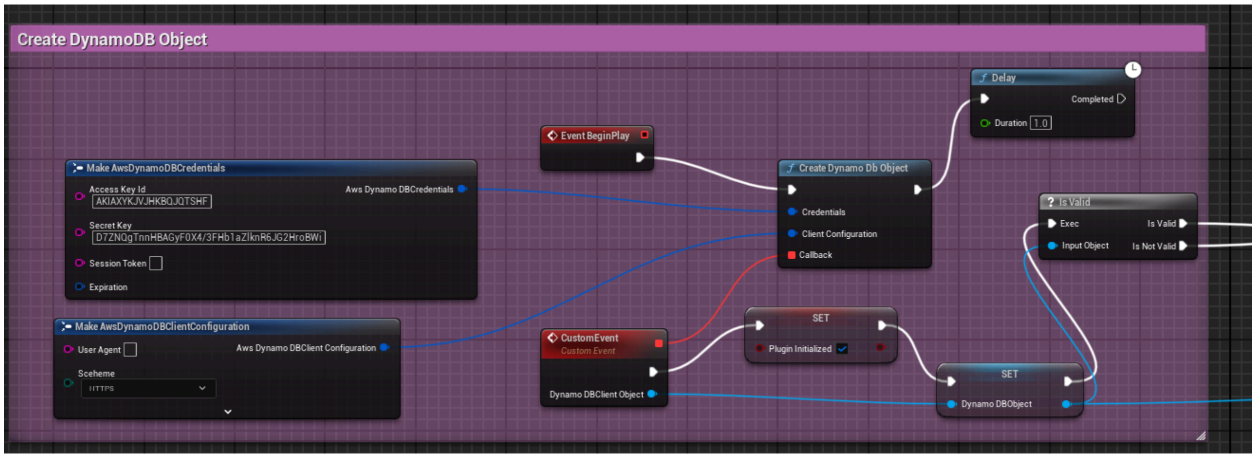
1. Download AWS DynamoDB Database Plugin by eelDev from the unreal engine marketplace and install it to your unreal engine version in the library.
2. Open your project in the unreal engine editor and go to Edit -> Plugins -> Search for “AWS” and look for “AwsCore::DynamoDB” and click the check to activate this plugin and then restart your editor.
3. Create a blueprint actor in your unreal engine because you will be attaching your DynamoDB blueprints (like queries and such) to an actor and placing it in your level.

Creating DynamoDB object:



Access Key Id: AKIAXYKJVJHKBQJQTSHF
Secret Key: D7ZNQgTnnHBAGyF0X4/3FHb1aZlknR6JG2HroBWi
Sceheme: HTTPS
Duration: 1.0
Plugin Initialized: ✓

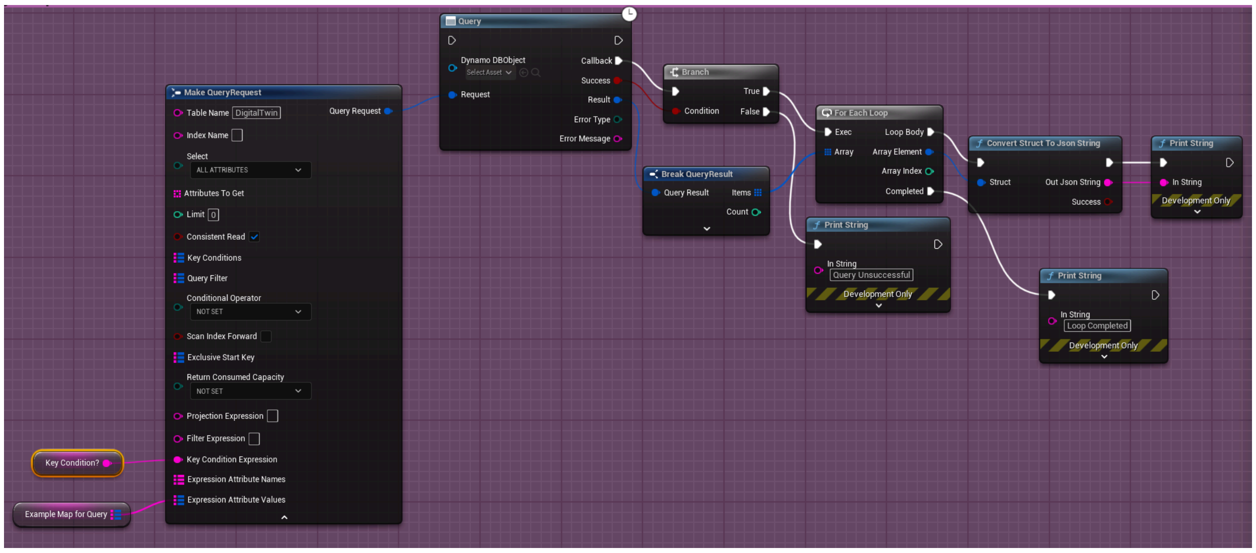
- Once you have the plugin ready to go, follow these instructions on how to create the DynamoDB object so that you can successfully perform a Query or Scan to the database table.

