SUMMARY:
Per-Åke Larson, Jonathan Goldstein, and Jingren Zhou.
MTCache: Transparent mid-tier database caching in SQL Server.
In *Proc. International Conference on Data Engineering (ICDE'04)*, pages 177-189, 2004.

DATE: 11 January 2009

The objective of the work described in this paper is to improve the scalability of multi-tier servers consisting of a backend database management system (DBMS) behind a middle tier of application servers. The paper observes that the DBMS can easily become the bottleneck in such a system. It proposes to alleviate this bottleneck by introducing additional DBMS in the middle tier. These additional DBMS are called MTCache systems because they effectively cache part of the data that is stored in the backend DBMS. All database requests from an application server are sent first to an MTCache system, which decides whether it handle the request itself (if possible) or to forward it to the backend DBMS. By handling requests itself, an MTCache system reduces the load on the backend DBMS. The idea is to scale the system by adding more MTCache databases in the middle tier.

Each MTCache system is aware of all of the tables and other objects in the backend database, although these objects are not stored in the MTCache system. What the MTCache system does store is select/project materialized views defined over the backend database. When an MTCache system receives a SQL query, its query optimizer determines whether the query can be answered, in whole or in part, using the local materialized views, and whether it is cost-effective to do so. If so, the query is answered locally. If not, some or all of the query may have to be sent to the backend DBMS for processing. Updates, insertions, and deletions are always forwarded to the backend DBMS. As the backend DBMS is updated, relevant changes are sent by the backend to MTCache, which uses them to incrementally refresh its materialized views. However, because the changes are propagated lazily, MTCache's materialized views may be stale.

The paper reports the results of some experiments based on the TPC-W benchmark. Not surprisingly, MTCache was more effective for the TPC-W browsing workload, which has few updates, than it was for the TPC-W ordering workload, which has many.

SUMMARIZED BY: Kenneth Salem