Database Cache Management System
H. V. Chainani1 S. S. Kale2 R. R. Sarad3 Prof. R. R. Karwa4
1,2,3UG Scholar 4 Assistant Professor
1,2,3,4Department of Computer Science & Engineering
1,2,3,4PRMIT & R Badnera, India

Abstract—The users of web applications are increasing day by day. As the request increases the database trips also increases. As a result, the time required to fetch data from database increases, which in turn reduces the performance of web application. Under commercial available operating systems data can only be manipulated in the main memory storage. Therefore part of the database has to be loaded into a main storage area before modification and has to be written back to the disk after modification. This main memory storage technically referred as cache memory is used to store frequently asked data for faster retrieval by minimizing the time for retrieval queries. We have to apply various mechanisms policies so the user will get maximum cache hit. Our system in collaboration with airlines reservation application makes use of SQL Cache Dependency and Page Output Cache to deliver optimal and faster results.

Key words: Cache Management, Buffer Pool, SQL Cache Dependency, Page Output Cache

I. INTRODUCTION

The users of web applications are increasing day by day. Therefore the performance of the system is decreasing. As the request increases the database trips also increases. As a result, the time required to fetch data from database increases. Therefore to reduce no of database trips, we can use cache memory to store important data on server. So that when user request for important data, the data will be checked in cache. If the requested data is presented in cache memory, the data will be delivered from cache. Otherwise the data will be fetched from database and delivered it to user. A single database trip is more costly than a cache access made on the other hand. Cache memory is a smaller and easily accessible memory that has least cost required when it is accessed. This in turn increases the efficiency of the system as faster results are obtained as compared to the existing system also the cost of data retrieval reduces comparatively Only storing the data in cache memory is not enough for cache memory management is more important as to achieve more and more cache hits as possible. For this it requires that optimal use of the memory is made. Only frequently accessed data should be present and that it should be up-to-date.

Using SQL cache dependency we cache data and create a dependency on a database table or row change. When the data alterations are made only then the cached objects based on that data are annulled and removed from the cache. From then on every next request made by the user for that item from the cache, if it is not present there in the cache; the system then re-adds the restructured version to the cache and be certain that users have the newest data.

SQL cache dependency is also open for the page output cache. Some static pages that show information about a particular aspect are set to that page's cache policy to be a SQL dependency as DBA would for an item add manually to the cache. The page is then stored in the cache until the table or row that it was dependent on changed. When the data changes, the page would be recreated and stored in the output cache again.

II. LITERATURE REVIEW

In 1984, Wolfgang Effelsberg and Theo Haerder in the implementation of a database buffer manager within a DBMS will enhance the Speed and performance of database Buffer pool also the interface between database and server side application. They have explained the interface requirements of a DBMS buffer manager and introduced the concept of fixing pages in a buffer to prevent their uncontrolled replacement. The authors have provided Solution for this that is different buffer allocation algorithms and explained their relationship to page replacement algorithms. Hash technique on buffer information is one of the best alternatives for the buffer storage and search function for data in database [1].
They have also shown implementation of a database buffer manager as a component of a DBMS and the interface between calling components of higher system layers and the buffer manager.

In 2015, Priti M. Tailor and Morena D. Rustam, have shown a research study to minimize the disk I/O by implementing Temporary purpose memory which will store only Frequent data called caching mechanism for object oriented databases. Persistence is the property of an object through which its existence transcends time and space which is becoming important factor for the field of database. They have stated that Object oriented database is a better option for object persistence just in case of composite object modelling and vast databases. Improvements are possible in internal storage structure by using static clustering; dynamic clustering; buffer replacement; and pre-fetching techniques together in a complimentary manner [2].

