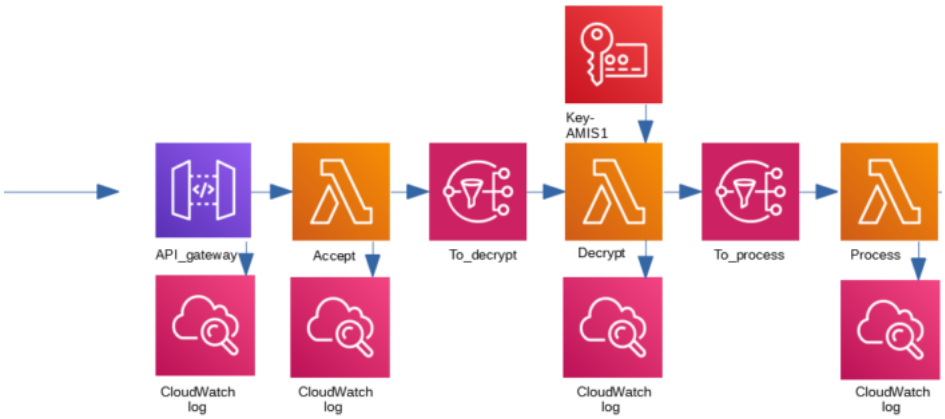


AWS Shop: DynamoDB, the AWS NoSQL database

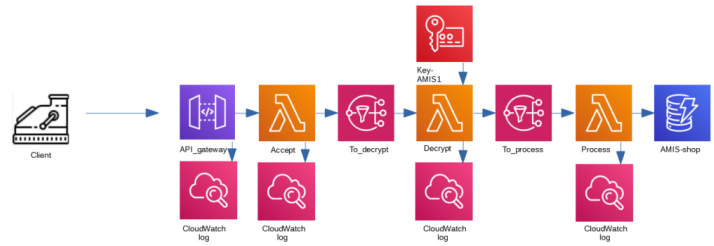
0

BY FREDERIQUE RETSEMA ON MAY 5, 2020

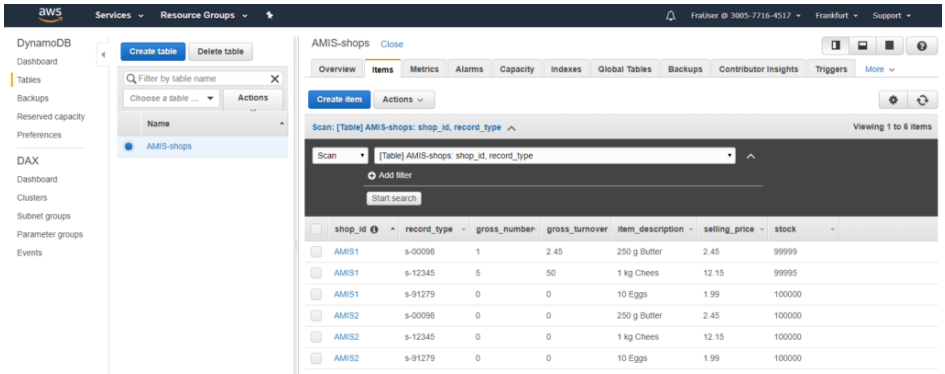
AWS, CLOUD, DATABASE, NOSQL, SERVERLESS

Introduction

The ultimate goal of our shop application [1] is to update the AMIS-shop table in the DynamoDB service. In this blog, I will tell a little bit more about DynamoDB.



DynamoDB is the NoSQL solution of AWS. The way we use this table in our example is straightforward: the process Lambda function will use an update statement to update the record with the shop_id and the item number:



When you play along and don't remember how to get in this screen, please see the first blog in this series [1]. The way we use this table is the same as you would do in a SQL environment. You might ask yourself: then what is the difference between SQL and No-SQL?

