Mark Wagner

SDEV350

Homework 02

4/10/2019

Comparison of Oracle 12c and DynamoDB

VS

Oracle has been a technology superpower for quite some time. Particularly known for their database with the same namesake. What we use for class, Oracle version 12c is the flagship Relational Database in the world. It works by organizing tables on a central schema model and interacting with them by Structured Query Language commands from a command line or an application-programming interface.

Dynamo allows for a more flexible layout as rather than objects attached to a set, it is composed of many loosely linked JSONs. Dynamo also is designed to run fluidly with other AWS services like RDS and Lambda, making it more scalable than ever both physically and programmatically. This is, at it's core because dynamo is a database service that acts like a database but is actually composed of X number of decentralized databases.

I chose to compare to Dynamo because compared to RDS, RedShift, PostGreSQL or HadoopFS, dynamo is the most up and coming database, plus it ties in with my other AWS knowledge which benefits development in some ways which I will cover below.

As far as the overall structure, Oracle databases use a proprietary model known as the Object Relational Mapping. Dynamo a partition key with which it retrieves values, column sets, or semi structured JSON or xml documents containing related attributes. There are several smaller components that are vastly different.

Process Architecture:

Oracle processes consist of client, background and server. These respectively control the application controlling the database, starts and stops instances, provides backup, and processes the requests. There can also be slave processes that run in addition to the latter two.

Dynamo uses dynamo streams to record interactions with data on the bank table.

Application Architecture:

Oracle uses a mandatory Client – Server setup. It also can distribute processing for multiple clients and/or over multiple servers of the same database.

Oracle also supports multitier processing where a thin client connects to an application server (backend in the case of web) and that server then connects to the database server.

Service oriented architecture:

Oracle includes Global Service Managers as part of Global Data Services. GSM provides load balancing, protection from faliure and centralized management. It also collects performance metrics, manages across regions, and creates run-time load balancing advisory alerts. The command for this is gdsctl.

The GDS catalog is a repository of data relating to the configuration.

Shared server architecture dispatches incoming session requests to a pool of shared server processes. The program interface is the software layer between the oracle database and the program. The software complexity of oracle is quite vast. There is even a process called the VKTM, the virtual keeper of time, which handles requests for both time stamping info and smaller interval time for measurements.

One big difference between SQL and NoSQL database applications is the language they use. In the case of Dynamo, you tell the DBMS what you want to achieve and it formulates the system for you.

Declarative Language	Procedural Language
User Describes What To Be Done	User Describes How To Do It

Many of the statements used to interact with the databases are actually similar.

Below is a chart of commands and in both Oracle and Dynamo.

Function	Dynamo	Oracle
Create a Table	CreateTable	Create Table
Describe a Table	DescribeTable	Describe X
Delete a Table	DeleteTable	DropTable
Insert an Item	PutItem	Insert Into*
Change Item	UpdateItem	Update Set Where
Delete an Item	DeleteItem	Drop / Delete

As you can see above, many of the commands are identical.

The similarity is by intentional design and only possible because Dynamo is so good at interpreting the command and making the necessary adjustments to the tables. This had to be built into the governing software of DDb.

Security is another topic:

Oracle secures the users with SSH level encryption and then by username and password. Dynamo, taking a note from AWS, functions via Integrated Access Management. This is extremely useful if an employee is tasked with managing a database and a parallel application running on an instance. Dynamo access can be granted by table, by permission types, and users can even log in using Oauth on facebook or Google in lieu of an Amazon ID. Oracle keeps most of the verification isolated from the database.

Memory and Physical Hardware:

Deep down hardware wise, Oracle 12c and Dynamo are more similar than they are different. They both use hard drives (likely Solid State Drives in oracle case – absolutely Solid State Drive for Dynamo), they both have physical servers somewhere. The key differences being that if you want to merge two databases or set up an additional server, this must be done manually and not without hardship in 12c (not to say it is not done daily). Amazon is masterful at making life easy for IT companies who are willing to pay to play. In other words, they commoditize the Infrastructure that you once had to pay someone to set up just for you.

Application network:

This is where things get very different. Dynamo is stateless, but Relational Database Management Systems require an opening of communication on a port, with a password, yada yada. Once a connection is opened, statements can be made and queries pulled, etc.

