**Problem Statement:**

* You have a read-heavy Cosmos DB storing large billing records (300 KB each).
* Older than 3 months = rarely accessed, but still needed (within seconds).
* The data volume (~2M records) is leading to rising Cosmos DB costs.

**Solution Proposed:**

* Archive billing records older than 3 months from Azure Cosmos DB to Blob Storage using a PowerShell script (invoked via Azure Function Timer or manually).
* Modify the data retrieval logic with a fallback: if the record isn’t found in Cosmos DB, read it from Blob Storage.
* Keep writes unchanged.
* Maintain API contract and response time requirements.

**Key Benefits**:

* Substantial cost savings (Blob Storage is 10–20x cheaper than Cosmos DB for cold data).
* Maintains **zero downtime**, **no data loss**, and **unchanged API contract**.

**Principle behind choosing this Approach:**

The **CAP Theorem** says that in a distributed system, you can **only guarantee two of the following three**:

| **Consistency** | **Availability** | **Partition Tolerance** |
| --- | --- | --- |
| Always returns the most recent write | Always responds to requests (even during failure) | System continues working even when network partitions occur |

Based on the given problem statement the key priorities should be:

1. **Availability** (must respond to reads — hot or cold — within seconds)
2. **Partition tolerance** (needed in any distributed cloud system)

**PROMPTS USED:**

Prompt-1:

We have a serverless architecture in Azure, where one of our services stores billing records in Azure Cosmos DB. The system is read-heavy, but records older than three months are rarely accessed. Over the past few years, the database size has significantly grown, leading to increased costs. We need an efficient way to reduce costs while maintaining data availability. Current System Constraints Record Size: Each billing record can be as large as 300 KB. Total Records: The database currently holds over 2 million records. Access Latency: When an old record is requested, it should still be served, with a response time in the order of seconds. Provide cost effective solution.

Prompt-2:

Can you use the CAP Theorem in the solution.

Prompt-3:

I am concerned about "Access Latency: When an old record is requested, it should still be served, with a response time in the order of seconds." can you recommend as store that satisfies this and at the same time most cost efficient as glacier of aws is cost efficient but takes time to retrieve data.so using azure ecosystem build a solution.