ABOUT AUTHOR

Frederique Retsema

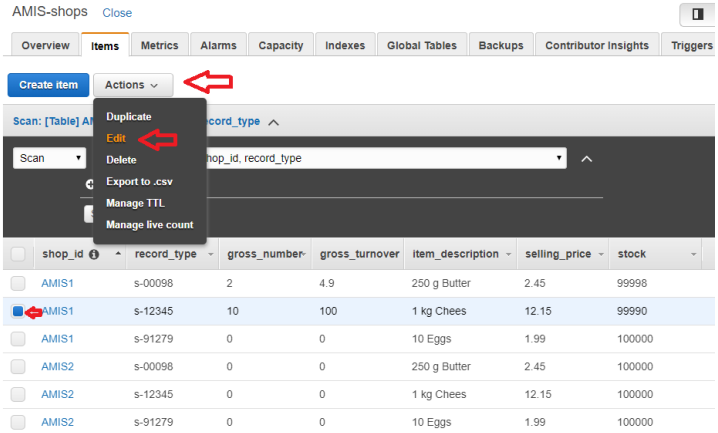
Frederique Retsema is active in IT si 1993. Senior Consultant and develop on diverse areas including SQL and Java. She likes work with automation tools like Bamboo, Jenkins, Ansible, Terraform and CloudFormation.

View all posts

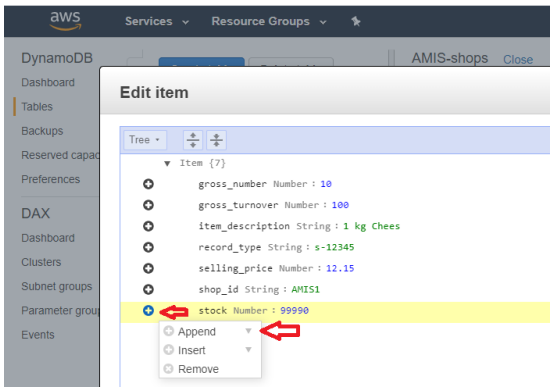
Difference between SQL and No-SQL

Let's look at our table design. We have a shop_id and we have item numbers, and then information about that combination. The way it looks now, every item has the same attributes. In a SQL database, the attributes are part of the table. This means that an attribute always has to be filled, either with a value or with NULL (when we don't know the value). In a NoSQL table, the attributes per item can change.

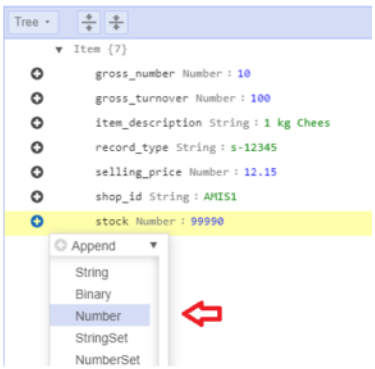
As an example, think about our shop. In our example, we stop at the database table. In real life, when we are nearly out-of-stock, we would like to order more from our suppliers. Let's try this out: click on the checkbox before AMIS1 and record_type s-12345. When you did so, click on Actions and then on edit:



In the next screen, click on the plus next to stock, and then click on the down-arrow next to Append:



Click on Number:



Let's call the new attribute ordered, and let's say we ordered 200. After you filled in this attribute, click on Save:

ADF Performance Monitor: Thread Wait and Bl…
Last week we had a new version of the ADF Performance Monitor available – version 9.5. In this
technology.amis.nl

AMIS Conclusion
@AMISnl
New Article : Windows Sandbox with Chocolatey, Scoo Git, VisualStudio Code and more ift.tt/3b2c1gN by Luc Jellema

Windows Sandbox with Chocolatey, Scoop, Gi…
I recently wrote an article to introduce Windows Sandbox. I explained how the Sandbox can have
technology.amis.nl

AMIS Conclusion
@AMISnl
New Article : AWS Shop: DynamoDB, the AWS NoSQ database ift.tt/2Wv582f by Frederique Retsema

