Database Development and Design (CPT201)

Tutorial 7&8

Dr. Wei Wang

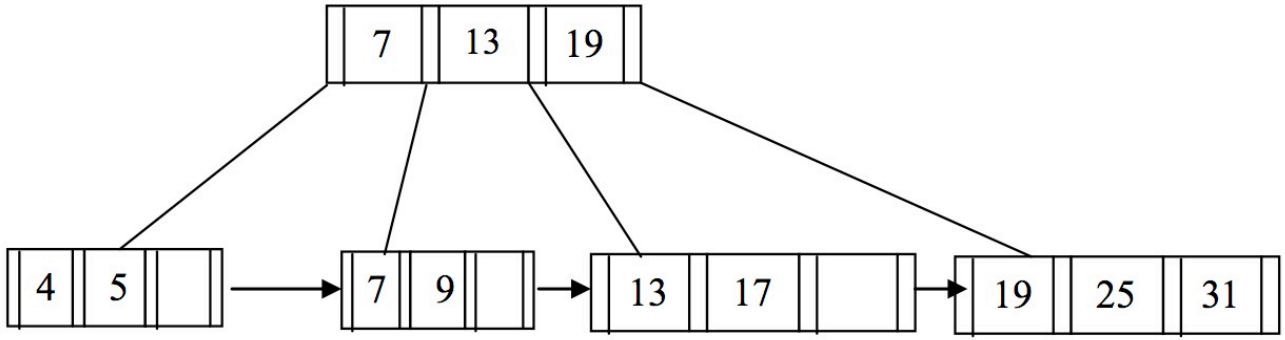
Department of Computing

Q1

- A) Relation *R* has 1,000 fixed length, fixed format records. One disk block can hold 5 records and no record spans over one block. Suppose that B+ tree is used to index the tuples based on the candidate key. The number of pointers of the B+ tree node, N=10. Assume a dense index has one index entry for each tuple, and a sparse index has one index entry for each block.
  - What is the maximum height for the dense index?
  - What is the maximum height for the sparse index?

Q1

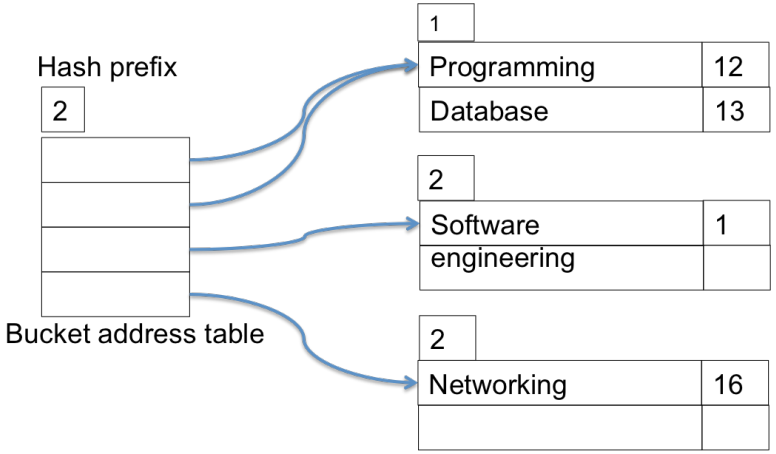
- B) Consider the following B+ tree. The number of pointers that fits in one node is N=4. Each operation must be performed based on the previous ones. Draw the trees after:
  - Insert 17
  - Insert 33
  - Insert 11
  - Delete 25



Q1

- C) An extendable hash index is created on the *title* attribute for the relation *module* as shown below. Based on the hash values on the titles in the table, draw the hash index after insertion of the following tuples (tuple is of the form "*sID, title*"): (1) "16, Machine Leaning"; (2) "20, Machine Leaning".