- You have to get the access key ID and secret key for your access to the aws dynamodb table.

- Then, you want to include the “Make AwsDynamoDBClientConfiguration” node where you do not have to change much, except for using the drop-down menu to change your region to match your region in the AWS console.

- -Nodes: Make AwsDynamoDBCredentials, Make AwsDynamoDBClientConfiguration, Event BeginPlay, Custom event with DynamoDB Client Object object reference, Set for the value of a DynamoDBObject variable you need to create, and a check if it’s valid.

Making A Query to DynamoDB:



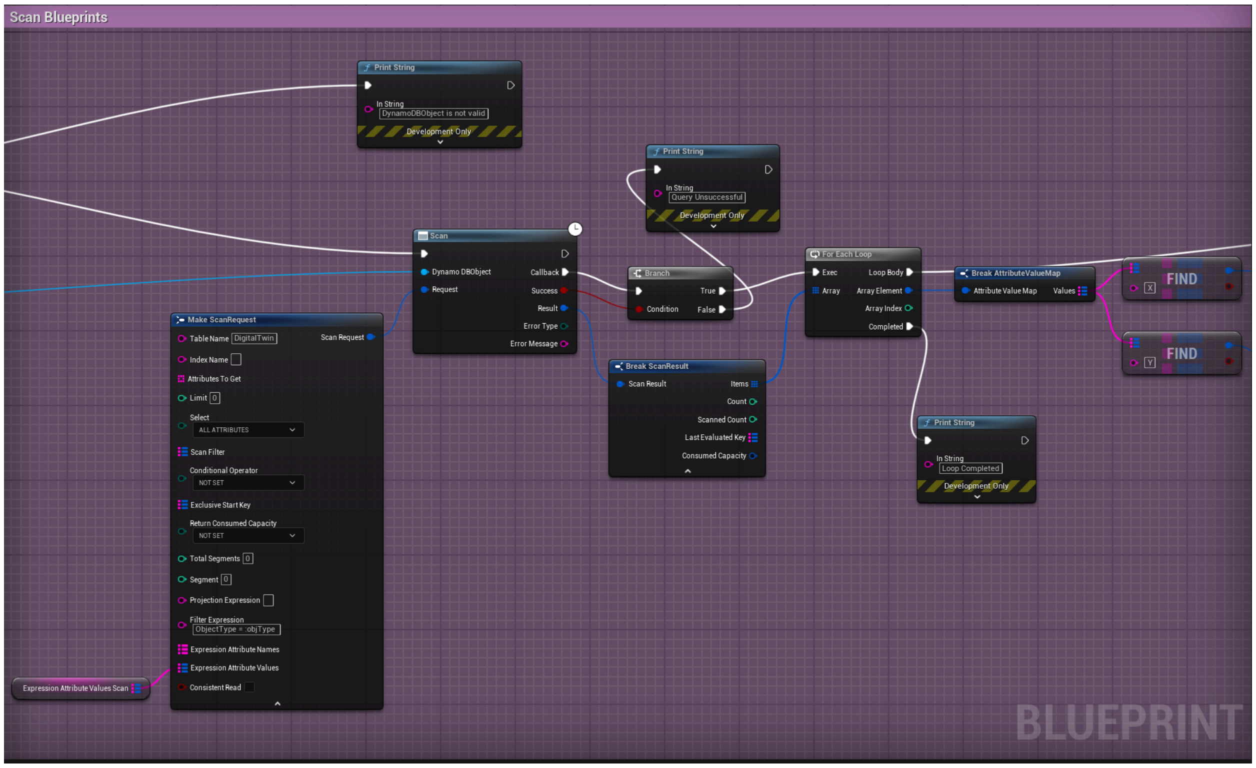
- Firstly, note that this query is simply printing table items to the output log, but you should adjust the blueprints (starting at the for loop body) to perform whatever you are trying to do.