In 2010, Mustafa Canim, George A. Mihaila Bishwaranjan, Bhattacharjee Kenneth A. Ross, and Christian A. Lang presented a paper that dealt with the configuration of SSD as a layer RAM and Hard Disk in the database system. Because the Solid State Drives much faster access to data as compared to the regular Hard Disk Drives. They have presented a technique in their paper to use these Solid State Drive as a medium for caching the Data. Populating the cache with hot regions i.e. the area where from the most frequently data is accessed are created pages of in the memory. Due to which the I/O operations of database access subsequently decreased. There are two drawbacks to such an approach. First, a user must explicitly run a profiling tool to gather statistics, and then perform potentially expensive physical reorganizations as data is moved to/from the SSD. Second, decisions are made at the granularity of an entire table or index. In some cases, only part of the data in a table (e.g., the most recent data) may be frequently accessed, and one might hope to put just fragments of the table on the SSD. [3]

III. EXISTING SYSTEM

The exchange of data between the disk and the main storage is normally done through I/O operations also the data that is frequently required gets cached in the systems cache memory. The cache memory of the system corresponds to 40%-50% of the RAM available in the system. But this too is subjective as the cache memory may get reduced as more and more applications are installed over the system. Due to which physical access to the database becomes mandatory as cached data gets erased again and again.

IV. PROPOSED SYSTEM

Physical access made to the database is expensive then the access made to the cache memory to retrieve the same data. The main goal of a proposed system is the minimization of the physical access of database. This goal is accomplished by using cache memory more efficient as we apply pooling time concept to refresh and refill it after certain amount of time itself. Also data should always be brought from cache only not database this makes it very fast and reduces the retrieval time.

Proposed system’s objectives are:

- To improve application efficiency with the help of cache management technique.
- To implement SQL cache Dependency and Page Output Cache in and e-commerce application system.
- Implementation and analysis of caching techniques.

V. IMPLEMENTATION DETAILS

For analyzing whether the cache memory retrieval provides optimal results or not we had implemented a module that gives exact time difference in microseconds for database trips and cache accesses simultaneously. When the application is initially started the time taken to retrieve the data from the database as well as the cache is the comparatively equivalent as cache access made at that time can only be a hit when it has some data that is fed from database trip made at first. After which every time another access is made the data is brought from the cache. After the initialization the cache keeps on getting filled with all the data that is required whenever it is asked for by the user requests. The figure below shows the comparative time taken if the data had been brought from the database.

The time differences shown explain how faster the cache is in comparison to the database access A comparative chart showing the access time in microseconds for 12 data retrievals is shown below. This explains the effectiveness and optimal time usage that increased from earlier system to now.

VI. CONCLUSION AND FUTURE SCOPE

Database Cache management System is important to the performance of any DBMS. A cache memory is a fast local memory used for storage of a more distant, larger and slower memory in order to improve the average memory access speed. We have therefore to reduce no of database trips use cache memory to store important data on server; So that when user request for important data, the data will be checked in cache. If the requested data is presented in cache memory, the data will be delivered from cache. Otherwise the data will be fetched from database, the cache will be refilled with it and data will be delivered to user from cache itself. Also the page output that is cached helps to deliver faster results when outputs of web pages are cached. So we conclude that the performance of the Database and Data delivery is improved by implementing the SQL Cache Dependency and Page Output Cache in web applications.
Improvement can be done if we are able to build caches from views as well for which we use XML files in the current scenario.

- In addition to this we can also add concept of other tools such as Web Services which will allow bookings within the application itself also it will help in finding out how feasible the application is with databases of two entirely different applications and Data dictionary cache in order to improve performance.

- High-end solid state drives (SSDs) offer much quicker access to data as compared to regular hard disk drives. A new technique can be implemented for using solid-state storage as a caching layer between RAM and hard disks in database management systems [3].

REFERENCES


[7] Sejin Park New Computing Lab SK Telecom and Chanik Park Dept. of Computer Science and Engineering POSTECH, “FRD: A Filtering based Buffer Cache Algorithm that Considers both Frequency and Reuse Distance”.