AWS Shop: DynamoDB, the AWS NoSQL data…
Introduction The ultimate goal of our shop application [1] is to update the AMIS-shop table in
technology.amis.nl

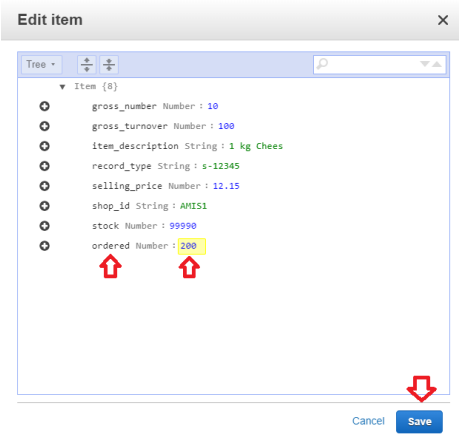
May 5, 2(

AMIS Conclusion
@AMISnl
The latest The AMIS Oracle Technology Weekly! paper.li/AMIS_Services/… #bastionhost #softwaredevelopment

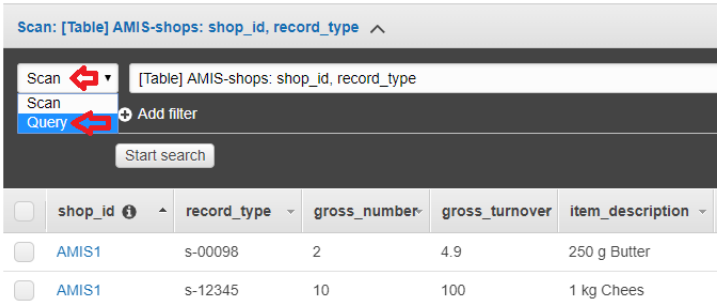
Oracle Cloud Infrastructure and SSH Keys – J…
martinberger.com In our Trivadis Oracle Cloud Infrastructure training environments, we never use
paper.li

May 4, 2(

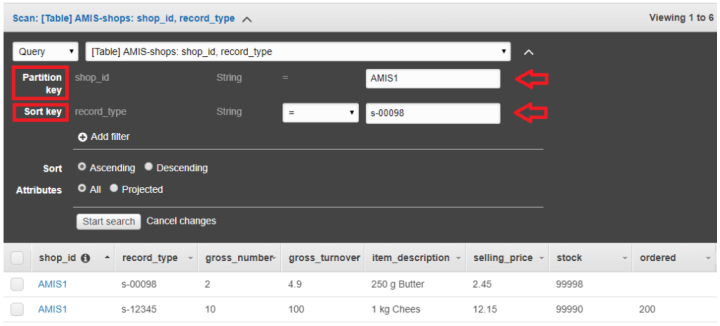
AMIS Conclusion
@AMISnl
New Article : Deploy Angular and Node.js webapp in Azure Pipelines (part 5) ift.tt/35shQTI by Joost Luijben



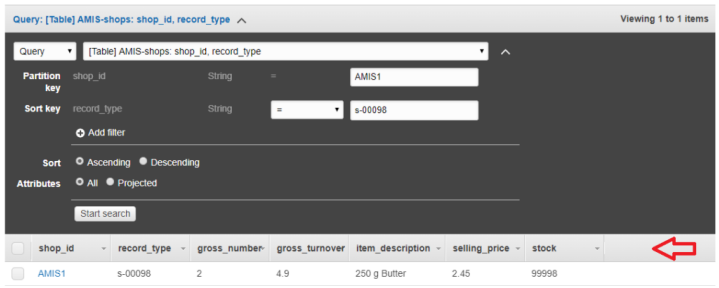
When you look at the overview screen that follows, then it seems that every record has an ordered attribute and that just one record has a value, but in reality the ordered attribute isn't present in the other items. To show you this, click on Scan, and then on query:



In this table, there is a partition key (shop_id) and there is a sort key (record_type). I will explain more about partition keys and sort keys, but for now fill in AMIS1 as partition key and s-00098 as sort key. We expect to see only the first record:



When we click on Start search, we see just the one record we searched for. In this record, the ordered attribute is not present:



In a NoSQL database, the names and types of the attributes are stored in the record of the item, not in the table.

