

# CPS 216-Spring 2003

## Advanced Database Systems

### Quiz Two & Review Questions

Instructor: Prof. Jun Yang  
TA: Junyi Xie  
Department of Computer Science  
Duke University  
{junyang, junyi}@cs.duke.edu

April 4, 2003

## 1 Exercise 1

Consider the following relational schema and SQL query:

```
Suppliers(sid: integer, sname: char(20), city: char(20))
Supply(sid: integer, pid: integer)
Parts(pid: integer, pname: char(20), price: real)
```

```
SELECT S.sname, P.pname FROM Suppliers S, Parts P, Supply Y WHERE
S.sid = Y.sid AND Y.pid = P.pid AND S.city = 'Durham' AND P.price
< 1,000
```

### Questions

- 1. What information about these relations will the query optimizer need to select a good query execution plan for the given query?
- 2. How many different join orders, assuming that cross-products are disallowed, will a System R style query optimizer consider when deciding how to process the given query? List each of these join orders.
- 3. What indexes might be of help in processing this query? Explain briefly.
- 4. How does adding DISTINCT to the SELECT clause affect the plans produced?
- 5. How does adding ORDER BY sname to the query affect the plans produced?
- 6. How does adding GROUP BY sname to the query affect the plans produced?

### Answer

The answer to each question is given below.

- 1. The query optimizer will need information such as what indexes exist (and what type) on:  $S.sid, Y.sid, Y.pid, P.pid, S.city, P.price$  It will also need statistics about the database such as low/high index values and distribution between fields.
- 2. Only left-deep plans are allowed:  $(S \bowtie Y) \bowtie P$  and  $((Y \bowtie P) \bowtie S)$ .
- 3. A sorted, clustered index on  $P.price$  would be useful for range retrieval. A B+ Tree index on  $S.sid, Y.sid, Y.pid, P.pid$  could be used in an index-only sort-merge.

- 4. To support the DISTINCT selection, we must sort the results (unless they already are in sorted order) and scan for multiple occurrences. Different sorted orders are known as "interesting orders" in the System R optimizer, and these orders are considered when determining the plan.
- 5. The ORDER BY *sname* selection would have effects similar to question 4. The optimizer would consider plans which left *sname* ordered as a side effect and plans which ordered *sname* directly.
- 6. The GROUP BY *sname* clause requires us to sort the results of the earlier steps on *sname*, and to compute some aggregate (e.g., SUM) for each group (i.e., set of tuples with the same *sname* value).

## 2 Exercise 2

Consider a relational algebra expression of the form  $\sigma_c(\pi_l(R \times S))$ . Suppose that the equivalent expression with selections and projections pushed as much as possible, taking into account only relational algebra equivalences, is in one of the following forms. In each case give an illustrative example of the selection conditions and the projection lists (c, l, c1, l1, etc.).

- (a) Equivalent maximally pushed form:  $\pi_{l1}(\sigma_{c1}(R) \times S)$ .
- (b) Equivalent maximally pushed form:  $\pi_{l1}(\sigma_{c1}(R) \times \sigma_{c2}(S))$ .
- (c) Equivalent maximally pushed form:  $\sigma_c(\pi_{l1}(\pi_{l2}(R) \times S))$ .
- (d) Equivalent maximally pushed form:  $\sigma_{c1}(\pi_{l1}(\sigma_{c2}(\pi_{l2}(R)) \times S))$ .
- (e) Equivalent maximally pushed form:  $\sigma_{c1}(\pi_{l1}(\pi_{l2}(\sigma_{c2}(R)) \times S))$ .
- (f) Equivalent maximally pushed form:  $\pi_l(\sigma_{c1}(\pi_{l1}(\pi_{l2}(\sigma_{c2}(R)) \times S)))$ .

**Answer**

- 1.  $\sigma_{A=1}(\pi_{ABCD}(R \times S)) = \pi_{ABCD}(\sigma_{A=1}(R) \times S)$
- 2.  $\sigma_{A=1, C=2}(\pi_{ABCD}(R \times S)) = \pi_{ABCD}(\sigma_{A=1}(R) \times \sigma_{C=2}(S))$
- 3.  $\sigma_{C=5}(\pi_{BC}(R \times S)) = \sigma_{C=5}(\pi_C(\pi_B(R) \times S))$
- 4.  $\sigma_{B=1, C=3}(\pi_{BC}(R \times S)) = \sigma_{B=1}(\pi_C(\sigma_{C=3}(\pi_B(R) \times S)))$
- 5.  $\sigma_{B=1, C=3}(\pi_{BC}(R \times S)) = \sigma_{C=3}(\pi_C(\pi_B(\sigma_{B=1}(R) \times S)))$
- 6.  $\sigma_{A=1, B=D}(\pi_{BC}(R \times S)) = \pi_{BC}(\sigma_{B=D}(\pi_{BCD}(\sigma_{A=1}(R) \times S)))$

## 3 Review Questions of Query Optimization

Briefly answer the following questions.

- 1. What is the goal of query optimization? Why is it important?
- 2. Describe the advantages of pipelining.
- 3. Give an example in which pipelining cannot be used.
- 4. Describe the iterator interface and explain its advantages.
- 5. What role do statistics gathered from the database play in query optimization?
- 6. What information is stored in the system catalogs?
- 7. What are the benefits of making the system catalogs be relations?

## Answer

- 1. The goal of query optimization is to avoid the worst plans and find a good plan. The goal is usually not to find the optimal plan. The difference in cost between a good plan and a bad plan can be several orders of magnitude: a good query plan can evaluate the query in seconds, whereas a bad query plan might take days!
- 2. Pipelining allows us to avoid creating and reading temporary relations; the I/O savings can be substantial.
- 3. Bushy query plans often cannot take advantage of pipelining because of limited buffer or CPU resources. Consider a bushy plan in which we are doing a selection on two relations, followed by a join. We cannot always use pipelining in this strategy because the result of the selection on the first selection may not fit in memory, and we must wait for the second relation's selection to complete before we can begin the join.
- 4. The iterator interface for an operator includes the functions open, get next, and close; it hides the details of how the operator is implemented, and allows us to view all operator nodes in a query plan uniformly.
- 5. The query optimizer uses statistics to improve the chances of selecting an optimum query plan. The statistics are used to calculate reduction factors which determine the results the optimizer may expect given different indexes and inputs.
- 6. Information about relations, indexes, and views is stored in the system catalogs. This includes file names, file sizes, and file structure, the attribute names and data types, lists of keys, and constraints. Some commonly stored statistical information includes:
  - (a) Cardinality - the number of tuples for each relation
  - (b) Size - the number of pages in each relation
  - (c) Index Cardinality - the number of distinct key values for each index
  - (d) Index Size - the number of pages for each index (or number of leaf pages)
  - (e) Index Height - the number of non-leaf levels for each tree index
  - (f) Index Range - the minimum present key value and the maximum present key value for each index.
- 7. There are several advantages to storing the system catalogs as relations. Relational system catalogs take advantage of all of the implementation and management benefits of relational tables: effective information storage and rich querying capabilities. The choice of what system catalogs to maintain is left to the DBMS implementor.