

Homework 1

Problem 1 Bitmap Index

For the example table discussed in the class, provide the remaining three compressed bitmap values.

Key	Bitmap	Compressed
Toys	00011010	10110001
PCs	00100100	10101010
Pens	01000001	01110101
Suits	10000000	00

Steps for compression

Step 1: Get sequence of runs

- Scan bitmap from left to right.
- Count the number of 0 until you find 1
- Output Count
- Discard all bits has been scanned till now (0s and 1)
- Repeat (a) to (d) until no 1 left

Output after this step:

Keys	Runs
Toys	[3,0,1]
PCs	[2,2]
Pens	[1,5]
Suits	[0]

Step 2: Convert above output from base 10 to base 2

Output after this step:

Keys	Runs
Toys	[11,0,1]
PCs	[10,10]
Pens	[1,101]
Suits	[0]

Step 3: Build position indicators

For each binary number in above output table

- Use same number of bits as the binary number did
- Mark the last bit 0 (as a stop signal)
- Mark all other bits 1
- Output this binary number as position indicator

Output after this step:

Keys	Position Indicator	Runs
Toys	[10,0,0]	[11,0,1]
PCs	[10,10]	[10,10]
Pens	[0,110]	[1,101]
Suits	[0]	[0]

Step 4: Catenate position indicator with runs

(a) For each position indicator, connect it with corresponding runs

Keys	Position Indicator
Toys	[1011,00,01]
PCs	[1010,1010]
Pens	[01,110101]
Suits	[00]

(b) For each Keys, connect all its position indicator and output the final result

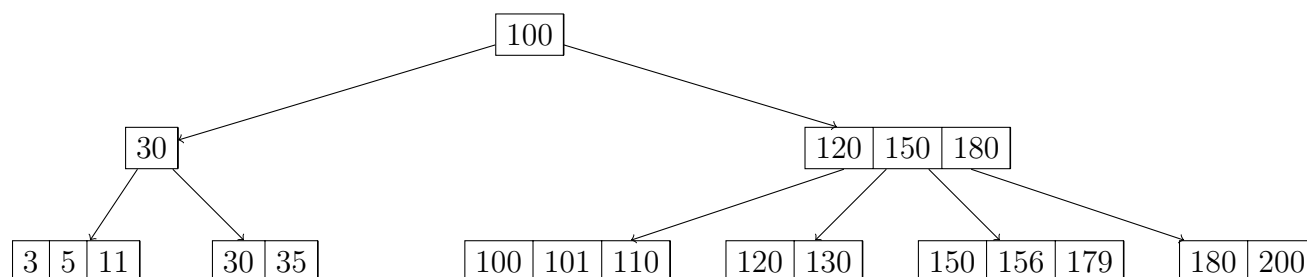
Key	Bitmap	Compressed
Toys	00011010	10110001
PCs	00100100	10101010
Pens	01000001	01110101
Suits	10000000	00

Problem 2 B+ Tree Index

Draw the tree that will result from the insertion of a record with key 115. **NOT JUST ANY TREE THAT HAS THESE DATA.**

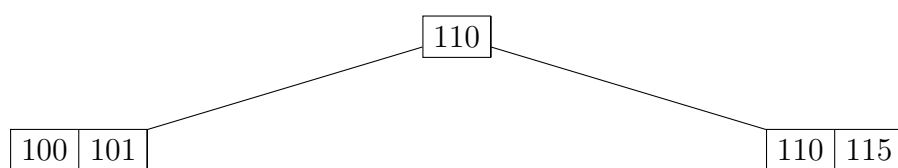
To save yourself from some drawing, do not redraw subtrees that are not modified. Just put the label of the subtree in the picture. For example, the subtree C stands for the node marked with C and all its children.

n=3



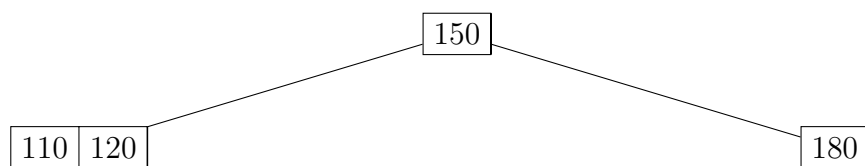
Step 1: Insert 115 into block F (100,101,110) and find there is no enough space for new element 115.

