1. Components

Azure Cosmos DB

Holds active (< 90 days) records exactly as today.

Azure Blob Storage

Archive container for old records as compressed JSON.

Azure Table Storage

Simple map: `PartitionKey="billing"`, `RowKey=recordId`, `BlobUri`.

Azure Function (Timer Trigger)

Moves old records to Blob + updates Table + deletes from Cosmos.

( reuse the same Function for read fallback if you front it with HTTP.)

2. End-to-End Flow

1Writes

- Unchanged: all new billing records go to Cosmos.

2. Daily Archival

Timer-triggered Function runs (e.g. 2 AM UTC).

Queries Cosmos for `c.\_ts < now()–90 days`.

For each record:

1. Serialize + gzip → upload to Blob as `"{id}.json.gz"`.

2. Upsert `{ PartitionKey:"billing", RowKey:id, Uri:blobUri }` in Table.

3. \*\*Then\*\* delete the record from Cosmos.

3.Reads

In your existing DAL/Function:

FUNCTION

- API endpoints and payloads stay exactly as today.

3.code: Archival Function (Python-style)

---

### 4. Deployment Steps (≅ 15 min)

1. \*\*Create\*\* a new Storage Account

- Blob Container: `billing-archive`

- Table: `BillingIndex`

2. \*\*Deploy\*\* Azure Function

- Bind Cosmos DB input (Query) & output (none)

- Bind Table Storage & Blob Storage via SDK

- Schedule on a daily or hourly Timer trigger

3. \*\*Update\*\* your service’s read logic

- Wrap Cosmos read in try/except and fallback to Blob as shown

4. \*\*Test\*\*

- Backfill a small batch (1 K records)

- Verify reads from Blob for > 90 days old IDs

- Confirm Cosmos RU/s drop

5. Ongoing Ops

Monitoring: Application Insights on Function failures

Blob Lifecycle: Enable automatic tiering or deletion after e.g. 1 year

Cost Tracking: Watch Cosmos RU usage + Blob/Table storage costs