Section 7 Worksheet

Join algorithms

- (Block) Nested Loop Join - two-level for-loop
- Hash Join - compute a hash table of one input; probe the hash table with the other input
- Sort-Merge Join - sort both tables on one of the join conditions, then merge sorted lists

Indexes

- B-Tree index - supports point and range lookup
- Hashtable - supports point lookup

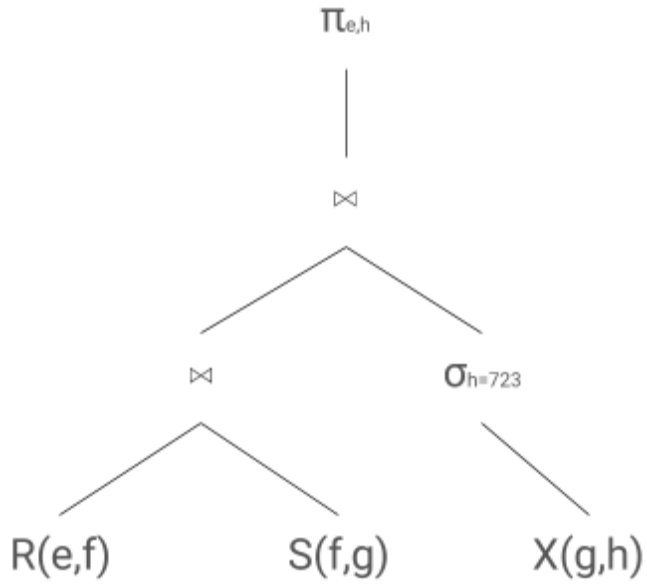
Cost Modeling

- B(R) - the number of blocks used to store the relation R on disk
- T(R) - the number of tuples in R (also known as R's cardinality)
- V(R, a) - the number of unique values of attribute an in relation R
- M - the number of pages that fit in memory

Cost-based Query Optimization compares plans by computing their estimated cost, then chooses the one with the cheapest estimated cost to execute.

Query execution - we learned about the iterator method (iterator interface) of executing the operators in a query plan. You may see it called the "pull-based model of query execution", because each operator "pulls" data from its child operators by calling next(). The three methods used are open(), next(), and close().

Problems



1. (Adapted from 414 SP 17 Final)
Consider the relations R(e, f), S(f, g), and X(g, h) in the query plan depicted above.
- Joins are natural joins.
- Every attribute is integer-valued.
- Assume that **every intermediate result is materialized** (i.e., written to disk).
- Assume that we are executing queries on a machine that has **11 memory pages** available.
- Assume uniform distributions on the attributes for the purpose of computing estimates.

Consider the following statistics:

Table	#tuples	#blocks
R	1,000	100
S	5,000	200
X	100,000	10,000

Attribute	# distinct values	Minimum	Maximum
R.f	100	1	1,000
S.f	1,000	1	2,000
S.g	5,000	1	2,000
X.g	1,000	1	10,000
X.h	1,000	1	500,000

A. Estimate the number of tuples and blocks in the selection $\sigma_{h=723}(X)$.

B. Estimate the number of tuples and blocks in the join $R \bowtie S$.

This is a HARD question. Cardinality estimation of joins has been an active research topic for many years. We don't have enough statistics on R and S to make a good estimate. Here is a common estimate:

$T(R \bowtie S) \approx T(R) * T(S) / \max\{V(R, f), V(S, f)\} = 1{,}000 * 5{,}000 / \max\{100,1000\} = 5{,}000$
Assume about 300 blocks, since the estimated number of tuples does not differ from S but the number of attributes has increased by 50%.

C. What is the estimated I/O cost of $R \bowtie S$ if implemented by a block nested loop join?

D. What is the estimated I/O cost of $R \bowtie S$ if implemented by an indexed join? Assume that we have an unclustered index on S(f).

2. Create the index that is easiest to create that will make the following queries run faster. Assume there are no previous indexes.

a.
```
SELECT * FROM R
WHERE R.f > 100 AND R.f < 700
```

b.
```
SELECT * FROM S
WHERE S.g = 344
```

3. We have the relation V(m, n, p), W(p, q, r) and the following two queries. For each of the unclustered indexes below identify which queries will run faster under that index versus no index. Assume all attributes range from 0 to 1000 and are distributed uniformly.

```
(A) SELECT * FROM V
    WHERE V.m = 344
(B) SELECT * FROM V
    WHERE V.m = 344 AND Vp = 311
(C) SELECT * FROM V, W        (assume nested loop join)
    WHERE V.p = W.p
a. INDEX idx1 on V(m)
b. INDEX idx2 on V(m,p)
c. INDEX idx3 on V(p,m)
```

4. (344 AU16 MT)

Consider the relations

Purchase(pid, custId, quantity, price), Customer(custId, name, city)

with the statistics

T(Purchase) = 1000	T(Customer) = 3000
B(Purchase) = 100	B(Customer) = 200
V(Purchase, price) = 100	V(Customer, custId) = 3000

Price range = [0, 200)

V(Purchase, custId) = 500

Number of memory pages available = 20

a. Consider the query below and circle the indexes that will produce a speed up:

```
SELECT * FROM Purchase P, Customer C
WHERE P.custId = C.custId
  AND P.price < 100 AND C.custId = 42
```

(1) Hashtable index on Purchase(price)

(2) B-tree index on Purchase(pid, price)

(3) Hashtable index on Customer(custId)

(4) Hashtable index on Purchase(custId)

(5) B-tree index on Purchase(price, pid)

(6) Hashtable index on Purchase(price, pid)

b. Write the optimal query plan that implements this query, including when you would perform selections and what algorithm you would use for the join. Assume that there are no indexes available. Briefly explain why your plan is the cheapest.