Search key	Hash value
Programming	0001 1001 0000 1111
Database	0101 1001 0101 1100
Operating system	0011 1011 0000 1100
Machine learning	1100 1001 0101 1010
Networking	1110 0001 0101 1110
Software engineering	1010 0101 1111 0010

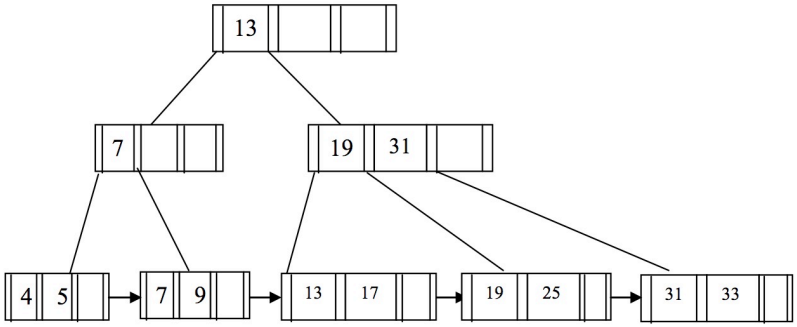


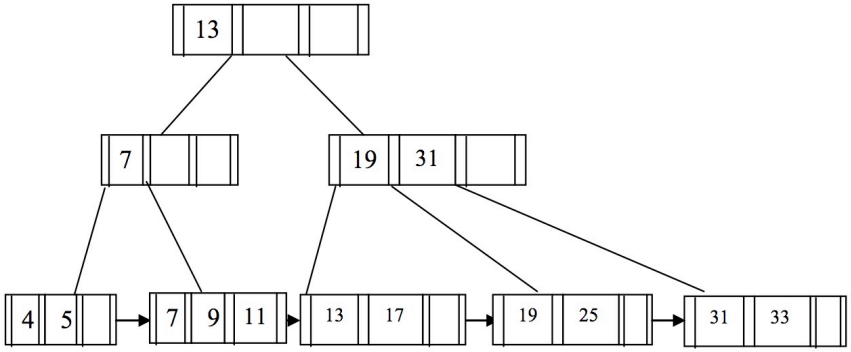
Q1

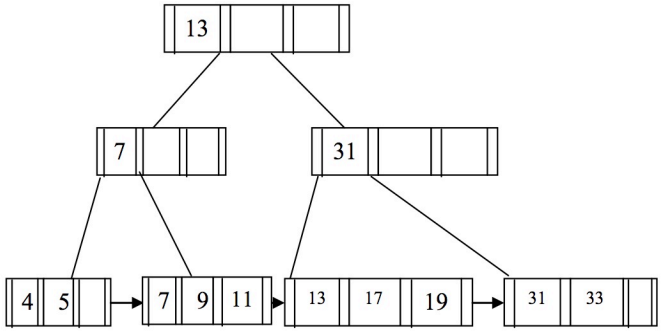
- D) Suppose that a relation called *student* holds 25,000 tuples, which are stored as fixed length and fixed format records. The length of each tuple is 350 bytes. The key attribute, *student_ID*, occupies 10 bytes and another attribute *address* occupies 50 bytes. The records are sequentially ordered by *student_ID* and stored in a number of blocks. Each block has the size of 4,096 bytes (i.e., 4 Kilobytes). Assume that a complete record or an index entry must be stored in one block.
  - How many blocks are needed to store the relation *student*?
  - Consider creating a primary index on the *student_ID* attribute. Each index entry contains a search key and a 10-byte long pointer. Suppose the primary index is sparse (i.e., one index entry for one block), compute the number of blocks needed to store the index.

Q1 Solutions

- A)
  - for dense index, the maximum height is $\lceil \log_{(10/2)} 1{,}000 \rceil = 5$.
  - for sparse index, the maximum height is $\lceil \log_{(10/2)} (1{,}000/5) \rceil = 4$.
- B)

