**Cost Optimization Challenge**

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**

1. Record Size: Each billing record can be as large as 300 KB.
2. Total Records: The database currently holds over 2 million records.
3. Access Latency: When an old record is requested, it should still be served, with a response time in the order of seconds.

**Requirements**

1. Simplicity & Ease of Implementation
2. No Data Loss & No Downtime
3. No Changes to API Contracts

**Solution:**

**Hot-Cold Data Tiering with Azure Functions**

A great approach would be to **implement a Hot-Cold Data tier storage strategy** using **Azure Cosmos DB for hot data** and **Azure Blob Storage for cold data** with a **transparent data access layer**. The basic idea is to move older records (which are less frequently accessed) from Cosmos DB to Blob Storage while maintaining a seamless read experience. Also we use **Azure Functions** or **Logic Apps** to automate the migration process without causing downtime. These can be scheduled to run during low traffic periods and can operate incrementally to move data in batches.

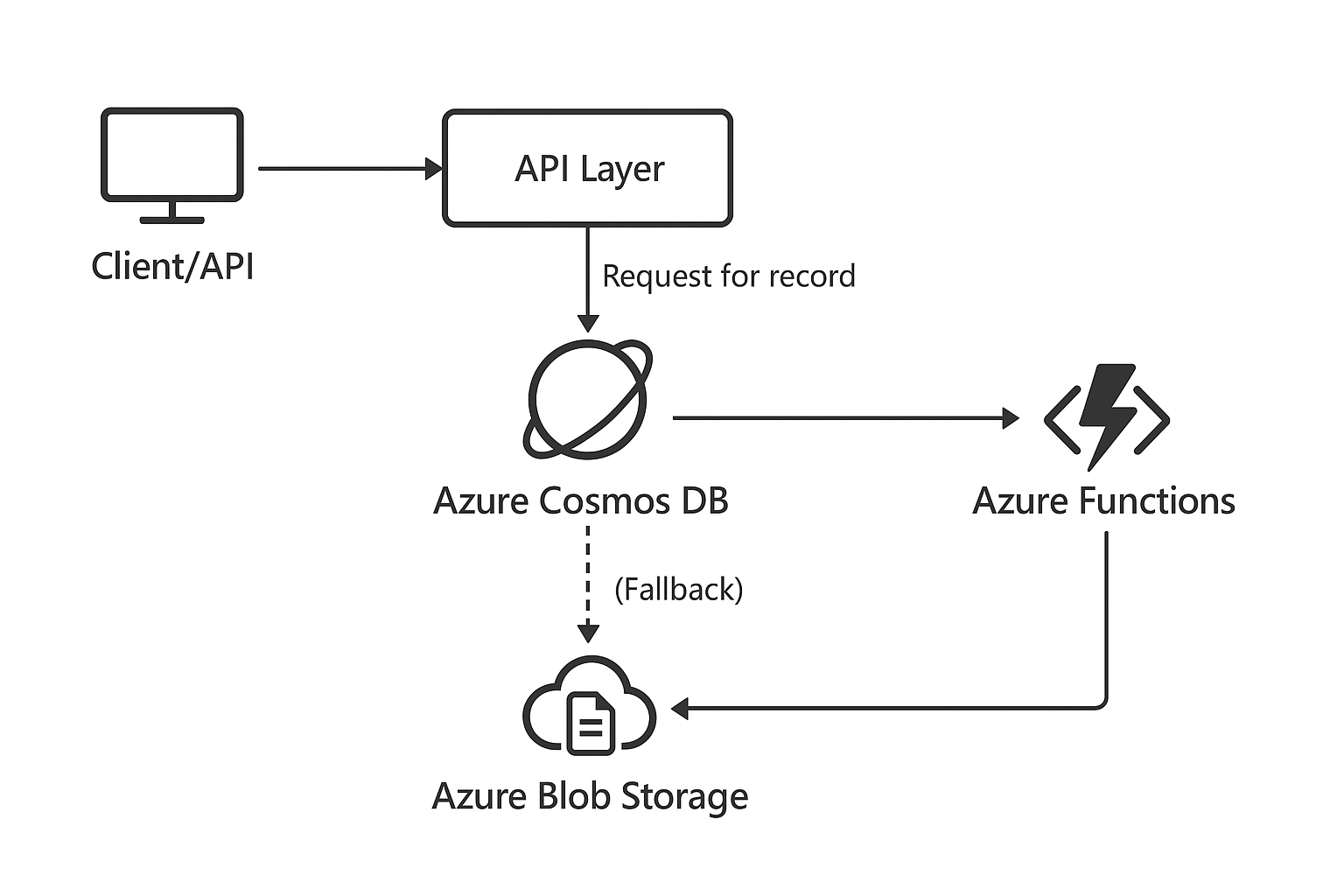
**Classify the Data**

We categorize data based on usage frequency:

* **Hot Data**: Recent records (≤ 3 months), frequently accessed — **stay in Cosmos DB**.
* **Cold Data**: Older records (> 3 months) rarely accessed — **moved to Azure Blob Storage** to reduce cost.
* Use **Azure Functions** to serve **cold data** on demand, triggered transparently behind the scenes.
* **Blob Storage** is much cheaper for long-term storage compared to **Cosmos DB.**

**Architecture Components**

| **Component** | **Description** |
| --- | --- |
| **Azure Cosmos DB** | Stores hot billing data (last 3 months) |
| **Azure Blob Storage (Cold Tier)** | Stores older billing records in compressed or serialized format (JSON, Parquet, etc.) |
| **Azure Functions** | 1. Migrate old records to Blob Storage 2. Fallback reader to fetch from Blob if Cosmos misses |



**Azure Function:**

**Hot-to-Cold Migration Function** — Moves old records from Cosmos DB to Blob Storage.

**Read Fallback Function** — Serves records from Blob Storage when Cosmos doesn't have them.