Qualifying Examination – Fall 1998
Parallel Programming Languages and Systems

You are the computing expert in a large insurance company, that handles all kinds of insurances (life, home, auto, ..). The insurance company maintains a huge country-wide database of its insured projects. The batch posting of all transactions done during the day is done overnight in a 10 hour shift between 9PM to 7AM. However, the company is growing at an exponential pace and 10 hours is not enough time to process all the transactions. The processing programs are written in an SQL like language with operators for selection, projection, join, union, etc.
You are given the task of coming up with solutions so that processing can be speeded up to keep it within the 10 hour limit.
Note: Any assumptions that you make should be clearly stated. You are allowed to make simplifying assumptions, e.g., the join operation is only performed on the key attribute, that the two relations involved in the join have the same key, etc.

1. What **simple** immediate solutions (e.g., without hiring an expert in parallel processing) you can come up with so as to ease the company's problems? Briefly discuss advantages and limits of your solution. [15 points]

2. Your company requires your solution to work in any situation. You think hard and decide to parallelize the operations in the database language that the processing programs are written in. This means that you have to parallelize operations such as projection, selection and join. Give efficient parallel implementation of these three operations, assuming that you have a shared memory machine (your implementation should allocate and use proper locks, etc., to ensure mutual exclusion, where needed). You can assume the relations to be static at the time of processing, i.e., the whole relation is known. [20 points]

3. In coming up with a solution for the Insurance company in Problem 2 you could assume that the relations involved were static (i.e., the entire relation is known to you in advance). Will your implementation work if the relations were not known to you in advance, i.e., their tuples becomes available to you asynchronously in an incremental way while you are processing the join?

   Assume that the relations are dynamic and assume also that the different tuples of the two relations (call them R1 and R2) being joined may appear in different processors. Give an implementation (pseudo-code is fine) for parallel join of two dynamic relations. [15 points]

   Note 1: To write your implementation, assume that you have a primitive, called get_next_elements available to you which, when executed on a particular processor, returns a structure containing two pointers. The first pointer points to a list containing tuples asynchronously generated on that processor for R1 since the last call to get_next_elements on that processor, while the second pointer points to a similar list for R2. When there are no tuples left in a relation the corresponding pointer in the structure is NULL.

Note 2: Assume that the relations being joined, R1 and R2, are stored in some global data-structure in the shared memory.

Note 3: You don't want a solution where you first obtain the two relations R1 and R2 entirely by repeatedly calling get_next_element(R1) and get_next_element(R2) and then using your solution in Problem 2. Your implementation should produce the joined tuple as soon as the two component tuples become available.