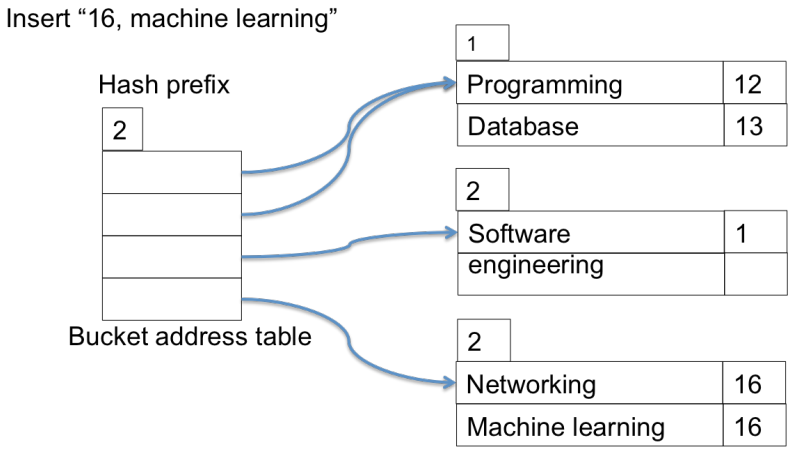


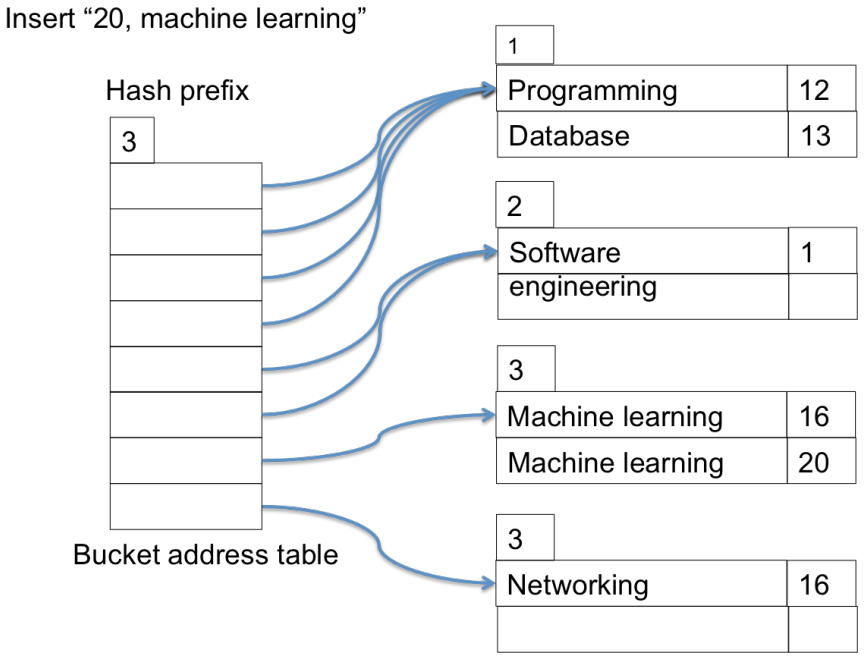




Q1 Solutions

- C)





Q1 Solutions

- D)
  - Each tuple of student is 350 bytes. Each block at most holds ⌊4096 bytes/350 bytes⌋ = 11 tuples (where ⌊ ⌋ indicates round down). There are 25,000 tuples, so ⌈25000 tuples/11 tuples per block⌉ = 2,273 blocks required (where ⌈ ⌉ indicates round up).
  - Each index entry is 10 bytes for the key plus 10 bytes for the pointer (20 bytes in total). Each block at most can store ⌊4096 bytes/20 bytes⌋ = 204 index entries. There are 2,273 blocks in the *student* relation, so ⌈2,273/204⌉ = 12 blocks are needed.

Q2

- Consider the following two relations and their catalogue information:
  - *account(account_Number, customer_Name, balance, branch_ID)*
  - *branch(branch_ID, branch_Name, branch_City, postcode)*
- The "*account_Number*" is the key for the *account* relation, and the "*branch_Name*" is the key for the *branch* relation. The *account* relation contains 300,000 records stored in 60,000 blocks, and the *branch* relation contains 500 records stored in 50 blocks. Assume that both relations are sequentially stored by the key attributes.
- Answer the following questions:
- A) Suppose that the linear search algorithm is used to evaluate the selection $\delta_{balance>5,000}$ in the *account* relation, how many block transfers are needed? How many seeks are needed?

Q2

- B) Suppose that none of the relations can fit in memory, and the nested loop join algorithm is used to evaluate "*account branch*". Which relation should be used as outer relation? How many block transfers are needed? How many seeks are needed?
- C) Suppose that the external sort merge algorithm is used to sort the *account* relation on the *account_Number* attribute. Assume that the memory size $M=30$ and the buffer for reading and writing $b_b=2$. How many block transfers are needed? How many seeks are needed?
- D) Suppose that the hash join algorithm is used to evaluate "*account branch*", the number of partitions, $n_h=70$, and the size of the buffer for reading and writing, $b_b=2$. How many block transfers are needed? How many seeks are needed?