Partition keys and sort keys

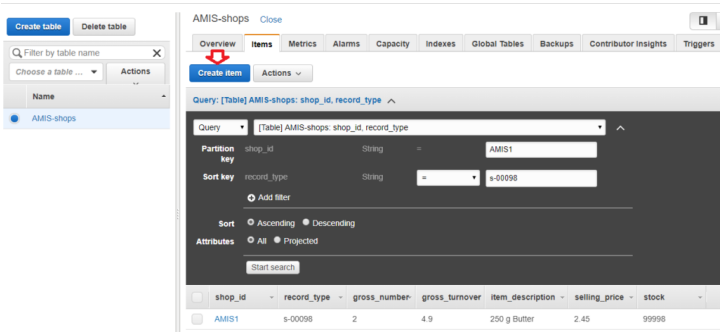
This table has shop_id as a partition key, and record_type as a sort key. Partition key means, that the table will be ordered by partition key when the data is written to a disk. Tables are stored in blocks of data, those blocks are called partitions. When a partition is full, then a new partition will be created.

The partition keys are used to know what data is stored on what page. The sort key is used to search within the partition: when a specific item is searched for by giving the partition key and the sort key, and in the partition a sort key is found that has a higher value than the value that is searched for, it is clear that the value that is searched for isn't in the table.

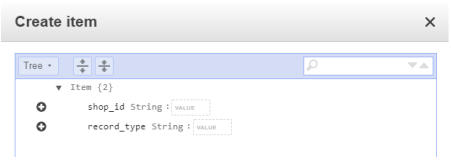
This means, that the combination of shop_id and record_type must be unique. When we use our table to store the stock of our shops, combining the shop_id and the item number makes it unique. When you try to insert a record with the same partition key and sort key, then the old value will be overwritten.

Storing different kind of data in the same table

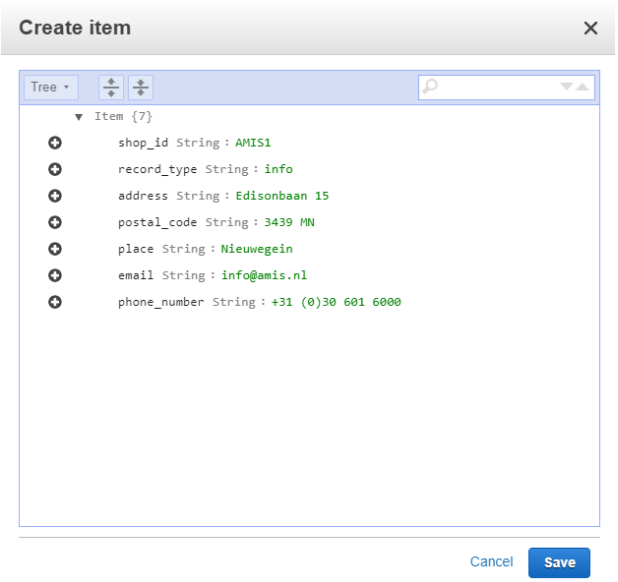
It is possible (and in NoSQL environments not unusual) to store multiple types of information in one table. Let's try this out: let's add information about the shop itself in this table. Let's store the address, postal code, place and telephone number in the same table: click on Create Item:



