DB Management Systems: Speedrun

Joel Klein – jdk514@gwu.edu

Database Services in AWS

IaaS, PaaS, DBaaS

- Since the late 2000's companies have been shifted away from simply providing software as their service and begun looking at new ways to abstract IT into service based models
  - Infrastructure as a Service (IaaS): Providing a service for managing IT infrastructure without needing to manage the hardware
  - Platform as a Service (PaaS): Providing a service for managing and running applications without needing to manage the infrastructure
  - Database as a Service (DBaaS): Providing a service for managing data without needing to manage software or hardware.

Implementing Cloud Databases

- While there is a specific name for databases in the cloud, DBaas, just like with everything else this is not the only way to do things

- AWS Database solutions:
  - Relational Database Service (RDS):
  - Athena:
  - Dynamo

RDS

Relational Database Service (RDS)

- RDS, despite its name, falls into the PaaS category. This is because it allows people to spin up VM's running DB applications.

- For RDS instances, we dictate the type of machine and the storage. However, the database software is installed and managed by AWS.
  - In addition, backups and restores are also managed via AWS services. Although you can still run manual backup processes

- With this setup, the database works exactly as if you setup the application, but you do not need to manage the software and only need to specify the hardware.

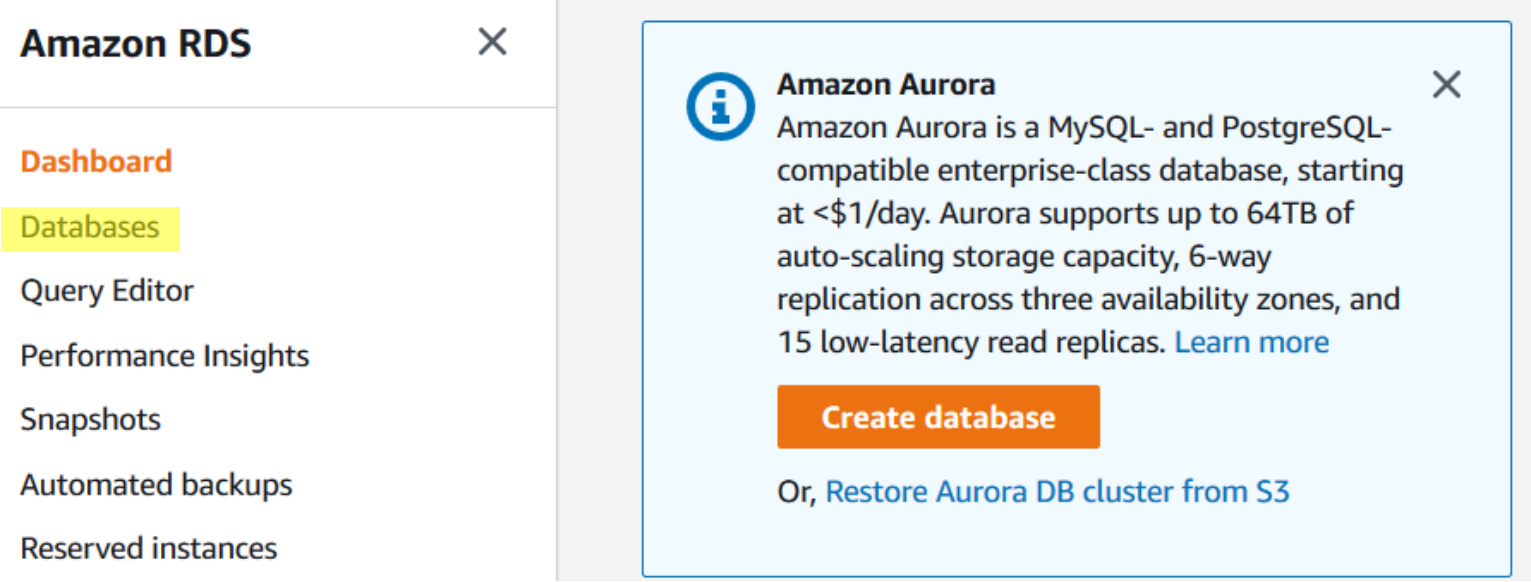
Current Problems with RDS-like Solutions

- While RDS provides an easy and effective way of deploying relational databases, there are some larger consequences.
- Systems like RDS use open source software, like Aurora DB, to provide their managed solutions. While this doesn’t void the warranty of the software’s license (at least currently), it does have a larger effect.
- Most open source software is maintained through paid support and/or features. Tools like RDS disrupt this workflow and are actually suctioning money away from open-source projects.

