Azura Cosmos DB

Chad Armstrong

Setup

To run this project you must sign up for an Azure account and sign in. Once there, on the left panel click the “+”(“Create a resource”) sign at the top of the panel. Then select or search for Azure Cosmos DB. Fill in the following information(if you don’t have a resource group make a new one) and go ahead and create the database.   
 Once you create it go to your home screen(Left panel->Home) and click on your recently created database. Go to the Keys page(Settings->Keys), you’ll need the URI and PRIMARY KEY to hook up Unity to this database.

Unity  
 To add your database URI and PRIMARY KEY, find the BasicInfo.cs and adjust the values that are in that class. From there start up the "Main.unity" scene and it should show three buttons on the left and just a blue-ish panel on the right.  
 When you run the scene for the first time, hit the "Populate Database" button on the left and procede to wait until this massive 50,000 entity database gets initalized. This may take up to 20-40 minutes(I averaged around 1,000 per 40 seconds). Once the database is populated, be careful to not click the "Reset database button or you would have to sit through the process all over again.  
 Now that you have a populated database go ahead and hit "Start Game". This is just a silly little game to show contribution to the database through some numbers changes. To add users to track go to the "GameController" Game Object in the Unity Scene Hierarchy and in the inspector add a list of names under "Other Users To Track". This will show the a tracker for these user next time you click "Start Game".

Notes

* I recommend using the Azura Cosmos Emulator, prevents develops from doing harmful operations on the main server and also they can control everything about it.  
  <https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator>
* Unity has threads happening in a thread pool behind the scenes, this makes it tricky using multithreaded operations/api calls while using Unity.
* If you want 50,000 element database involved when the first user is created, having an asynchronous call will not cut it. You’d have to create the database on the main thread, which for 50,000 elements takes 10+ seconds. After this the user can safely interact with the database and read/write as much as they need.
* Drops to 1 FPS if several users write/update at one time (5+ users). So scalabilty will be tough, unless this was more of a stress test and there won’t be an upwards of 125,000 updates to the database a second.
* One recommendation I have would be to look into Unity’s job system, it’s how the “handle” threading.