Did this paper address an important issue? Explain.
Considering that DBMS systems are becoming increasingly compute and memory bound, one would feel that they should perform well on modern processors employing sophisticated memory-latency-hiding techniques such as non-blocking caches and out-of-order and speculative execution. However, it is found that these applications do not perform as well as scientific applications on such processors. It is important to understand why DBMSs perform poorly even on modern processors. If this is understood well, then either the DBMS designers can take appropriate measures to improve the performance, or processor designers can create processors better suited for DBMS applications. This paper is trying to pinpoint the bottlenecks that cause poor performance of DBMSs, and so it is addressing an important issue.

Are the proposed approaches valid? Describe its strength and weakness.
Yes. The authors use different DBMSs running on the same platform, so that any common trends can be spotted. They want to exclude I/O effects, so they use a memory resident database. They used simple queries so that the results are easier to analyze. They have proposed a simple analytical framework to consider all the components of the execution time of a database query. They have used the performance counters on the processor that they are using (Pentium II Xeon) and an associated software tool called `emon' to control those counters. Using these counters they were able to measure the time spent in various types of stalls.

One of the weakness of their study is that they are not able to measure the time of all events directly. So they have to estimate the stall time for such events. Also, they did not take into account any contentions for resources among instructions.

Do the results support the conclusions? Explain.
Their main conclusions are:

- 90% of the memory stall time is due to L2 D-cache misses and L1 I-cache misses.
- Around 20% of all the stalls are due to architectural characteristics like branch misprediction, TLB misses, etc.
- The behavior of their simpler queries is similar to the behavior of TPC-D, though it is not very similar to TPC-C

Their results support the first two conclusions well. But the third conclusion seems a little arbitrary, since only the CPI breakdown of their queries and the TPC-D queries are similar, but the cache-related stall time breakdown for the two benchmarks are not similar.

Describe the potential future works?
In this paper, they only focus on finding the memory-access related bottlenecks. They have not considered I/O, which can be a significant bottleneck for OLTP-like workloads. Also, they only list the
bottlenecks, without giving any insight as to why that particular bottleneck exists. They could study
the actual causes behind the bottlenecks. If they use a processor with better support for performance
monitoring (e.g. Itanium), then they might be able to measure certain parameters that they were not
able to measure due to the limited performance monitoring support in the Pentium II Xeon.