Accessing RDS

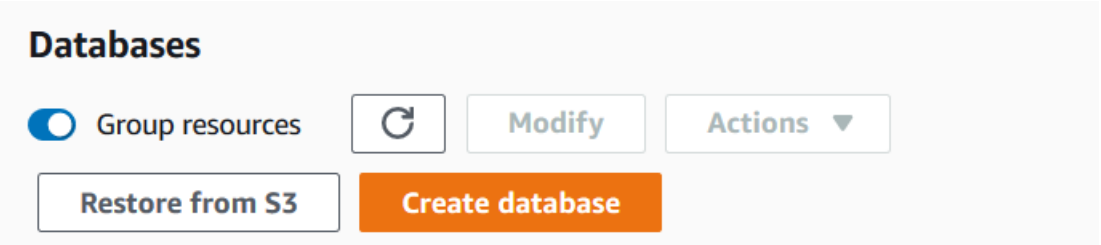
- From the RDS service page we are given a high-level overview of our RDS resources (this is the **dashboard**)

- On the left-hand pane, we can see a couple of options. For us, the primary menu will be the **Databases** page

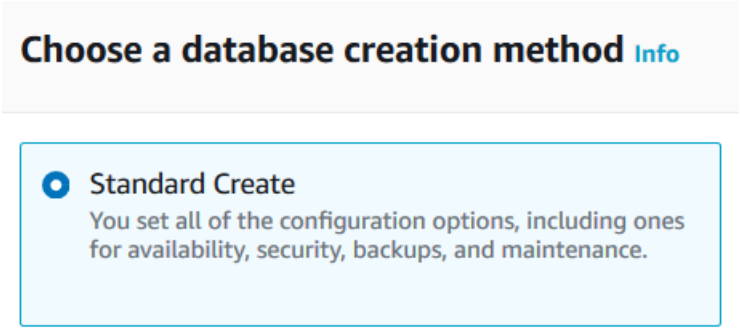


Creating a Database Instance

- From the **Databases** page, we should see the option to **Create database**. This option opens up the workflow to create a database instance



- For most of our use-cases we probably want to do a **Standard Create** as that gives us more control over the settings that influence cost!



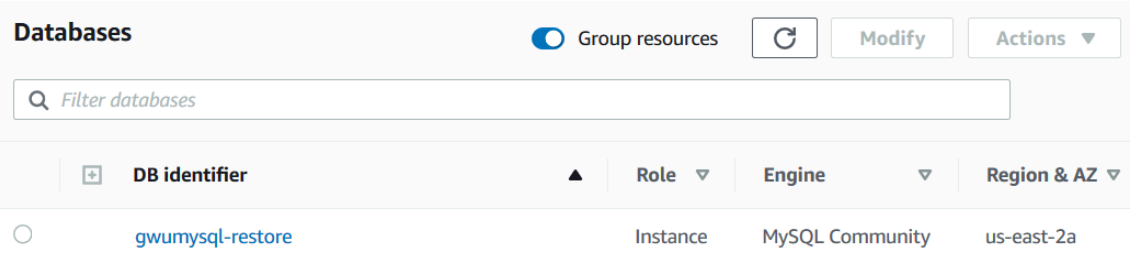
- From here the options begin to grow, as we have multiple databases to choose from, each with their own requirements