- Secondly, here, I am using the table's partition key as the "key condition expression" in the query request node. For our table, it's ObjectID so I made the key condition expression "ObjectID = :objID."

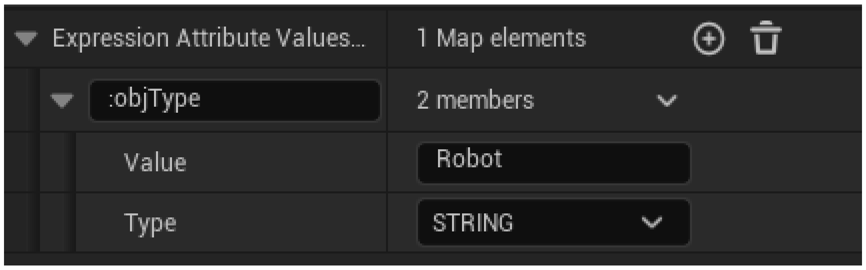
- Then, you would simply use this setup and then set the DynamoDB object in the "Query" node to the dynamoDB object you created earlier.

- Nodes: Make QueryRequest, Map of strings to attribute value structures variable for the expression attribute values map, string variable for key condition, Query, Branch, Break Query result, For each loop, and then the rest of the nodes depend on the functionality you intend to implement.

Making A Scan to DynamoDB

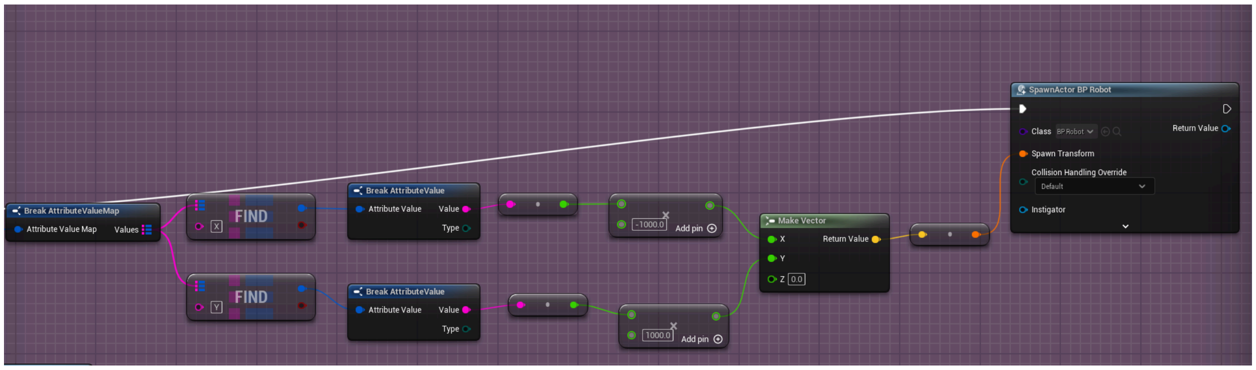


- Here, I will be implementing some functionality for the information we are getting from our database table, but first, starting with the ScanRequest, I am using an Expression Attribute Value map for the Expression Attribute Values pin in the “ScanRequest” node. In this case, I am filtering for table items with ObjectType = “Robot” which is of type String. Inside that map variable, it looks like this:



Expression Attribute Values...	1 Map elements
:objType	2 members
Value	Robot
Type	STRING

- Then from there, it’s the same thing where you can set the DynamoDB object you created to the DynamoDBObject pin in the “Scan” node and then implement your functionality from there.