Q2 Solutions

- A) 60,000 block transfers and 1 seek.
- B) The smaller relation branch should be used as the outer relation (denoted as *r*). Number of block transfers: $n_r * b_s + b_r$ = 500*60,000+50 = 30,000,050; Number of seeks: $b_r + n_r$ = 50+500 = 550
- C) Number of block transfers: $b_r (2 \lceil \log_{\lfloor M/bb \rfloor - 1}(b_r / M) \rceil + 1)$ = 60,000(2*$\log_{14}$(60,000/30))+1) = 420,000; Number of seeks: $2 \lceil b_r / M \rceil + \lceil b_r / b_b \rceil (2 \lceil \log_{\lfloor M/bb \rfloor - 1}(b_r / M) \rceil - 1)$ = 2*(60,000/30) + 60,000/2(2*$\log_{14}$(60,000/30))-1) = 4,000+30,000*5 = 154,000
- D) Number of block transfers: 3(*br+bs)+4*nh* = *3(*50+60,000)+4*70 = 180,430; Number of seeks: 2(⌈*br/bb*⌉+⌈*bs/bb*⌉*)+2nh* = 50+60,000+2*70 = 60,190

Q3

- Consider the following three relations and their catalog information.
  - *staff(ID, name, email, department)*
  - *teaches(ID, module_Code)*
  - *module(module_Code, module_Title, level)*
- where *staff.ID* is the key for *staff*, and *module_Code* is the key for *module; teaches.ID* and *teaches.module_Code* are the foreign keys referencing *staff* and *module*, respectively.
  - the number of records in *staff*, n_{staff} = 1,000;
  - the number of blocks in *staff*, b_{staff} =200
  - the number of distinct values for the attribute *department* in the *staff* relation, *V(department, staff) = 50*
  - index: a three-level primary B^+-tree index (height=3) on the *ID* attribute of *teaches* relation.
  - the number of records in *teaches*, $n_{teaches}$ = 3,000
  - the number of blocks in *teaches*, $b_{teaches}$ = 100
  - the number of records in *module*, n_{module} = 1,500
  - the number of blocks in *module*, b_{module} = 150

Q3

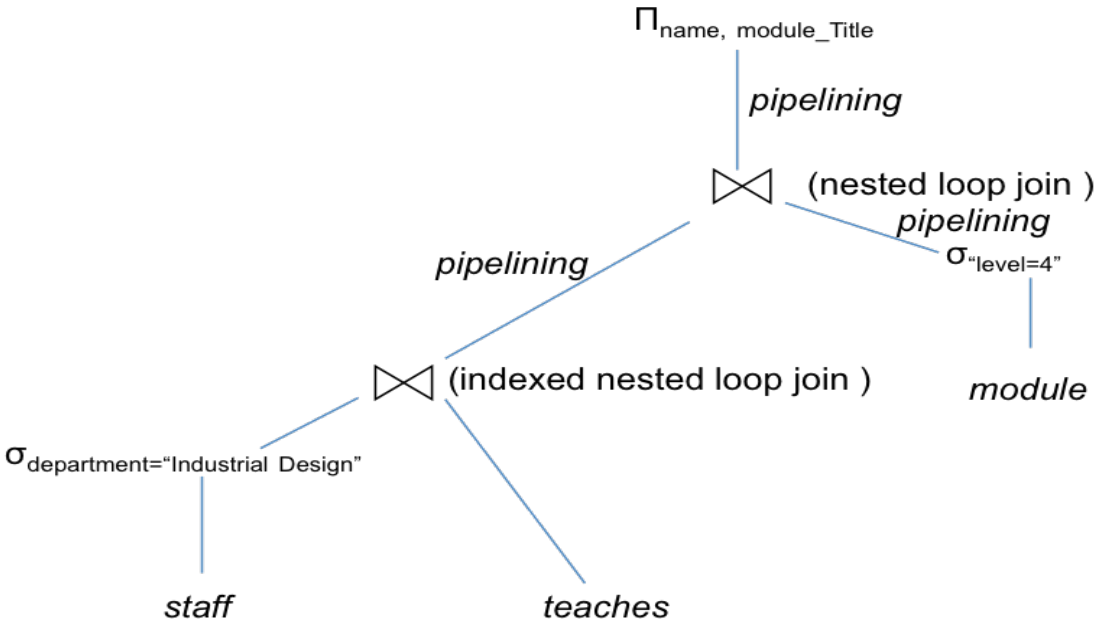
- Consider the following relational algebra expression and answer the questions below.
  - $\Pi_{name,\ module\_Title}(\sigma_{department="Industrial\ Design"\ \wedge\ level="4"}(staff \bowtie teaches \bowtie module)$
- A) One of the heuristic rules for query optimisation is to perform selection operations as early as possible. Write the equivalent algebra expression for the given expression based on this heuristic rule and the equivalence rules.
- B) Suppose that the join between relations *staff* and *teaches* is evaluated using the indexed nested loop join algorithm; the join between the result and the relation *module* is evaluated using the nested loop join algorithm. Also, assume that pipelining is used for selection, projection and nested loop join. Draw an annotated evaluation tree for the relational algebra expression obtained from Question 3.a).

