ORACLE
NOSQL DATABASE

Oracle NoSQL Database For Time Series Data

ORACLE WHITE PAPER | DECEMBER 2017

ORACLE

Introduction

As massive amounts of data are being created with a need to store and analyze this data, there needs to be efficient systems in place that can deal with this flood of information. For both the Internet of Things (IoT) and Big Data environments, there are convincing reasons to simplify and reduce the amount of data stored in databases that can then be further analyzed with other software. Examples include the ability to remove older data that is not relevant or expiration of data due that is no longer of value.

While not all data is associated with a time stamp, many types of data are only meaningful when associated for a fixed period of time. For data that is associated with a time stamp, developers can use that information for creating applications that can analyze past behavior or predict future behavior. However, the storage, querying, and use of "old" data that no longer has meaning or value can affect a smooth running analytics application and could delay the decision making process due to inefficiencies across the system.

Time Series Examples

A common domain where most of the information generated with a timestamp are sensors. For example, a sensor monitoring the temperature or light in a building area is an important data point when the time along with the temperature or amount of light is recorded. The data would consist of a time stamp + data, which is approximately double the amount of data than just storing the measurement alone

In a manufacturing operation consisting of an assembly line, there are numerous data gathering points which may be captured and stored with an associated time stamp. This data may be more than just a sensor reading and could include images or video (for example of a windshield being placed within a car frame). While this data can be valuable when determining possible failure modes, most of the data does not need to be stored in a database for long periods of time. In this example of recording a windshield placement by a robot, once the car has passed the assembly line quality assurance process, this large amount of data could be discarded.

Time series data can also be generated by financial events, factory automation, home appliances, utilities and mobile devices. Time series data is generally voluminous – massive amounts of "small" records. When this data is captured and stored over time, the total amount of data can easily exceed the storage capacity of a system.

A Data Management Solution with Time Stamps

An overall solution that integrates many types of data, both for rapid decision making and historical comparison, would need to have the capacity to quickly ingest the data as well as secure the data in a reliable database. Data would flow from the edge of the network (sensors) into and through a low latency database and then could be stored for longer term in a relational database with the addition to other information that may be obtained over time.

For example, Oracle NoSQL Database could be used as the short term database with its ability to reliably store massive amounts of data in the single digit millisecond range. After removing aged-out data, the remaining relevant

data could be stored in a relational database for further processing. With this architecture, the volume of information that needs to flow between databases is reduced.

Time-To-Live (TTL)

The Oracle NoSQL Database has an API feature which allows the developer to set the amount of time that data remains in the database. Time-To-Live is a mechanism that allows the developer to specify a time period on a record such that the removed will automatically be deleted from the system once it has expired.

The developer can specify a time limit, in hours or days, for a record to live. Queries can then filter out expired records and the NoSQL database will reclaim space. This feature is useful in many different applications where data is only needed for a limited amount of time; for instance, Web applications that want to purge profile data for users that are inactive for long periods of time. Another scenario may involve tracking the buying patterns of a customer in a chain of stores. Whenever the customer purchases an item, a time period to retain that data could be set, after which it would be assumed that the customer no longer shops in that particular chain of stores, and the customers' buying preferences could be removed from the database.

TTL is an efficient way of doing these types of operations in terms of system performance for cleanup and ease of implementation for programmers.

Example

The following example shows how to set a Time To Live of 5 days for the row of data. This would be similar to how an application that uses TTL can set the expiration date for a row of data.

```
package kvstore.basicExample;

import oracle.kv.KVStore;
import oracle.kv.table.Row;
import oracle.kv.table.Table;
import oracle.kv.table.Table.TimeToLive;
import oracle.kv.table.TableAPI;

...

// KVStore handle creation is omitted for brevity

...

TableAPI tableH = kvstore.getTableAPI();
Table myTable = tableH.getTable("myTable");

// Get a Row instance
Row row = myTable.createRow();


// Add a TTL value to the row
row.setTTL(TimeToLive.ofDays(5));
```

```
// Now put all of the cells in the row.
row.put("item", "Sensor");
row.put("description", "Temp and Humidity in Office");
row.put("temp", 75.3);
row.put("humidity", 0.49);

// Now write the table to the store.
tableH.put(row, null, null);
```

At some point in the future (before the 5 day expiration) the application may wish to extend the number of days that the row is available for query. This is simple to accomplish and can be done by simply retrieving the row instance, changing the value of the Time To Live of Days to the new value and storing the row back into the database.