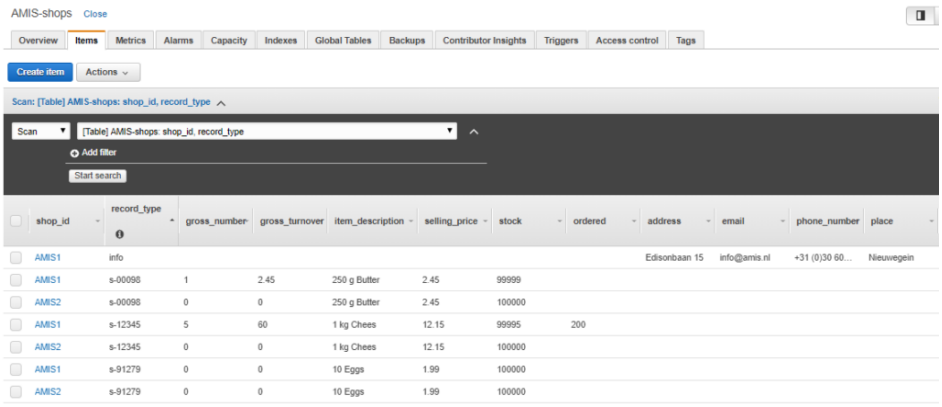
You see that just two attributes are present: the partition key and the sort key. These two attributes are mandatory.



Let's fill in this record:



When you press save (and then on Scan, and search all), you will see that the info record appears on the screen. Mind, that the order in which we entered the attributes is not the order in which the data appears on the screen. In NoSQL databases, the order in which you get your attributes back is undetermined.



When you want to learn more about the design of NoSQL tables, I can recommend the LinuxAcademy training for that [2].

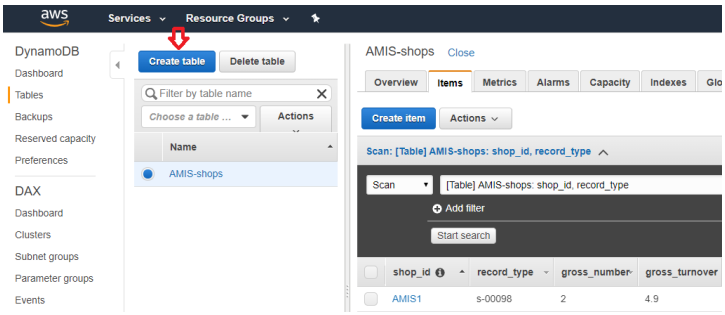
Getting data from NoSQL databases

In the previous examples, we sometimes stepped from “Scan” (to get all data from the table) to “Query” (to get one specific item) and back.

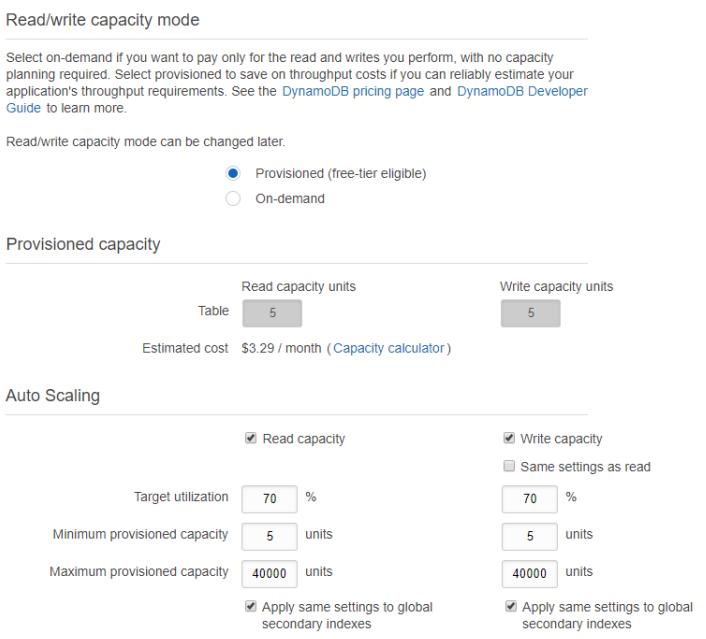
Scan is used as a table scan: you will look at every item and then look if the attributes in the data match your filter. Query uses the indexes: the database is able to search based on your indexes and only the records that match the indexed keys will be looked into for other filter criteria. Using indexes saves time and money.

