CSE 444 Final Review

Possible Final Topics

SQL
Conceptual Design (BCNF)
Transactions
    Recovery
    Concurrency
Query execution and optimization
    Dynamic Programming
Indexes
    B+ Trees
Cardinality Estimation
    Statistics
Parallel Databases
    Map Reduce
    Bloom Filters

Note: this is not a complete list of topics

Recall the following pseudo code for counting words in a document.

```
map(String key, String value):
        // key: document name
        // value: document contents
        for each word w in value:
                EmitIntermediate(w, "1"):

reduce(String key, Iterator values):
        // key: a word
        // values: a list of counts
        int result = 0;
        for each v in values:
                result += ParseInt(v);
        Emit(AsString(result));
```

Working with R(A, B) S(C,D), implement the following in Map Reduce pseudo code:

a) Select * from R where B<9 [selection]

```
map (String key, String value):
        if(value.B <9)
                EmitIntermediate(key, value);

reduce (String key, Iterator values):
        for each v in values:
                Emit(key + values);
```

b) Select distinct B from R [duplicate elimination]

```
map (String key, String value):
        EmitIntermediate(value.B, key + value);

reduce (String key, Iterator values):
        Emit(key);
```

c) Select * from R, S where R.B = S.D [join]

```
map (String key, String value):
        If(value.type==R)
                EmitIntermediate(value.B, key + value);
        Else
                EmitIntermediate(value.D, key + value);
```

```
reduce (String key, Iterator values):
        for each s in values:
                if(s.type == S){
                        for each r in values:
                        if(r.type == R){
                                Emit(key + s + r);
                        }

                }
```

[From fall 2007](A little dynamic programming) Assume that we have four relations W, X, Y, and Z, and we wish to find the lowest cost plan for computing the join of the relations W |x| X |x| Y |x| Z using *left - deep* query trees **only**. Assume that the sizes are as follows:

T(W) = 1500, T(X) = 2000, T(Y) = 1000, T(Z) = 3000

Also assume that the size of a join can be estimated to be 1% of the size of the cross product, i.e., the size of T(A |x| B) = 0.01 * T(A) * T(B). Fill in the following table with the best plans and the associated costs of the subqueries and final query. Some of the table is filled in for you. Remember that you should only consider left - deep, not right - deep or bushy, join trees.

Subquery	Size	Cost	(Best) Plan
WX	30k	0	WX
WY	15k	0	WY
WZ	45k	0	WZ
XY	20k	0	XY
XZ	60k	0	XZ
YZ	30k	0	YZ
WXY	300k	15k	(WY)X
WXZ	900k	30k	(WX)Z
WYZ	450k	15k	(WY)Z
XYZ	600k	20k	(XY)Z
WXYZ	9,000k	315k	(WXY)Z

(a) [8 points] Consider two tables $R(A, B)$ and $S(C, D)$ with the following statistics:

$$\begin{aligned} B(R) &= 5 \\ T(R) &= 200 \\ V(R, A) &= 10 \\ B(S) &= 100 \\ T(S) &= 400 \\ V(S, C) &= 50 \\ M &= 1000 \end{aligned}$$

There is a clustered index on $S.C$ and an unclustered index on $R.A$. Consider the logical plan:

$$P = \sigma_{A=??}(R) \bowtie_{B=C} S$$

There are two logical operators, $S = \sigma_{A=??}$ and $J = \bowtie_{B=C}$, and for each we consider two physical operators:

$$\begin{aligned} s1 &= \text{one pass table scan} \\ s2 &= \text{index-based selection} \\ j1 &= \text{main memory hash join} \\ j2 &= \text{index-based join} \end{aligned}$$

Both $s1$ and $s2$ are pipelined, i.e. the result of the select operator is not materialized. For each of the resulting four physical plans compute its cost in terms number of disc I/Os, expressed as a function of the statistics above. Your answer should consists of four expressions, e.g. $\texttt{COST}(s1j1) = B(R)B(S)/M + V(R, A)$ (not the real answer).

i. $\texttt{COST}(s1j1) =$ B(R) + B(S)

ii. $\texttt{COST}(s2j1) =$ T(R)/V(R,A) + B(S)

iii. $\texttt{COST}(s1j2) =$ B(R) + (T(R)/V(R,A)) * (B(S)/V(S,A))

iv. $\texttt{COST}(s2j2) =$ T(R)/V(R,A)) + (T(R)/V(R,A)) * (B(S)/V(S,A))

(b) **[2 points]** Indicate the cheapest plan of the four, together with its cost expressed as a number. You will get credit for this point only if you compute correctly all four expressions above.

45

[View CSE 444 final of fall 2009 for solution to this problem]
[fall 2009]
We would like to explore logical query plans involving data about downhill skiing races. Three tables are involved:

Athlete(aid, name, country)
Event(eid, year, location)
Result(eid, aid, time)

The Athlete table gives a unique id (aid) for each athlete and his/her name and country. The Event table gives a unique id (eid) for each event and the event year and location (for example, 2010, Vancouver). The Result table gives the athlete's time for each race that he/she participated in. In the Result table, eid and aid are foreign keys in the Athlete and Event tables.

We have the following statistics for these tables:

B(Athlete) = 100 B(Event) = 40 B(Result) = 250
T(Athlete) = 2,000 T(Event) = 400 T(Result) = 50,000
V(Athlete, country) = 100 V(Event, year) = 50
V(Event, location) = 25

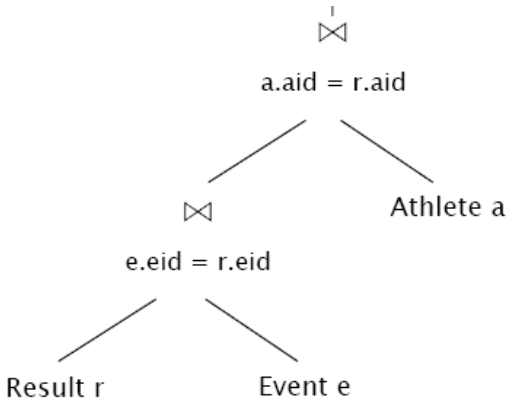
The Athlete and Event tables are clustered on their primary keys, aid and eid respectively. The Result table is not clustered. There are separate indices on all attributes of all tables except for time in Results.

You should assume that there is more than enough main memory (M) to hold any or all parts of these tables so we can use one- pass algorithms to implement queries.

(a) Translate the logical query plan in the diagram on the previous page to SQL.

(b) What is the estimated cost of the logical query plan given in the diagram on the previous page? For full credit you should both give formulas involving things like T(…), B(…), and V(…, …), and then substitute actual numbers and give a numerical answer as your final estimate.

(c) For this question we go back to the original set of joins at the base of the tree from the previous problem, using the same relations with the same statistics as before.



If we decide to implement this part of the logical plan exactly as given, what physical plan should we use to perform it? Your answer should specify the physical join operators used (hash, nested loop, sortmerge, etc.) and the access methods used to read the tables (sequential scan, index, etc.) For operations where it matters, be sure to include the details – for instance, for a hash join, which relation would be stored in the hash tables; for a loop join, which relation would be the inner or outer loop. You should specify how the top - most join reads the result of the lower one. You may assume there is
enough main memory available for whichever algorithm(s) you choose. Give the estimated cost of your physical plan in terms of the number of disk operations needed. Also give a brief explanation of why your plan is the best one in terms of the overall cost.