Step 2: Divide block F into 2 blocks

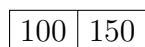


Step 3: Insert 110 into block C (120,150,180) and find there is no enough space for new element 110.

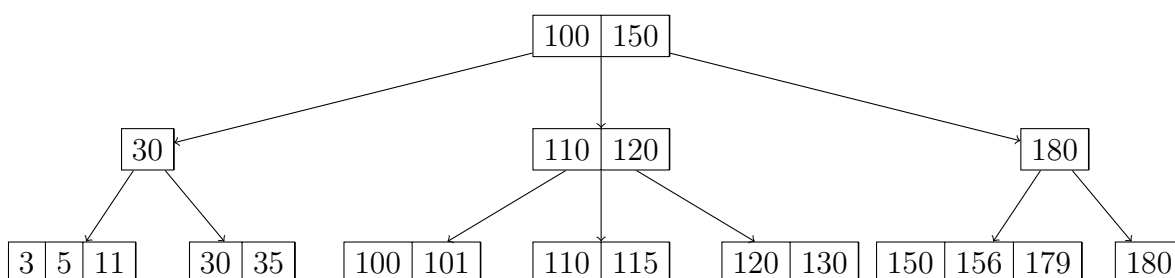
Step 4: Divide block C into 2 blocks. Since Block C is not leaves, we will not repeat key 150 twice.



Step 5: Insert 150 into Block A (100)

