Create a prompt for Free AI Diagram Generator for the following design:

Actuals:

Azure Cosmos DB, "BlackBox" Service having Read/Write APIs to fetch/store data from Azure Cosmos DB.

Specs for Azure Cosmos DB:

• Lots of reads.

• Holds more than 2 million billing records, each record can be around 300 KB.

• Records span many years but records older than 3 months are rarely accessed.

Goal & Caveats:

Optimize storage and operational cost keeping these in check:

• Records older than 3 months should be fetched if needed within seconds.

• Simple solution, easy to manage.

• No Data Loss, No Downtime.

• No Changes to Read/Write APIs.

Approach:

Move records older than 3 months to Azure Blob Storage [Cold Tier] using Azure Functions with Timer Trigger. Introduce a secondary layer behind the Read API which is aware of the dual storage namely Azure Cosmos DB and Azure Blob Storage.

Implementation:

code/archivefunctionapp/terraform/main.tf: This configuration creates a resource group, a storage account, a blob container named, and an app service plan with an app function for archive.

code/archivefunctionapp/\_\_init\_\_.py; code/archivefunctionapp/function.json: This contains the python code and function.json file for the archive function.

code/billingsmartapi/terraform/main.tf: This configuration creates an app service plan an app function for read.

code/billingsmartapi/\_\_init\_\_.py: This contains the python code for read function.

**Prompt for Free AI Diagram Generator:**

Design a cloud data archival solution with the following components and requirements:

**Actual System**

* **Azure Cosmos DB:** Stores all billing records (over 2 million), each around 300 KB.
  + Heavy read operations.
  + Data spans many years. Records older than 3 months are rarely accessed.
* **"BlackBox" Service:** Provides Read/Write APIs that interact with Cosmos DB. These APIs must **not change**.

**Goal & Constraints**

* **Primary Goal:** Optimize storage and operational costs.
* Records older than 3 months:
  + Should be fetched within seconds if needed.
  + Stored long-term using a low-cost solution.
* **Solution Must Be:**
  + Simple, easy to manage.
  + No data loss, no downtime.
  + Must NOT require changes to existing Read/Write APIs.

**Approach**

* **Azure Functions:** Timer-triggered function regularly moves records older than 3 months from Cosmos DB to Azure Blob Storage (Cold Tier).
* **Dual-Layer Read Logic:**
  + A secondary layer behind the Read API is aware of both Cosmos DB and Blob Storage locations.
  + When a read request is made, data is fetched from Cosmos DB if <=3 months old, or Blob Storage if older.
* **No API changes required for clients.**

**Implementation Details**

Show the following components in the diagram:

1. **Azure Cosmos DB:** Primary hot storage for new & recent records.
2. **Azure Blob Storage (Cold Tier):** Long-term archive for records older than 3 months.
3. **Azure Functions:**
   * **Archive Function:** Moves data older than 3 months to Blob Storage using a timer trigger.
   * **Read Function Layer:** Checks both Cosmos DB and Blob for data; returns to API as needed.
4. **"BlackBox" Service:** Consumes the Read/Write APIs (APIs interface is unchanged).
5. **Terraform Scripts and Python Files:**
   * Indicate presence of IaC (Terraform) for deploying Azure resources.
   * Indicate use of Python for app logic.

**Label all connections and indicate data flow:**

* Write/Read flow from APIs to Cosmos DB and (for old data) to Blob Storage via the Read Function layer.
* Scheduled flow from Cosmos DB to Blob Storage via the Archive Function.

**Style:**  
Use clear boxes for each service. Use arrows to indicate all flows. Explicitly mark which function moves data to cold storage, and which handles dual read logic.

**Title for Diagram:**  
“Cost-Optimized, Dual-Layer Archival Solution on Azure Without API Change”