With Dynamo, everything is done with get and post requests over https which means the URL parameters to gain access must be passed every time an interaction occurs and a 200 response is returned every time a reaction occurs.

This is actually more secure in terms of mitigating the risks of connection session hijacking. Below is a table of the differences:

Area	Oracle	Dynamo
Interacting	CLI, API library	API, CLI, Console
Connecting	Starts, Maintains, Terminates	HTTPS Get/Post
Security	Must authenticate before connecting	Every request carries signature
Authorization	User Based/App Based	IAM based
Sending Request	Issues SQL statement, RDBMS checks and then executes	It just executes (post-setup of course)
Receiving Responses	Result of Statement or Error code	HTTP response or http response code and custom message

This chart was translated from the AWS docs on dynamo DB but it's information is in keeping with the paragraphs above.

Data Types of Dynamo:

- Scalar – Number, String, Binary, Boolean, and Null.
- Multi-valued – String Set, Number Set, and Binary Set.
- Document – List and Map.

Data Types of Oracle:

- Char
- Varchar
- Number
- Date
- & Many more except Multi-Valued and Document

Multi Values data types are indicated to be unique and Document data types or maps are not used in non-object RDBMS.

Logic of Data Model:

Dynamo uses the models Tables, Items, and Attributes. From 12c, we know what tables are, but items are collections of attributes. Attributes, in turn, are basic units of information, such as Key Value Pairs. The difference between the tables in Oracle and the tables in Dynamo are that the ones in dynamo do not have fixed schemas. Dynamo requires a primary key. (actually I think they both do).

The two keys available in Dynamo are:

Hash type- if attribute uniquely identifies item, it can be primary

Hash and Range type- built upon the hashed key and the range key in the table – hashed index on the hash primary key attribute and range sort index on the range primary key attribute. – This statement I will not understand until I use it but it is the bread and butter of Dynamo DB.

Indexes:

Dynamo:

Local Secondary Index – (range key mandatory)

Global Secondary Index – (hash or hash+range)

GSI spans multiple partitions and go in separate tables. These are managed by the Global Service Managers. The hash key for this GSI will be used for partitioning.

Oracle:

Primary Index on a Tablespace and none of the rest of this. You select the keys.

Additional Info:

Dynamo can be used with Elastic Map Reduce to process large datasets into simpler larger ones and Redshift for data warehousing. This a convenience that doesn't exist with Oracle.

Partitions:

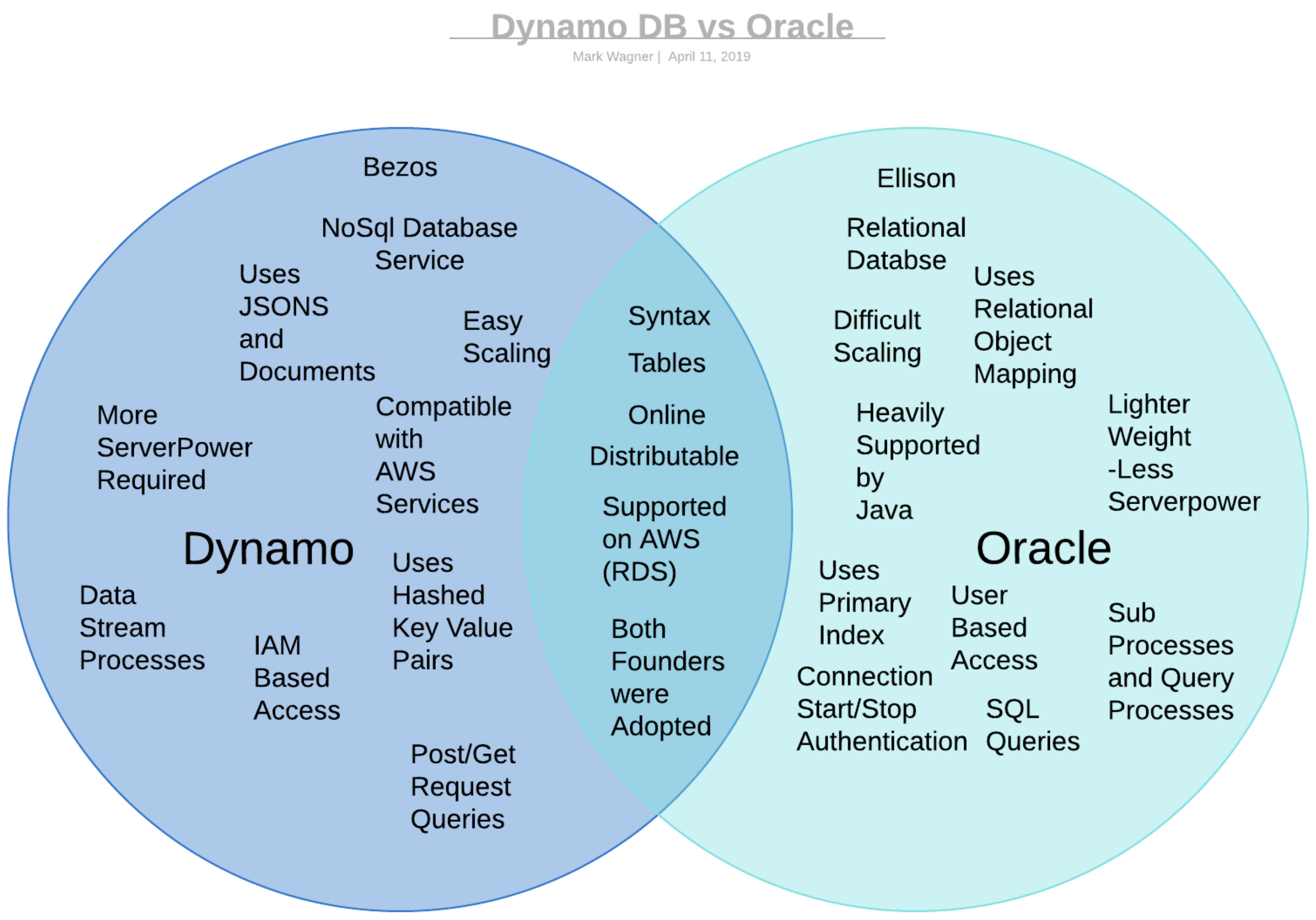
Partition logic in Dynamo is dependent on data size and throughput. Here is an info piece that explains how many partitions you will have.

$$\#\text{ of Partitions}_{\text{(For throughput)}} = \frac{\text{RCU}_{\text{ for reads}}}{\text{3000 RCU}} + \frac{\text{WCU}_{\text{ for writes}}}{\text{1000 WCU}}$$

$$\#\text{ of Partitions}_{\text{(For size)}} = \frac{\text{Table Size in GB}}{\text{10 GB}}$$

$$\#\text{ of Partitions} = \text{MAX}\left(\#\text{ of Partitions}_{\text{(For size)}} \mid \#\text{ of Partitions}_{\text{(For throughput)}} \right)$$

Venn Diagram of Specs & Traits



Capacity: A large part of the push for NoSQL is the recent flood of computing power for sale at scaling prices. This has made it possible to do what Dynamo offers, where as ten years ago, none of this would have been feasible. NoSQL is inherently less efficient, using key value pairs, and even storing and searching documents to find information, however when computing power is no longer a question, it can be easier and more flexible to order it this way. By this method, you can merge object oriented databases and datasets of different types relatively easily.. okay maybe I am starting to talk about hadoop! In any case, Dynamo and NoSQLs akin to it are tools for big data tasks. Oracle will not go anywhere soon as it is still a powerhouse for the majority of application database needs.

Citations:

A. (2019). Accessing The Database. Retrieved April 10, 2019, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.Accessing.html

AlfredBaudisch 1064, M., & Mooredsmooreds 1113. (2017, February 11). DynamoDB Streams with Lambda, how to process related messages in order? Retrieved April 10, 2019, from https://serverfault.com/questions/826109/dynamodb-streams-with-lambda-how-to-process-related-messages-in-order

Makumbi, A. (2016, December 17). Database Comparison: 12c vs Dynamo. Retrieved April 10, 2019, from https://makumbi.github.io/output/blog/2016/db-comparison-oracle-vs-dynamo.html

Patra, C. (2019, March 22). Amazon DynamoDB: What it is and what you really should know. Retrieved April 10, 2019, from https://cloudacademy.com/blog/amazon-dynamodb-ten-things/