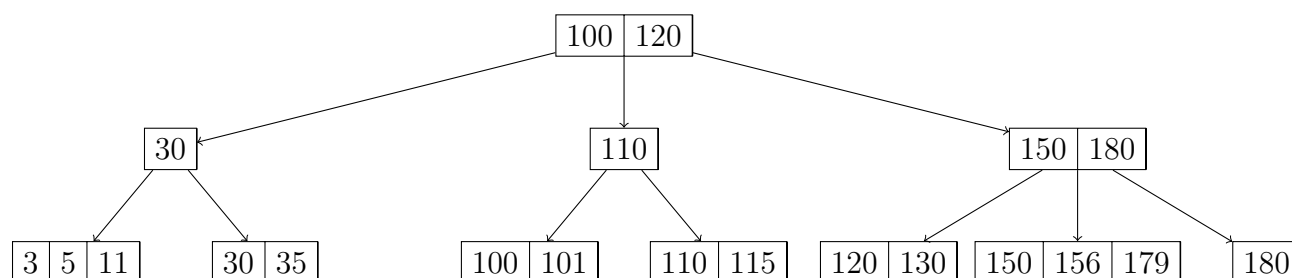


Step 6: Final result



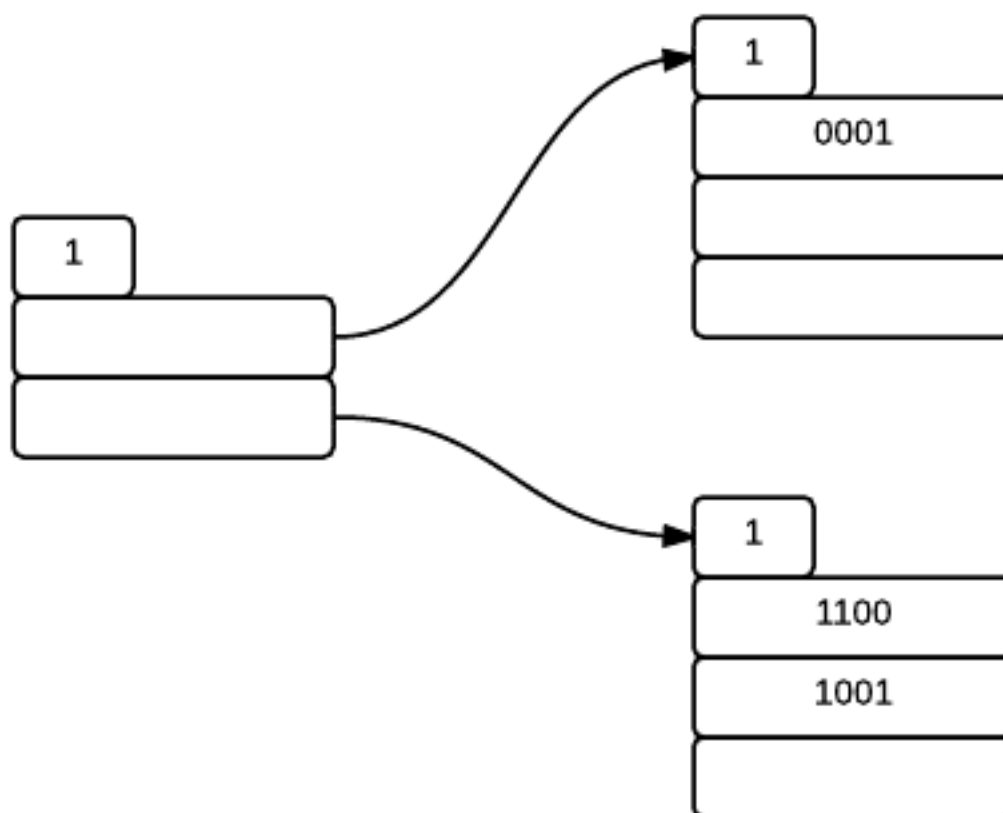
Step 7: Depend on which number you choose as key, there might be multiple results for this problem.

For example, another possible result will be:



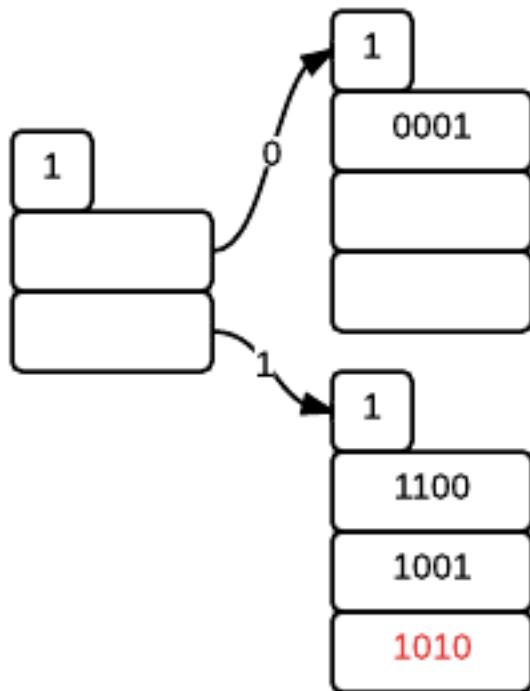
Problem 3 Extensible hash index

Consider an extensible hash index that uses 4 bits and each block of the index contains up to 3 entries. Initially the index indexes three keys with respective hash values 0001, 1100, 1001. As you see, we will use the starting bits in this exercise. As we did in the slides' example also, instead of showing the actual key values we show just the hashed values of the keys.

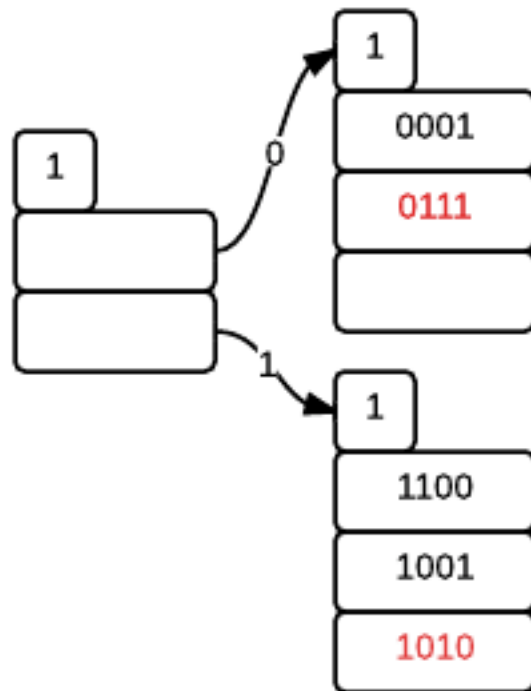


Show the state of the extensible index after keys with hash values 1010, 0111, 1000 and 1011 have been inserted. Order of keys within the block does not matter.