RDS for this Class

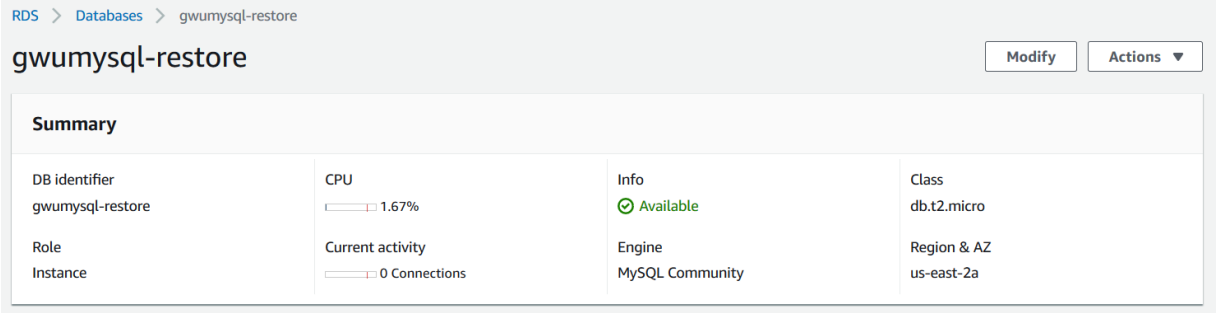
- Engine: MySQL
- Template: Free tier
- DB Instance Size: db.t2.micro
- Storage: General Purpose @ 20GB
- VPC: Class Generated VPC (which opens up MySQL port)
- Authentication: Password

Accessing An Instance

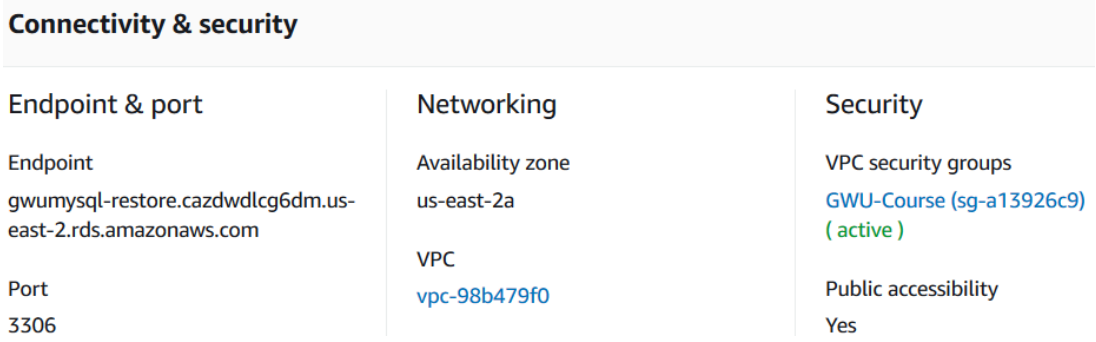
- Going back to the **Databases** page, we should now be able to see our instance.



- By Clicking into our instance, we can get additional details about the running instance.



- From here we can even find connectivity information.



