

This homework is based on topics mostly from Chapters 4 and 5 of [AHV95] and on [Gra93]. Please submit your work electronically as a PDF file named using the scheme `LastnameIJ-hw1-NNNN.pdf` (replacing `LastnameIJ` with your last name and initials, and `NNNN` with a 4-digit number) by anonymous FTP upload to `ftp.cs.umd.edu`, directory `/incoming/chaw/724`. The first question below is worth 50 points and the rest are worth 10 points each (although they vary in difficulty).

1. You may use either PostgreSQL or Oracle (or another DBMS, if you check with me first) for this question.

Write a program that pretty-prints the query execution plan (used by PostgreSQL or Oracle) for a given query. For a query of your choice, demonstrate the optimizer producing at least three different plans. (Hint: you can influence the optimizer by creating and dropping indexes, and by changing the cardinalities of your database relations.)

Submit all your *source* code (only) for this question as a gzipped, tar file that includes files `README` and `Makefile` containing the customary information. Name your submission using the scheme `LastnameIJ-hw1-src-NNNN.tar.gz` similar to that described above. In addition, unless your code produces very well formatted results, depict the different query plans in your main submission (as figures using the usual conventions, for example).

2. (The following questions were originally posted to the newsgroup.) (1) Are the 5-minute and 10-byte rules still valid? (2) How have changes in technology between the paper's publication date and now changed the applicability of the rules? (3) What would be your prediction for the rules five years into the future? (4) Are there any current or potential future technological developments that would invalidate the rules (qualitatively)?
3. Describe how the following query is executed in query engines based on each of (1) temporary files; (2) interprocess communication using pipes; (3) transformation into iterative program; and (4) iterators:

```
select A, D
from R, S
where R.B = S.C;
```

The database schema includes relations $R(A, B)$ and $S(C, D)$. Describe clearly each step between the input of the above query and the output of the query result. Use sample instances of R and S to illustrate your answer. Indicate which steps are common for the four query engine architectures and which are different.

4. Is a *sort* operator, with the obvious semantics, a logical or physical operator? Does your answer change if the query language changes (say, from SQL to relational algebra)? Repeat for an operator *merge-sort*, with the obvious semantics.
5. Explain how the following query should execute in an iterator-based query engine that exploits common subexpressions.

```

select R1.A
from (select A from R where B > 5) as R1(A),
      (select A from R where B > 5) as R2(A)
where R1.A > R2.A;

```

Write a simpler SQL query that is equivalent to the above query.

6. Provide a quantitative analysis of generating level-0 runs using in-memory sorting compared with using replacement selection.
7. For a (real) machine of your choice (provide details), identify the disk and I/O subsystem and provide the specifications. Run a micro-benchmark of your choice to measure I/O throughput and latency and compare your results with the specifications. Explain the differences, if any. Hint: [Bra01] (but you are welcome to use another benchmark, perhaps one you design yourself).
8. Explain the *Guy Lohman test* for join techniques. Provide a *concrete* example, other than the one in the paper, of a method that fails the test, and another of a method that passes the test.
9. Solve Exercise 4.16 of [AHV95]. Hint: Focus on satisfiability of queries; see page 60.
10. Solve Exercise 4.22(d) of [AHV95]. If you claim one language is dominated by another, give at least an informal justification of your claim. (For example, indicate how an arbitrary query in one language can be translated to an equivalent one in the other.) If you claim strict domination, demonstrate at least one query that can be expressed in one language but not the other.
11. Solve Exercise 4.30(a) of [AHV95]. Hint: For the strictness result, take an example query over a binary relation that returns tuples whose first attribute is less than their second attribute.
12. Solve Exercise 5.2 in [AHV95]. Note that you are required to provide queries in both algebras (named and unnamed). Further, for each calculus query you write, provide at least an informal justification for domain independence.
13. Solve Exercise 5.4(d) in [AHV95]. Hint: Use induction on the number of operators in a query.
14. Solve Exercise 5.18 in [AHV95]. For translations to algebra, use Theorem 5.4.8. Show at least one translation in full detail; for the rest, you can skip steps.

15. Solve Exercise 5.33 in [AHV95]. Even though the exercise is marked as difficult, there is an easy proof of the result. (Hint: You are free to use the results in the chapter.)
16. (More difficult) Solve Exercise 5.34. If you cannot prove everything, try to clearly outline how a proof would proceed and which steps are missing. Even if you have trouble with the formal machinery, try to give some informal arguments for part (b). Partial credit will depend on how clearly you present your work.

References

- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [Bra01] Tim Bray. Bonnie. Available at <http://www.textuality.com/bonnie/>, 2001. Benchmark description, code, and results.
- [Gra93] Goetz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–169, 1993.