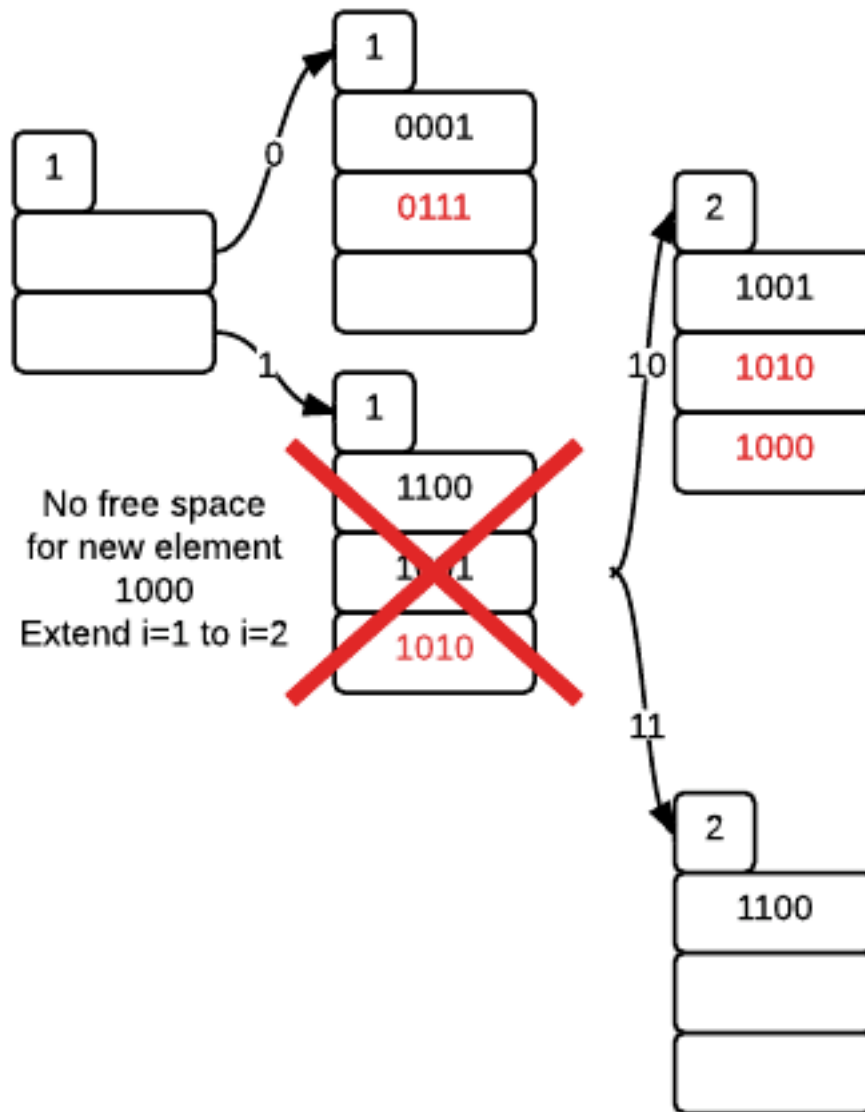
Step 1: Insert 1010



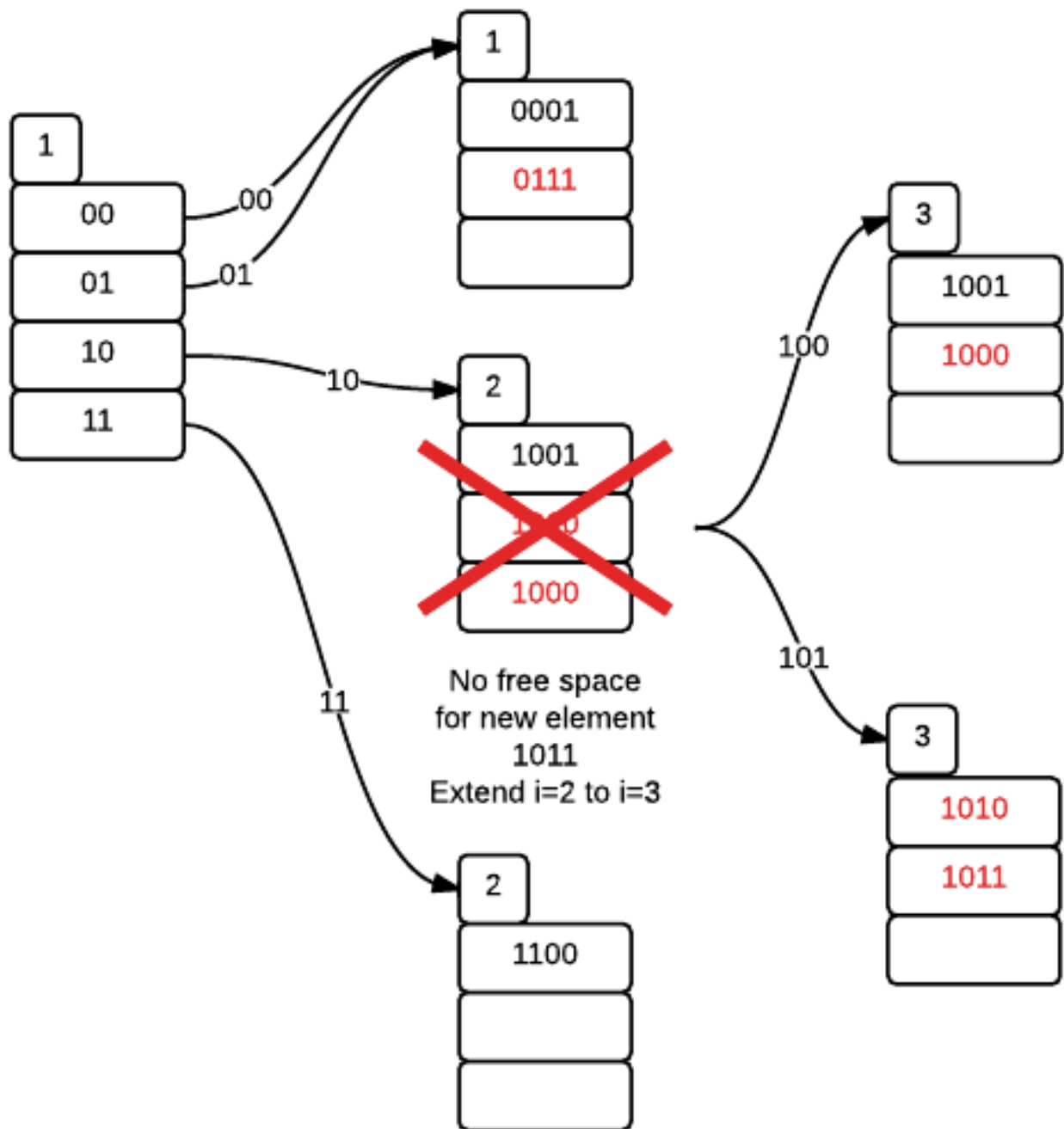
Step 2: Insert 0111

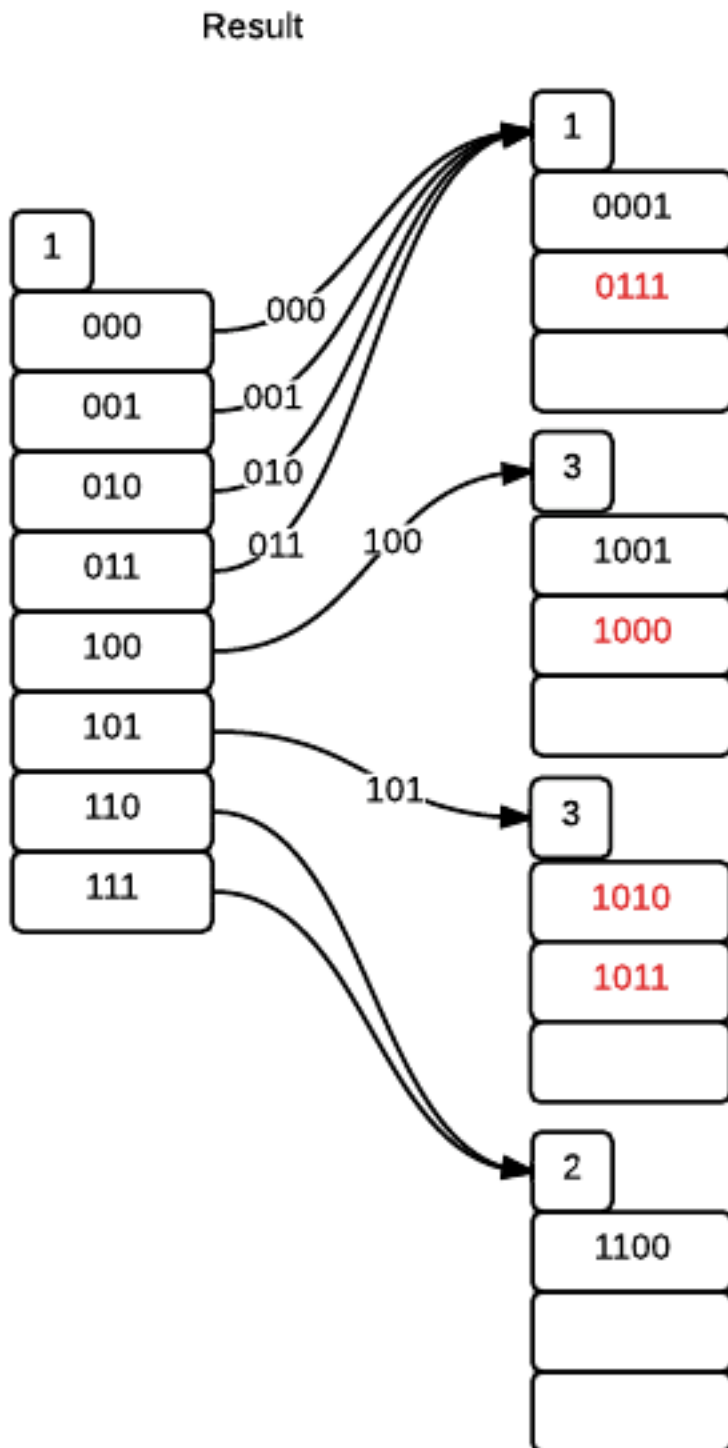


Step 3: Insert 1000



Step 4: Insert 1011





Problem 4 Outerjoins

WRITE EXAMPLE WITH A1, A2, B, C1, C2

Consider the extended projection operator we discussed in the lectures and assume that a function that may be used in the extended projection is the $\text{null}()$, which produces the special value NULL. For example, assuming the following R

A	B	the result of $\pi_{A,B,\text{null}() \rightarrow F} R$ is	A	B	F
1	2		1	2	NULL
f 1	2		1	2	NULL
3	4		3	4	NULL

Next consider the natural left outerjoin operator **LOUT**. Given two relations R and S where B is the set of attributes that are common to R and S, A is the set of attributes that appear only in R and C is the set of attributes that appear only in S, the R **LOUT** S is defined as follows:

- its set of attributes is the concatenation of A, B and C attributes
- for every pair of tuples r from R and s from S such that r and s have equal values in the attributes of B, the result has a tuple with the A, B and C values of r and s.
- for every tuple r of R such that there is no tuple of S that has equal values with r on the common attributes B, the result has a tuple with the A and B values of r and NULL values in C

Provide an algebraic definition for R **LOUT** S using operators we have seen in the notes, including the generalization of projection described above.