Discuss Pros/Cons of RDS

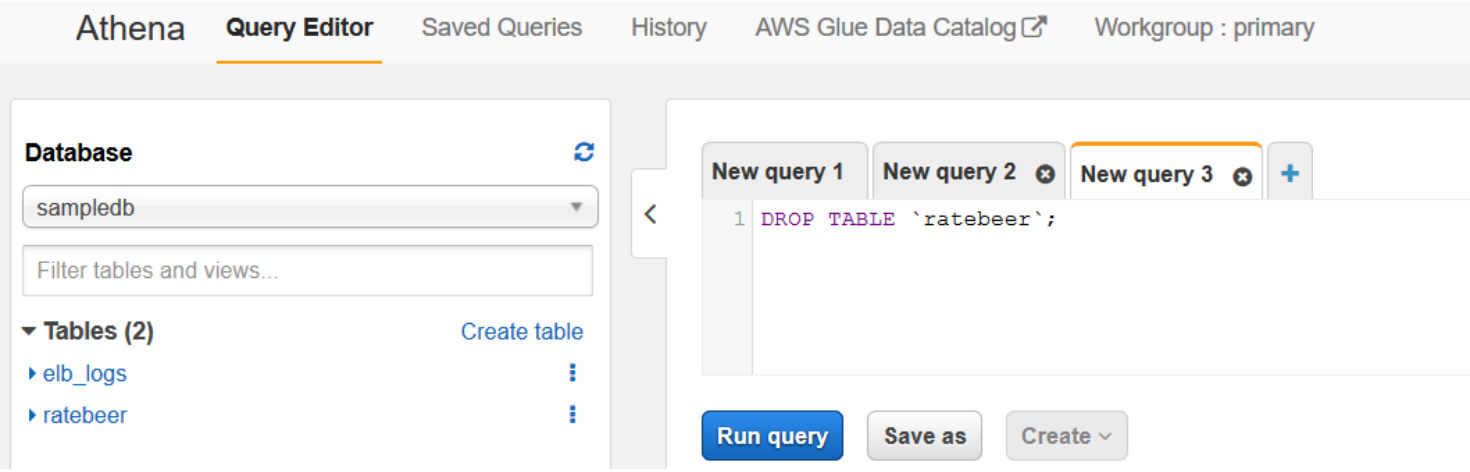
ATHENA

Athena

- AWS Athena is probably most accurately defined as a DBaaS, however it fills a niche that makes it somewhat difficult to categorize
- Athena is a system developed to enable users to query flat files stored in S3. In this way, users do not need to perform any ETL processes to leverage their data.
- However, this system is not nearly as efficient as a database and has numerous drawbacks.
  - NOTE: Parquet helps eliminate some of these drawbacks

Accessing Athena

- The landing page for the Athena service is a little different than other AWS services, as it dumps you directly into a SQL editor.



- The reason for this is that AWS Athena is tightly integrated with AWS Glue
  - AWS Glue is an AWS ETL service that can also help automate the creation of Athena queryable tables

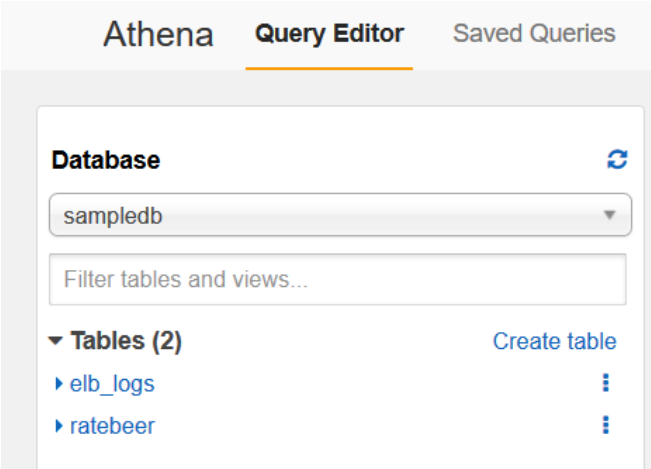
Creating an Athena Table

- Athena tables can be generated a couple of ways:
  - AWS Glue: Glue can scrap S3 buckets and generate meta data/schemas that can be leveraged by Athena
  - Athena Query: You can run SQL commands to generate Athena tables directly
  - Athena Table Creation: Workflow that will generate the SQL for you