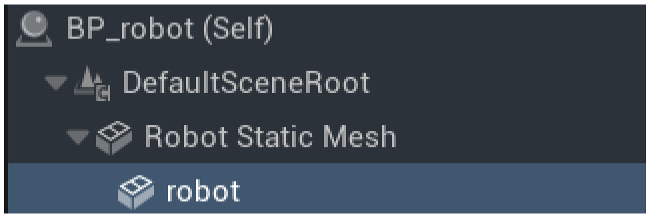


- Here is the rest of our Scan blueprints that get the objects of type robot and spawn them into the level at X and Y coordinates provided to us by the Database table. I get these X and Y coordinates by using these "FIND" nodes after I break the attribute value map from the array element pin the for each loop node. Then I get the value which converts the X and Y numbers from string to floats for the vectors that the robots will spawn at.

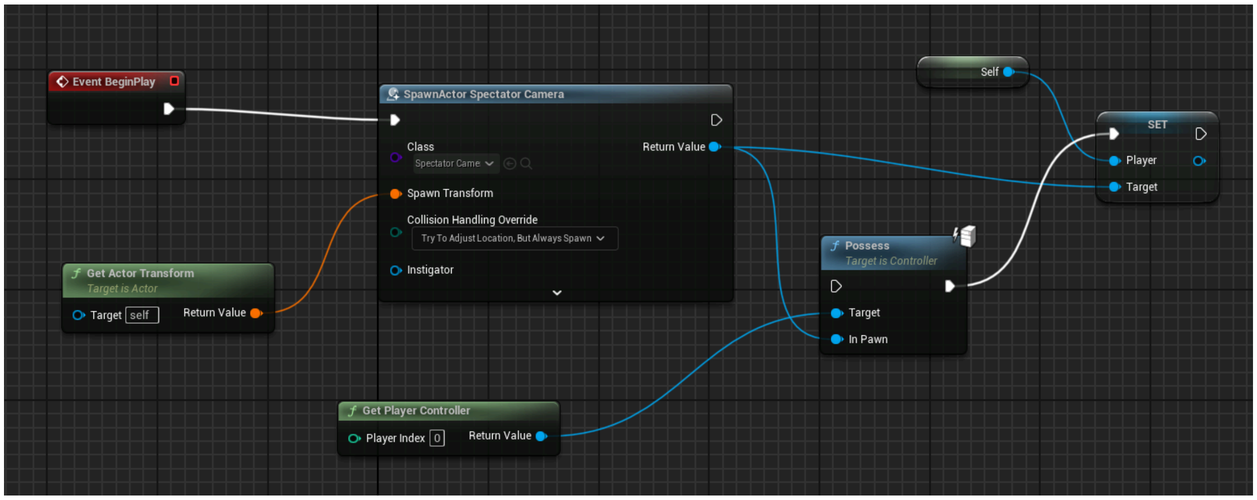
- After that, when you hit the play button in the level, the robots will spawn in at the X and Y coordinates from the database.

