# Cost Optimization Challenge: Managing Billing Records in Azure Serverless Architecture

## 📌 Problem Statement

A serverless architecture in Azure stores billing records in Azure Cosmos DB. The database is read-heavy, with over 2 million records, and older records (3+ months) are rarely accessed. The goal is to reduce storage and query costs without compromising availability or requiring changes to existing APIs.

---

## Solution: Tiered Storage Architecture

This solution introduces a \*\*tiered storage model\*\* using Azure services:

### Hot Tier: Azure Cosmos DB

- Stores the most recent 3 months of billing records.

- Ensures fast, low-latency access for active data.

### Cold Tier: Azure Blob Storage (Cool Tier)

- Archives records older than 3 months.

- Significantly cheaper storage with acceptable latency (seconds).

- Data stored as `{recordId}.json` blobs.

### Azure Functions: Serverless Automation

1. \*\*Read-Through API Function\*\*:

- Reads from Cosmos DB first. If not found, falls back to Blob Storage.

- API remains unchanged for the client (transparent logic).

2. \*\*Archival Function\*\*:

- Scheduled job (e.g., daily/weekly).

- Migrates records older than 3 months from Cosmos DB to Blob Storage.

- Deletes record from Cosmos DB only after confirming successful archival.

---

## Deployment Plan

1. \*\*Set up Azure Blob Storage\*\*

- Create storage account & container.

- Set default access tier to \*\*Cool\*\*.

2. \*\*Update Read API\*\*

- Add fallback logic to check Blob Storage if record isn't in Cosmos DB.

3. \*\*Develop Archival Function\*\*

- Time-triggered Azure Function.

- Ensures safe migration using write-then-delete pattern.

4. \*\*Deploy in Phases\*\*

- Deploy updated API first (no changes in behaviour yet).

- Perform initial backfill (move old records).

- Enable scheduled archiving for continuous cost optimization.

---

## Benefits

- \*\*Up to 90% cost reduction\*\* by moving inactive data to Blob Storage.

- \*\*Zero downtime\*\*, no data loss (due to write-before-delete).

- \*\*No changes to API contracts\*\*.

- \*\*Fully serverless & scalable architecture\*\* using native Azure services.

---

## Azure Services Used

- Azure Cosmos DB

- Azure Blob Storage (Cool tier)

- Azure Functions

- Azure Data Factory (optional for complex pipelines)

- Application Insights (for monitoring)

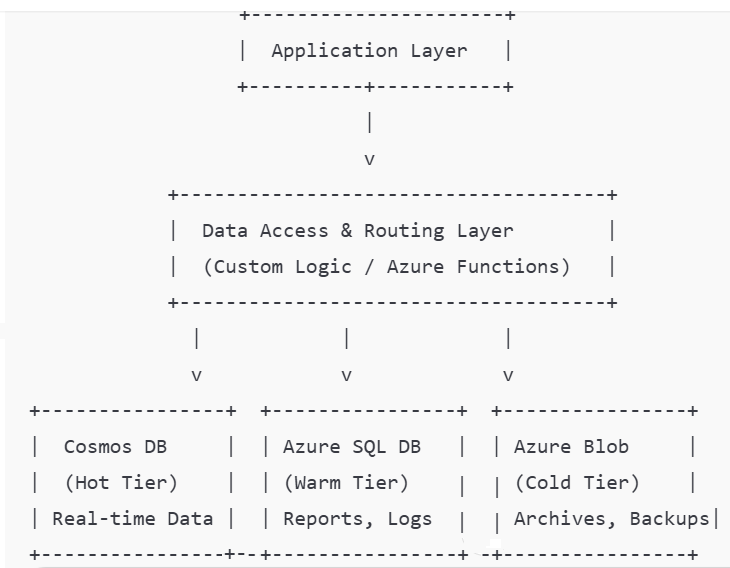
---

## Testing & Monitoring

- Validate read API with both recent and archived data.

- Monitor archival job execution with logging & alerts.

**Tiered Storage Architecture in Azure (for Cost Optimization):  
--------------------------------------------------------------------------------**

****

**Blob Lifecycle Management (Hot → Cool → Archive)**

Azure Blob Storage supports automatic tiering using Lifecycle Management rules. You can define these rules to transition data based on last modified date.

-> Steps in Azure Portal:

1. Go to your Storage Account.
2. Navigate to:  
   Data Management ➜ Lifecycle Management ➜ + Add a rule
3. Configure the rule:
   * Rule Scope: Entire storage account or specific container.
   * Filter: Prefix (e.g., logs/) or blob type.
   * Actions:
     + Move to cool after X days.
     + Move to archive after Y days.
     + Delete after Z days (optional).

**Example JSON Rule:  
  
{**

**"rules": [**

**{**

**"enabled": true,**

**"name": "tier-transition-rule",**

**"type": "Lifecycle",**

**"definition": {**

**"filters": {**

**"blobTypes": ["blockBlob"],**

**"prefixMatch": ["logs/"]**

**},**

**"actions": {**

**"baseBlob": {**

**"tierToCool": {**

**"daysAfterModificationGreaterThan": 30**

**},**

**"tierToArchive": {**

**"daysAfterModificationGreaterThan": 90**

**}**

**}**

**}**

**}**

**}**

**]**

**}**

**This rule transitions blobs in logs/ from Hot → Cool after 30 days, then Cool → Archive after 90 days.  
Move data between Cosmos DB, SQL DB, and Blob Storage based on age or custom metadata.**