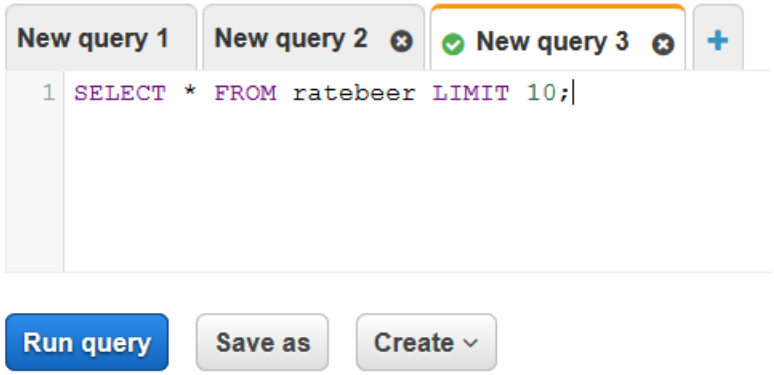
```sql
CREATE EXTERNAL TABLE IF NOT EXISTS sampledb.ratebeer (
    `beer/name` string,
    `beer/beer_id` string,
    `beer/brewer_id` string,
    `beer/abv` string,
    `beer/style` string,
    `review/appearance` string,
    `review/aroma` string,
    `review/palate` string,
    `review/taste` string,
    `review/overall` string,
    `review/time` string,
    `review/reviewer_sn` string,
    `review/review`    string
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
 'serialization.format' = '1'
) LOCATION 's3://hadoop-data-emse6992/ratebeer_small_json'
```

Accessing AWS Tables

- Athena tables are viewable on the left-hand pane on the main landing page



- You can click in on any of the tables, or simply write a query directly in the **Query Editor**.
  - Just like MySQL Workbench, the resultset appears on the bottom of the page



Results

	beer/name	beer/abv	beer/style	review/appearance	review/aroma
1	Livery Pivot IPA	6.25	India Pale Ale (IPA)	3/5	8/10
2	Livery Pipenbock Maibock	10.5	Sour Ale/Wild Ale	3/5	8/10
3	Livery Barrel Aged Maillot Jaune	5.5	Sour Ale/Wild Ale	2/5	4/10
4	Livery Bourbon Barrel Aged Liverator Doppelbock	9.0	Doppelbock	2/5	7/10

Discuss Pros/Cons of Athena

Dynamo (Key-Value Store)

Dynamo

- Dynamo is a Key-Value Database System
  - Key-Value means that records are broken down into a key (identifier) and a value (data)
  - Key-Value may commonly look like Document Databases, however they are typically only queryable against the key/very specific indexes.

- The key design of Dynamo is an easily scalable system, where the data does not conform to a set structure

Terminology

- Table: Collection of Key-Value items
- Item (Record): A group of attributes that is tied to an identifier (a key)
- Attribute: Fundamental data element, our basic unit of information
- Primary Key: The unique identifier of an Item
- Partition Key: Key that dictates how data is distributed throughout Dynamo (really an optimization issue)
- Sort Key: Similar to partition key, but dictates how data is organized (Items cannot share same partition and sort key)
- Secondary Keys/Indexes: A secondary key enables the use of a secondary key for querying data
- Query: A command to extract information from a Database generally limited to the primary key

Dynamo Table Layout

- As mentioned earlier, each Item/Record in Dynamo is comprised of a Primary Key and Attributes (and possibly a sort/secondary key)

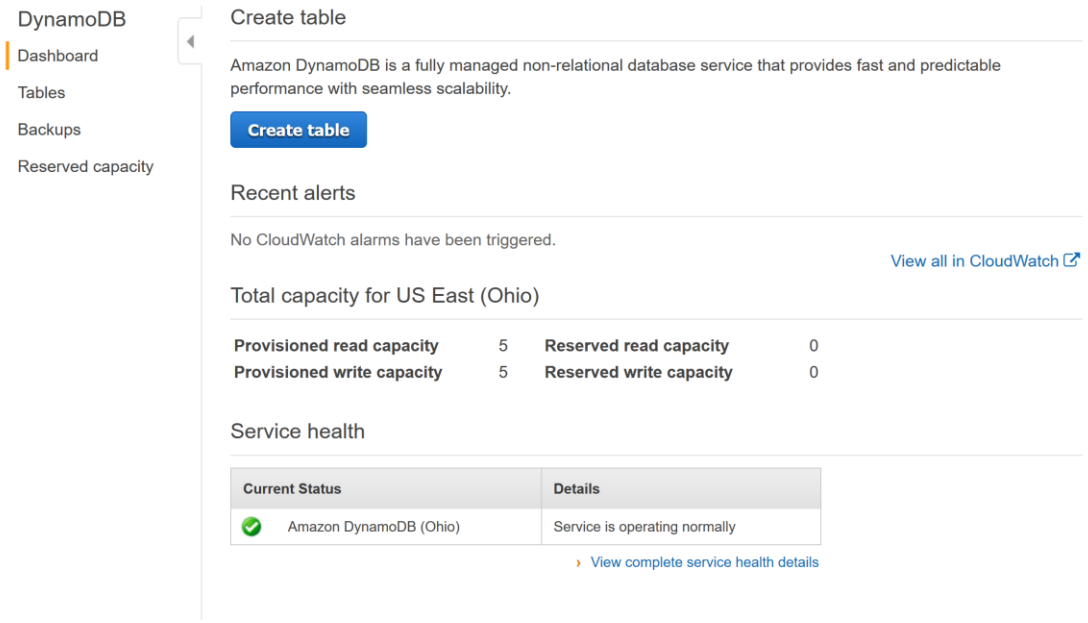
- Nothing identifies a key within the Item. As we can see on the right, the keys have no distinguishing attributes (‘title’ and ‘year’)

