, and then T2 reads X and modifies it as well. T2's modification effectively overwrites the changes made by T1, which would result in the dirty read anomaly. Which is in S where R1(X); W1(X); (read and write X once) R2(X) W2(X); (read and write X second time).

For each of T1 and T2, insert all the lock and unlock operations to make the transaction satisfy the strict 2PL protocol.

Answer:
T1:
Lock(X)
R(X)
W(X)
Lock(Y)
R(Y)
W(Y)
Unlock(Y)
Unlock(X)

T2:
Lock(X)
R(X)
W(X)
Unlock(X)

Show that modifying the schedule according to strict 2PL can prevent the anomaly

Answer:
By modifying the schedule according to strict 2PL, we ensure that a transaction holds all its locks until it commits or aborts. In the modified schedule, T1 acquires a lock on X before reading and writing it. T2 also acquires a lock on X before reading and writing it. This ensures that both transactions have exclusive access to X during their respective operations, preventing the lost update anomaly.

The modified schedule would look like this:
S: Lock1(X); R1(X); W1(X); Lock2(X); R1(Y); W2(X); C2; W1(Y); A1; Unlock1(X)