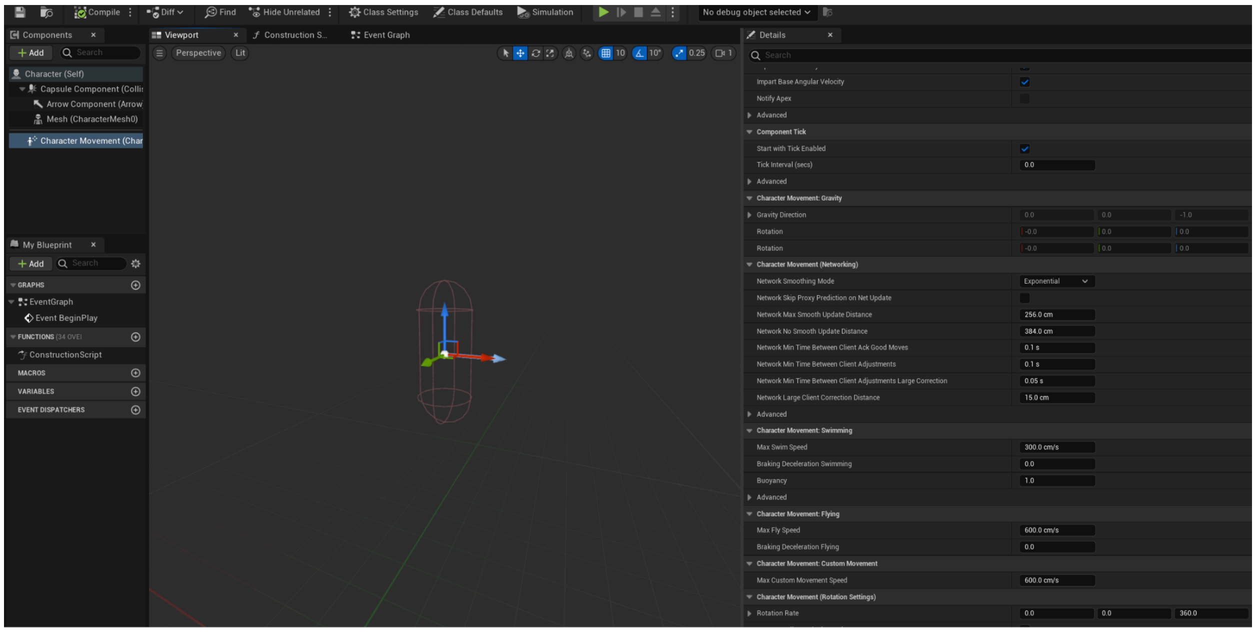
- Nodes: Make ScanRequest, Map of strings to attribute value structures variable for the expression attribute values map, Scan, Branch, Break ScanResult, For each loop, Break attribute Value map, and then the rest of nodes depend on the functionality you are attempting to implement, but for these blueprints we do include the nodes: Find (for X and Y in the table), then Break AttributeValue, convert from string to float, multiply to scale up to the map level size, make those X and Y values the vectors (for the robot 3D objects), convert the vector to transform and we used SpawnActor BP_Robot(custom blueprint class we made for the robot 3D objects) node to spawn in the robots at those specific transforms.

- Bp_robot (actor blueprint class): Just do this in the components section of the blueprint actor (robot here is the 3D object:



Simple Character & Camera Blueprints:





Impart Base Angular Velocity	✓		
Notify Apex	☐		
Advanced			
Component Tick			
Start with Tick Enabled	✓		
Tick Interval (secs)	0.0		
Advanced			
Character Movement: Gravity			
Gravity Direction	0.0	0.0	-1.0
Rotation	-0.0	0.0	0.0
Rotation	-0.0	0.0	0.0
Character Movement (Networking)			
Network Smoothing Mode	Exponential		
Network Skip Proxy Prediction on Net Update	☐		
Network Max Smooth Update Distance	256.0 cm		
Network No Smooth Update Distance	384.0 cm		
Network Min Time Between Client Ack Good Moves	0.1 s		
Network Min Time Between Client Adjustments	0.1 s		
Network Min Time Between Client Adjustments Large Correction	0.05 s		
Network Large Client Correction Distance	15.0 cm		
Advanced			
Character Movement: Swimming			
Max Swim Speed	300.0 cm/s		
Braking Deceleration Swimming	0.0		
Buoyancy	1.0		
Advanced			
Character Movement: Flying			
Max Fly Speed	600.0 cm/s		
Braking Deceleration Flying	0.0		
Character Movement: Custom Movement			
Max Custom Movement Speed	600.0 cm/s		
Character Movement (Rotation Settings)			
Rotation Rate	0.0	0.0	360.0