- The remaining attributes in the Item are stored within the field ‘info Map’, but that is just the structure of this Item
  - All attributes must have a type (String, Number, Binary)

```
▼ Item {3}
  ▼ info Map {9}
    ▶ actors List [3]
    ▶ directors List [2]
    ▶ genres List [3]
      image_url String : http://ia.media-imdb.com/images/M/MV5BMTkxOTIxMDU2OV5BMl5BanBnXkFtZTcwNjM5NjQyMg@@._V1_SX400_.jpg
      plot String : A film crew goes to a tropical island for an exotic location shoot and discovers a colossal giant gorilla who takes a shine to their female blonde star.
      rank Number : 3551
      rating Number : 8
      release_date String : 1933-03-07T00:00:00Z
      running_time_secs Number : 6000
    title String : King Kong
    year Number : 1933
```

Dynamo Dashboard

- The Dynamo dashboard in AWS gives a quick overview of the current status of your Dynamo setup
- From here it is quite easy to create a table
- We can also view the **capacity** of running Dynamo tables
  - Read and Right capacities are in X per second



Creating a Table

Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*

Primary key* Partition key

String

☐ Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- On-Demand Backup and Restore Enabled NEW!

You do not have the required role to enable Auto Scaling by default.
Please refer to documentation.

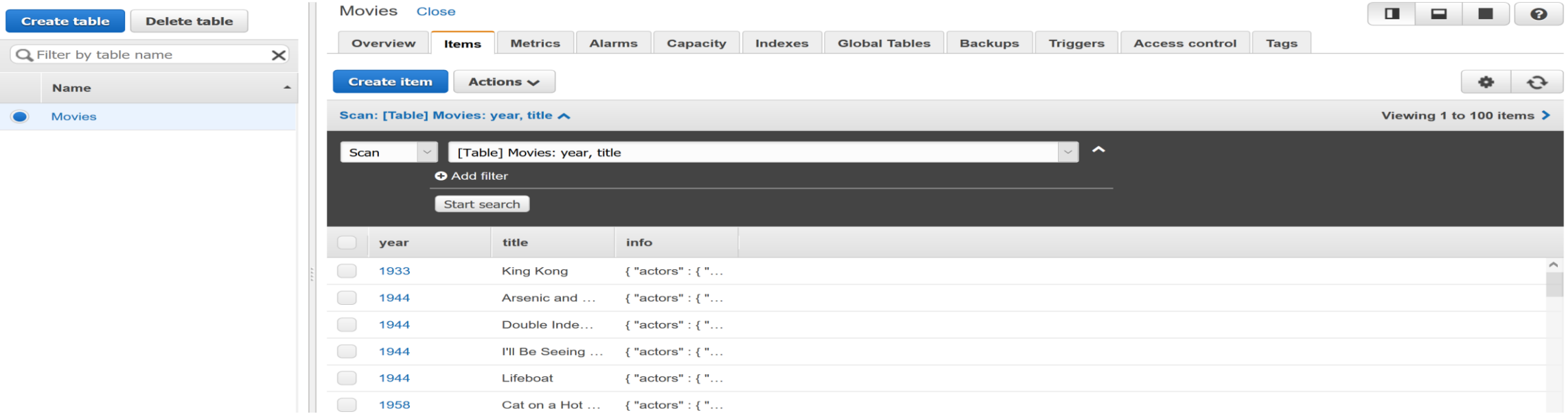
Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel Create

