

Please be brief and precise in your responses; you will lose points for wordy or ambiguous answers. You do *not* need to use formal notation in your proofs, although you are welcome to do so; however, your proofs must be presented in a manner that makes it easy to follow your reasoning. You are free to use any print or electronic resources. However, you must submit only text written by you, following the usual norms of quoting and citation when appropriate. You must not discuss this exam with anyone before the time it is due. If you need clarifications, remember to ask **before noon on Friday, March 21st**.

1. (15 pts) Consider the algebraic query language (in the named perspective) defined by the following template for queries:

$$\pi_{A_1, A_2, \dots, A_i} \sigma_F(\rho_{f_1} R_1 \bowtie \rho_{f_2} R_2 \bowtie \dots \bowtie \rho_{f_m} R_m)$$

where each ρ_{f_i} is a renaming operator for $\text{sort}(R_i)$ and where each A_i is an attribute name that occurs in the union of the sorts of $\rho_{f_i} R_i$.

How does this language relate to the SPJR language [AHV95, Section 4.4.] in terms of expressive power. Prove the relation you claim (equivalence, strict domination, or neither).

2. (20 pts) The SPJR algebra [AHV95, Section 4.4] uses the so-called *atomic* selection operator that permits only two forms: $\sigma_{A=c}$ and $\sigma_{A=B}$, where A and B are attribute names, and c is a constant. Consider the language S_{\neq} PJR formed by extending SPJR to allow inequalities of the forms $A \neq c$ and $A \neq B$ in the selection operator. Similarly, consider the language S_{\leq} PJR formed by extending SPJR to allow comparison predicates of the form $A \leq c$ and $A \leq B$ in the selection operator. For simplicity, assume that the domain is totally ordered. Finally, let $S_{\neq, \leq}$ PJR be the language that permits both inequalities and comparison predicates in the selection operator. Describe the relative expressive powers of the four languages: SPJR, S_{\neq} PJR, S_{\leq} PJR, and $S_{\neq, \leq}$ PJR. That is, indicate how the expressive powers of each language in this set relates to that of every other language in the set. Prove the domination and equivalence relationships you claim, and also justify lack of such relationships.
3. (15 pts) Which of the five aggregation operators in standard SQL (min, max, count, sum, and avg) are expressible in CALC? Justify your answer (by giving, for each operator, a method to express it in CALC or a proof that it cannot be expressed). For the operators expressible in CALC, briefly compare the efficiency of query plans generated for CALC versions of aggregation queries with that of plans generated for the extended algebra versions [GMUW02, Section 5.4].

Indicate how universal quantification is expressible in SPJR. For example, how is the following query expressed in SPJR, given $\text{Compatible}(\text{printer}, \text{opsys})$? Find the printers that are compatible with all operating systems. Compare the query plan for SPJR expressions of such queries with those described in [Gra93, Section 6]

4. (15 pts) Devise a syntactic restriction on calculus queries that yields a language CALC_x that strictly dominates the safe-range calculus, CALC_{sr} . Prove that CALC_x permits only safe queries and that it strictly dominates CALC_{sr} . Hint: Try modifying the definition of *safe range* queries in [AHV95, Section 5.4]. Try to make CALC_x as natural as possible. For example, don't define CALC_x to be CALC_{sr} augmented with a predefined safe query that is not in CALC_{sr} !
5. (20 pts) Describe a multi-pass hybrid hash join algorithm. Present pseudocode and an analysis that includes the main memory required, and the disk I/O incurred as functions of the input. Indicate how main memory is divided between buckets that are stored in their entirety and buckets that have only one block each in memory. Hint: See [GMUW02, Sections 15.5.6 and 15.8.3] and [Gra93, Section 5.3].

Explain how this algorithm copes with errors in the estimates of the cardinalities, skew in hash values, and other problems that may lead to buckets overflowing their allotted space.

Describe how the rate at which this algorithm produces output tuples (tuples per second) varies over the period during which the join is evaluated. Try to provide a precise description. You may make simplifying assumptions and may use a well-chosen example that illustrates the main ideas. (That is, you do not need to give a fully general description.)

6. (15 pts) Describe a method for storing XML data using relations. You may use any method you like, but you cannot assume the existence of user- or system-defined functions that parse XML text to query or extract elements and attributes. Present a method for mapping XPath 1.0 queries on the original XML data to equivalent *standard* SQL queries on the relational representation. You are free to use recursion as supported by SQL-99 [GMUW02, Section 10.4]. Try to describe a method that is general (not specific an XML instance or class of instances) and that produces simple SQL queries.

Submission: Please submit your work electronically as a PDF file named using the scheme `PublicJQ-mt-NNNN.pdf` (replacing PublicJQ 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`. Please make sure that the PDF file you upload is viewable using the `gv` program on the `linuxlab` machines.

References

- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [GMUW02] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice-Hall, 2002.
- [Gra93] Goetz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–169, 1993.