**1.Simplicity & Ease of Implementation – The solution should be straightforward to deploy and maintain?**

**Solution Components**

**Azure Cosmos DB:** Will store recent billing records in less than 3 months that require low-latency access.

**Azure Data Lake Storage:** Will be used as archival storage for billing records older than 3 months.

**Azure Functions:** Azure Functions used for archive and retrieval of records.

**Azure Data Factory:** To copy bulk data initially to data lake we can do with the ADF pipeline.

**Archive Function:**

Create function app from app service and write function app code to execute below steps that archive cosmos db data older than 3 months based creation or modified timestamp of the records.

We can use cosmos db change feed trigger to trigger function app whenever there is change in records.

Below are the steps to execute in function:

* + 1. The function checks the record's timestamp.
    2. If the record is older than 3 months, it will read from Cosmos db.
    3. The record is then written as a new file to the appropriate folder in Azure Data Lake (e.g., /billing/today\_timestamp/billing\_id.json).
    4. After successful archival to Data Lake, the function app updates the Cosmos db record to mark it for deletion using Time to Live.

**Retrieve Function:**

This function is used to retrieve the requested records if that are older than 3 months from the Data Lake if it is not found in the Cosmos db.

We can trigger this function with HTTP post trigger from the application

1. When a request comes to retrieve a billing record by its ID, this function first attempts to read it from cosmos db.
2. If the record is found in cosmos db it's return it.
3. If the record is not found in cosmos db, the function then queries the data lake path based on the record's ID.
4. The function retrieves the record file from data lake and returns its content.

**2. No Data Loss & No Downtime – The transition should be seamless, without losing any records or requiring2 service downtime?**

To ensure the no downtime and data loss during the transition,

Designed the above archive function to delete the cosmos db records only after successful archival and in the retrieve, function try to request records from the both cosmos db and data lake to ensure no downtime.

**Reference:**

[**https://sqlroadie.com/2019/07/21/azure-cosmos-db-real-time-data-movement-using-change-feed-and-azure-functions/**](https://sqlroadie.com/2019/07/21/azure-cosmos-db-real-time-data-movement-using-change-feed-and-azure-functions/)

[**https://learn.microsoft.com/en-us/answers/questions/1281134/customer-plan-to-use-cosmos-db-and-retain-operatio**](https://learn.microsoft.com/en-us/answers/questions/1281134/customer-plan-to-use-cosmos-db-and-retain-operatio)

**3.No Changes to API Contracts – The existing read/write APIs for billing records must remain unchanged?**

Since we only change the backend data retrieval logic no need to change the client facing api when reading the data.

**4. Include an architecture diagram illustrating your proposed solution.**

**A diagram of a data flow

AI-generated content may be incorrect.**

**5. Provide pseudocode, commands, or scripts for implementing core logic (such as data archival, retrieval, and cost optimization strategies).**

**a. Pseudocode for data archive and retrieve is in folder code/**

Used the ChatGPT to get the azure function codes with below prompts.

**References:**

**Prompt:**

**Azure function code to archive the data records in cosmos db which are older than 3 months to azure datalake.**

**Give azure function code to retrieve the first by checking in the cosmos and if not available get from data lake using billing ID/**

**b. Cost optimization**:

* We can use provisioned and auto scale throughput.
* We can use Data Lake gen2 tier which is cost effective

Reference: <https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/how-to-provision-autoscale-throughput?tabs=api-async>

<https://learn.microsoft.com/en-us/azure/data-factory/connector-azure-data-lake-storage?tabs=data-factory>

**Challenges in production environments:**

* Race condition in the archival and retrieval function may cause delete of data in cosmos db before updating the in data lake.
* Data duplication while archiving data lake.

**Other Possible solutions:**

**Solution 2:**

**A diagram with text on it

AI-generated content may be incorrect.**

This also one of the cost optimized solutions for archive and retrieve the data, but it lack some features like folder structures and integrate with other services.

**Solution 3:**

**A diagram of a graph

AI-generated content may be incorrect.**

This solution will help to copy bulk data to data lake or blob storage using ADF pipeline, and it might not be suitable for regular archival of data.

**Conclusion:** Considering the cost and requirements the above solution 1 would be feasible.