Solution: $R \text{ LOUT } S = (R \bowtie S) \cup (\pi_{A,B,\text{null}() \rightarrow C}(R - \pi_{A,B}(R \bowtie S)))$

Steps for solution

Step 1: For every pair of tuples r from R and s from S such that r and s have equal values in the attributes of B, the result has a tuple with the A, B and C values of r and s.
 $\implies R \bowtie S$ (this is the definition for natural join)

Step 2: For every tuple r of R such that there is no tuple of S that has equal values with r on the common attributes B. $\implies (R - \pi_{A,B}(R \bowtie S))$

$$\left. \begin{array}{l} R \text{ contains all tuples} \\ (R \bowtie S) \text{ contains tuples which has equal values to tuples in S} \end{array} \right\} \implies (R - \pi_{A,B}(R \bowtie S))$$

Step 3: The result has a tuple with the A and B values of r and NULL values in C $\implies \pi_{A,B,\text{null}() \rightarrow C}(R - \pi_{A,B}(R \bowtie S))$

Step 4: Combine the above 2 parts and we will get the final solution:

$R \text{ LOUT } S = (R \bowtie S) \cup (\pi_{A,B,\text{null}() \rightarrow C}(R - \pi_{A,B}(R \bowtie S)))$

Assume relations R, S and T. The notation $c(A)$ refers to a condition that refers to a list of attributes A. As is the case with SQL, any condition that is evaluated on an attribute whose value is NULL returns false, for the particular tuple.

Declare true or false each of the following. If the answer is “no”, also provide counterexample.

- $\sigma_{C(A)}(R \text{ LOUT } S) = (\sigma_{C(A)} R) \text{ LOUT } S$, where R has a list of attributes A
- $\sigma_{C(C)}(R \text{ LOUT } S) = R \text{ LOUT } (\sigma_{C(C)} S)$, where S has a list of attributes C but R has no attribute of the list C.
- $R \text{ LOUT } (S \text{ LOUT } T) = (R \text{ LOUT } S) \text{ LOUT } T$, where all of R, S and T have a single common attribute A and no pair of R, S and T has a common attribute other than A.
- $\delta(R \text{ LOUT } S) = (\delta R) \text{ LOUT } S$

Solution

- (A.) $\sigma_{C(A)}(R \text{ LOUT } S) = (\sigma_{C(A)} R) \text{ LOUT } S$, where R has a list of attributes A

True

Prove:

- (1.) The tuple t appears n times in final result of left expression
- (2.) The tuple t appears n times in final result of right expression.
- (3.) So the given equation is correct.

- (B.) $\sigma_{C(C)}(R \text{ LOUT } S) = R \text{ LOUT } (\sigma_{C(C)} S)$, where S has a list of attributes C but R has no attribute of the list C.

False

Counter Example:

R		S	
A	B	B	C
1	2	1	4
2	3	2	3

Given $C(C) : S.C = 3$

$\sigma_{C(C)}(R \text{ LOUT } S)$ will be:

A	B	C
1	2	3

$R \text{ LOUT } (\sigma_{C(C)} S)$ will be:

A	B	C
1	2	3
2	3	NULL

So we can find the given equation is wrong.

- (C.) $R \text{ LOUT } (S \text{ LOUT } T) = (R \text{ LOUT } S) \text{ LOUT } T$, where all of R, S and T have a single common attribute A and no pair of R, S and T has a common attribute other than A.

False

Counter Example:

R		S		T	
A	B	A	C	A	D
1	3	1	3	3	1
2	4	3	4	2	2
				4	3

A	B	C	D
1	3	3	NULL
2	4	NULL	NULL

A	B	C	D
1	3	3	NULL
2	4	NULL	2

So we can find the given equation is wrong

- (D.) $\delta(R \text{ LOUT } S) = (\delta R) \text{ LOUT } S$

False

Counter Example:

R		S	
1	2	2	1
3	1	2	1

A	B	C
1	2	1
3	1	NULL

A	B	C
1	2	1
1	2	1
3	1	NULL

So we can find the given equation is wrong.

Problem 5 Algebra and Estimation

Consider three relations $R(A;B;C)$, $S(A;D)$, $W(B;E)$ and the query

SELECT *

FROM R, S, W

WHERE $R.A=S.A$ AND $R.B=W.B$ AND $R.C=1$

Consider an optimizer that produces all join expressions, where the selection $\sigma_{R.C=1}$ is pushed down and applied directly on the table R.

Plans cannot have cartesian products or trivial natural joins that are equivalent to cartesian products.

Write all possible logical query plans (i.e., algebraic expressions) that this optimizer will produce but do not write twice expressions that can be derived from each other just by using the commutativity of the natural join.