Q3

- C) Based on the given catalog information and query evaluation tree, what is the estimated size of the selection $\sigma_{department="Industrial\ Design"}(staff)$? How many blocks are needed to store the results?
- D) Based on the given catalog information and query evaluation tree, what is the estimated size of the join "*staff* $\bowtie$ *teaches*" using the indexed nested loop join algorithm?
- E) Based on the results from Question 3.D), what is the estimated size of the nested loop join with the relation *module*? Justify your answer.

Q3 Solutions

- A) $\Pi_{name,\ module\_Title}((\sigma_{department="Industrial\ Design"}(staff) \bowtie teaches) \bowtie (\sigma_{level="4"}(module))$
- B)



Q3 Solutions

- C) As V(department, staff) = 50, there are 1,000 records in the staff relation. So the estimated size is 20. They may occupy 20/ (1,000/200)=4 blocks.
- D)
  - Method 1: As the join attribute *ID* is the foreign key for *teaches* relation, so the upper bound of the size is 3,000. V(department, staff) = 50, So the estimated size is 60.
  - Method 2: This can be estimated by using min(n_{staff}*$n_{teaches}$/V(A, *staff*), n_{staff}*$n_{teaches}$/V(A, *teaches*)). The join attribute ID is the key for staff relation, and the size of the selection is 20 from Question 3.a), so n_{staff}*$n_{teaches}$/V(ID, *staff*)=20*3,000/1,000 = 60. n_{staff}*$n_{teaches}$ /V(ID, *teaches*) is unknown. So the estimated size is 60.
- E) The estimated size of the indexed nested loop join is 60; the size of the module is 1,500. The join attribute *module_Code* is the foreign key referencing the *module* relation. One tuple in the result of "*staff* ⋈ *teaches*" can join with one tuple in the relation *module*. So the size of the nested loop join is 60.

Q4

- A) Draw the precedence diagram for the schedule below and determine if it is conflict serialisable.
- B) Is the schedule in Question 4.A recoverable" and why?

T1	T2	T3	T4
Read(x)			
Write(x)			
Read(y)			
	Read(x)		
		Write(y)	
	Read(z)		
			Read(y)
			Read(w)
			Write(w)
	Read(w)		

Q4

- C) Consider the following schedule.
  - *T1:write(X); T1:write(Y); T2:read(X); T2:write(Y); T2:read(Z); T1:write(Z); T1:commit; T3:read(Y); T2: commit; T3:write(Z); T4:read(Z); T3:commit; T4:abort.*
  - Draw the precedence diagram for the schedule and determine if it is conflict serialisable.
- D) Is the schedule in Question 4.C recoverable? Justify your answer.
- E) Is the schedule in Question 4.C cascadeless? Justify your answer.

Q4 Solutions

- A) It is conflict serialisable as the precedence diagram is acyclic.
- B) A recoverable schedule is one where, for each pair of transactions *Ti* and *Tj* such that *Tj* reads a data item previously written by *Ti*, the commit operation of *Ti* appears before the commit operation of *Tj*. It is not possible to determine if the schedule is recoverable or not as the commit operations of the transactions are not given.

Q4 Solutions

- C) It is not conflict serialisable as there is a cycle in the graph (between T1 and T2).
- D) The above schedule is recoverable.
  - T2 reads X which is written by T1 and T1 commits before T2 does.
  - T3 reads Y which is written by T2 and T2 commits before T3 does.
  - T4 reads Z which is written by T3 and T3 commits before T4 aborts.
- E) The above schedule is not cascadeless.
  - T2 reads X which is written by T1 but T1 commits after T2 reads X.
  - T3 reads Y which is written by T2 but T2 commits after T3 reads Y.
  - T4 aborts later so it is not considered.