Additionally, if application needs to set the data to never expire, they would use the following:
row.setTTL(TimeToLive.DO_NOT_EXPIRE);

Benefits of Managing Expiration of Data based on Time

As additional data is stored into a database, not only does the size of the database grow in terms of resources used, but search time may increase as well. Data that is no longer needed for queries should be removed permanently from the database in order to retain high performance and adequate free storage capacity. In the figure below, data that is stored has various TTL values. As can be seen as we move towards the right on the TIME axis, records get removed from the database based on the TTL value, and the size of the database will shrink enabling a new record to be inserted.

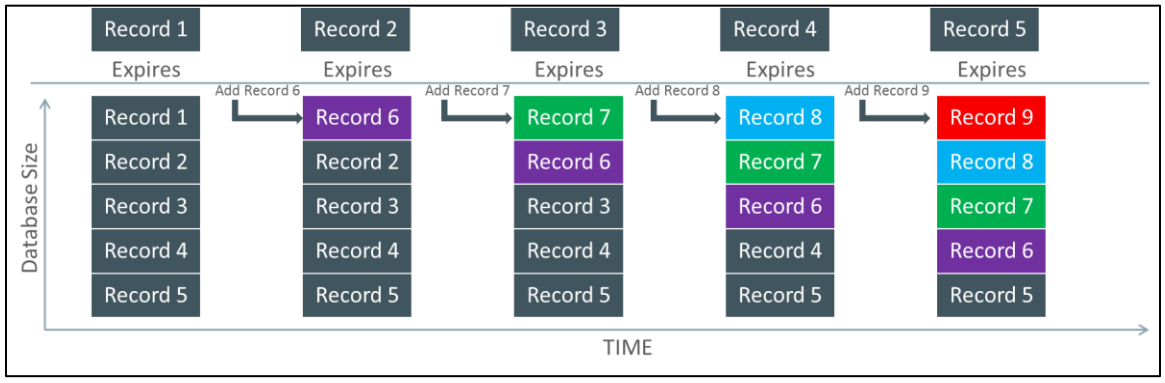

Figure 1 - Additional Records Inserted as old records expire

Oracle NoSQL Database Designed and Optimized For Time Series Data

When dealing with voluminous amounts of time series data, it is important that the underlying database is designed to provide fast response with predictable latency. Oracle NoSQL Database's "smart client driver" always routes read requests to the most efficient node; this means that any record fetch can be satisfied in a "single request message/single response message" manner, which ensures predictable low latency.

Oracle NoSQL Database supports secondary key indexing so it is possible to retrieve records efficiently by specifying a predicate on a secondary key column. Oracle NoSQL Database secondary indices are partitioned and shard-local – there's one index partition per shard, and each index partition only refers to the records on that shard. This means that secondary indices are updated transactionally, along with the records that they index. Secondly, a secondary index lookup can be executed very efficiently by accessing all the index partitions (on all shards) in parallel. Consequently, secondary index access is both efficient and always returns the correct and complete result set. In some other databases, the secondary index is updated asynchronously, which means that secondary index queries may result incomplete or out-of-date results..

Summary

Oracle NoSQL Database offers an API that allows a developer to determine how long a record will remain in the datastore. When used, this feature helps to reduce the amount of data that is stored in the database as well as decrease search and scan times of the data. Since some data only has value for a certain time period, this feature helps application developers to manage the amount of relevant data stored and increase performance.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services

Oracle NoSQL Database Ideal for Time Series Data
November 2017
Author: Michael Schulman
Contributing Authors: Ashok Joshi