Solution:

(1.) $(\sigma_{R.C=1} R \bowtie S) \bowtie W$

(2.) $(\sigma_{R.C=1} R \bowtie W) \bowtie S$

Other solutions are not correct because of the blue and red sentences.

Size Estimation

Assume that the optimizer selects the best logical query plan by estimating the size of each intermediate result (i.e., of each subexpression) and selecting the plan that has the smallest sum of intermediate result sizes. For each of the logical query plans you provided earlier, calculate the size of each intermediate result and decide which is the best plan. Note that you do not have to estimate the size of the end result since it should always be the same.

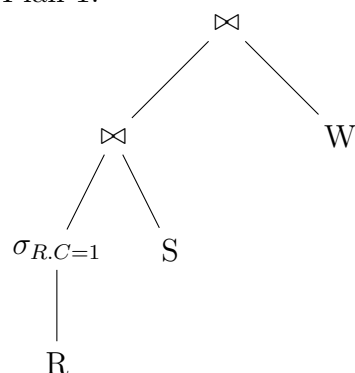
Use the following data for your estimates. Write whether your estimate is the precise number, i.e., whether the number will be the same for all possible databases that conform to the following characteristics.

- B is a key of R and W.B is a non-null foreign key that references R.B
- A is a key of S and R.A is a non-null foreign key that references S.A
- $T(R) = 10^6$
- $T(S) = 10^5$
- $T(W) = 10^3$
- $V(R.C) = 10^2$

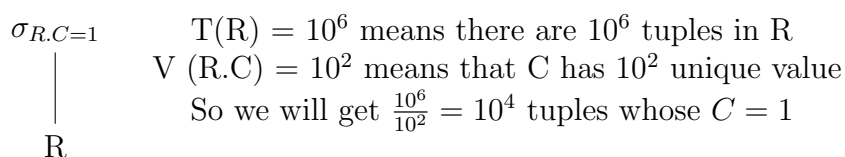
Solution:

The second one is better.

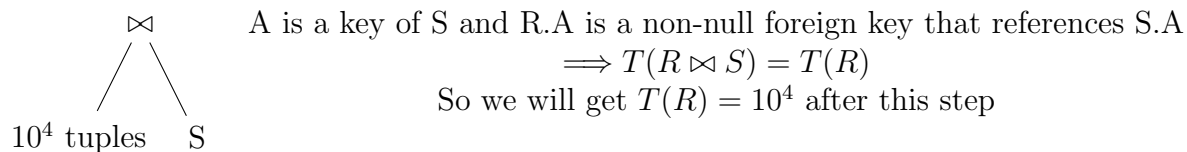
Plan 1:



Step 1:



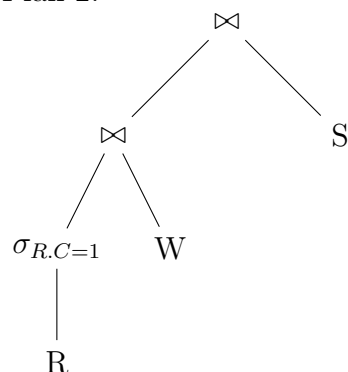
Step 2:



Step 3: Note that you do not have to estimate the size of the end result since it should always be the same. It means we do not need to compute more after step 2.

Step 4: Final result SizeEstimation = 2×10^4

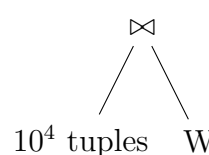
Plan 2:



Step 1:

$\sigma_{R.C=1}$ $T(R) = 10^6$ means there are 10^6 tuples in R
 \downarrow $V(R.C) = 10^2$ means that C has 10^2 unique value
 R So we will get $\frac{10^6}{10^2} = 10^4$ tuples whose $C = 1$

Step 2:


 B is a key of R and W.B is a non-null foreign key that references R.B
 $\implies T(R \bowtie W) = T(W)$
 So we will get $T(W) = 10^3$ after this step

Step 3: Note that you do not have to estimate the size of the end result since it should always be the same. It means we do not need to compute more after step 2.

Step 4: Final result SizeEstimation = $10^4 + 10^3$

So we can find the second plan is better, it will has the smallest sum of intermediate result sizes.