Capacity

When you create a DynamoDB table, you have to answer some questions about read and write units. Let us look at those questions. Click on Create table:



When you scroll down, you see the settings for Read/write capacity mode:



Read and write request units are used to determine the amount of traffic that is allowed to or from your table. With one write request unit, you can write one Kb of data per second to the table. With one read request unit, you can either read 4 Kb per second, or 8 Kb

per second. When you need “strongly consistent reads”, you can read 4 Kb per second, with “eventually consistent data”, this is 8Kb per second.

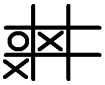
“Strongly consistent” means, that when you get the data, you can be sure that you have all the data that was present on that moment. When you don’t need this, for example because you know that so much data is written that you will not have the most recent results anyway, then you might use “eventually consistent data“. This is cheaper, and gives higher throughput when you read the table.

In the configuration screen, you can first choose if you want to use On-demand mode or Provisioned mode. On-demand means, that AWS will adapt the number of read and write units to what you really need. The initial amount of read and write units will be 2000 write request units and 6000 read request units. This can either grow or shrink based on what the table really needs. The maximum capacity is 40.000 write request units and 40.000 read request units. If you need more, than that is possible – you can ask AWS to increase this number.

With Provisioned read and write units, you pay per hour for the capacity you ask for. You can use a fixed amount of read and write units or you can use auto scaling and give lower and upper limits to the number of read and write units. The minimum number of read and write units is 1.

The disadvantage of On-demand read and write request units, is that it is more expensive per unit that is used than provisioned read and write request units. In test environments, it can be better to have a smaller limit than 40.000 read or write requests per second: when there is a bug which sends lots of data to a table where this was not intended, you will get throttling errors and in that way save money. For production environments, you might still use On-demand tables, because in that way your costs will go up and down with what you really need. I already mentioned the Corona crisis: when your sales drop, the number of transactions drop – and so do your costs.

Play along



I scripted the solution [3]. You can follow along and create this solution in your own environment, see the README.md file in the vagrant directory and see the introduction blog for more information.

Links

[1] This is the fourth blog about this shop. Links to the previous blogs:

– Introduction: https://technology.amis.nl/2020/04/26/example-application-in-aws-using-lambda/

– Lambda and IAM: https://technology.amis.nl/2020/04/29/aws-shop-example-lambda/

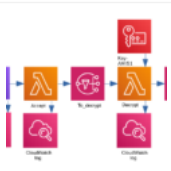
– SNS: https://technology.amis.nl/2020/05/02/aws-shop-about-the-aws-simple-notification-service-sns/

[2] https://linuxacademy.com/course/amazon-dynamo-db-data-modeling/

[3] https://github.com/FrederiqueRetsema/AMIS-Blog-AWS , directory shop-1

Related Posts:

How to deploy InfluxDB in Azure



Example application in AWS using

AWS shop example: Lambda

OOW 2012: Questions to get

Loading Data into Always Free Oracle

ABOUT AUTHOR

FREDERIQUE RETSEMA

Frederique Retsema is active in IT since 1993. Senior Consultant and developer on diverse areas including SQL and Java. She likes to work with automation tools like Bamboo, Jenkins, Ansible, Terraform and CloudFormation.

RELATED POSTS

MAY 2, 2020 0

AWS Shop: about the AWS Simple Notification Service (SNS)

APRIL 29, 2020 0

AWS shop example: Lambda

APRIL 27, 2020 0

A Free Apache Kafka Cloud Service – and how to quickly get started with it

Leave a Reply

Enter your comment here...

This site uses Akismet to reduce spam. Learn how your comment data is processed.