Viewing Tables

- From the **Tables** page on the Dynamo dashboard we can view our current tables
  - This gives a view of the tables **Partition** and **Sort Keys**, as well as **Indexes** and **Capacity**
- This also enables us to click through to a **Table** viewer

Viewing Tables



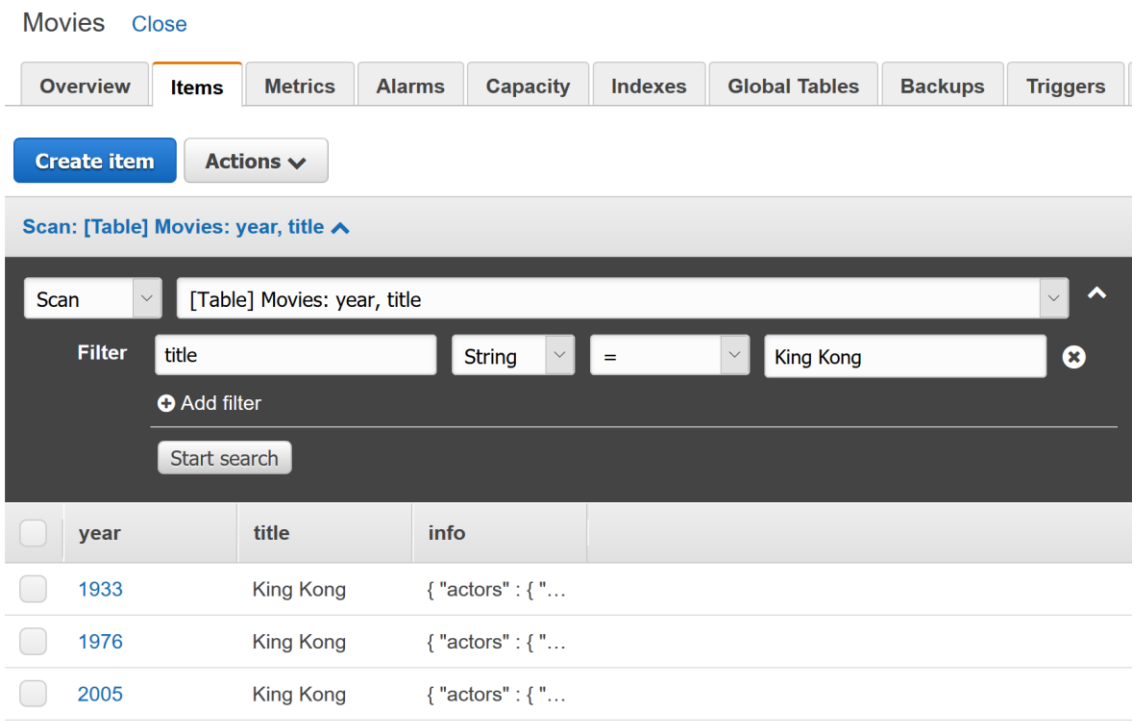
- From the Table view we have the **Items** tab, which gives us access to the following features:
  - Create Item
  - Scan/Query the data
  - Edit/Duplicate data
  - View an item

Scanning/Querying

- From the **Items** tab we can also query/scan for items

- To accomplish this we simply need to state what filters we want applied
  - Filters are case-sensitive
  - Filters can only be applied to keys/indexes

- A Scan will take a while to find all items
- A Query will be more/less instantaneous, but requires **sort** and **primary** key to be defined



Discuss Pros/Cons of Dynamo

Kafka (Eventing Database)

Eventing Systems

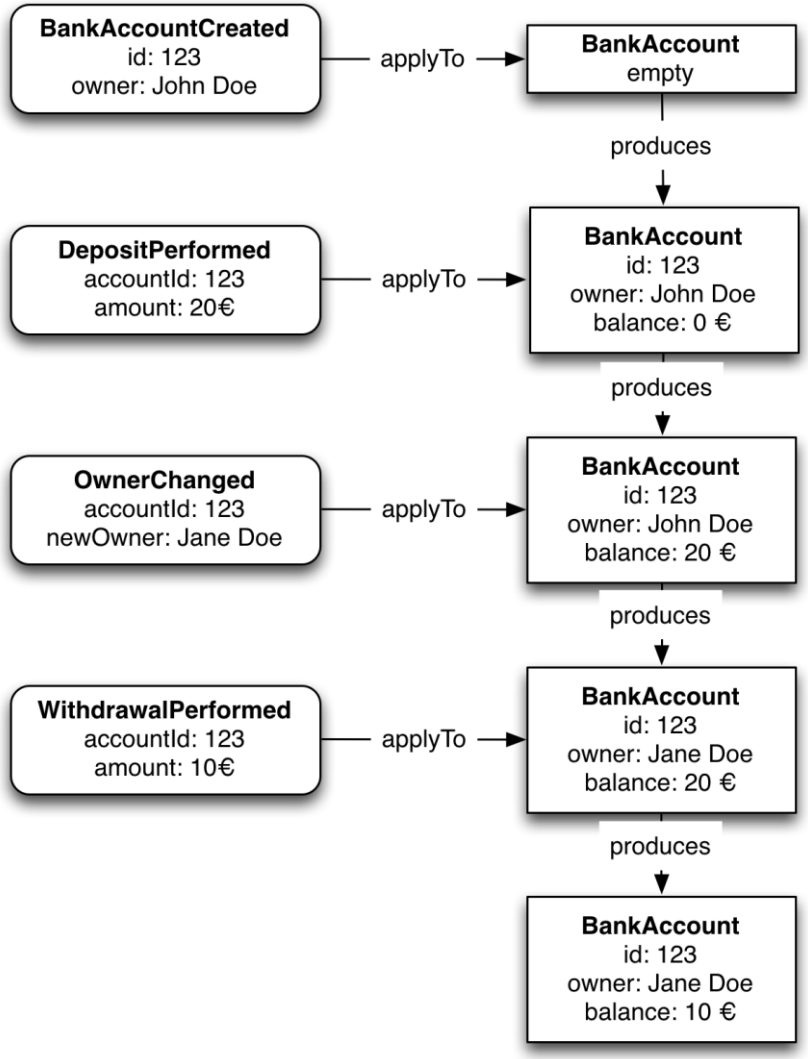
- The database systems we've explored so far have been concerned with representing data only at the current moment in time.
  - Any updates or modifications to a record update it and no information about the previous state is maintained.

- Eventing systems however are focused on logging information as events or changes occur. The data in eventing systems are immutable.
  - This means that they store interactions over time, rather than just the current state for a record.

Event Sourcing

- One of the advantages of eventing databases like Kafka is their ability to support event sourcing
  - Event sourcing is an approach for data management that logs all the state changes of an entity within the system.

- Within an event sourcing we can merge events into a new state through an **Aggregate** (a representation of entities made by playing back events).
  - A **projection** is an aggregate designed to read the current state



Discuss Pros/Cons of Kafka

SQLite (Embedded Databases)

SQLite

- SQLite is considered an embedded database system as it does not require a server to operate, instead SQLite is managed through code and files.
  - A SQLite database is simply a local file with a **.sqlite** extension
  - Rather than executing queries against a server, queries are managed by the software and update the **.sqlite** file accordingly

- This setup is commonly used in embedded devices as it doesn't require running additional software.
  - It's actually a fairly common database design for mobile applications.

Discuss Pros/Cons of SQLite

End Slide

DBMS for Data Analytics