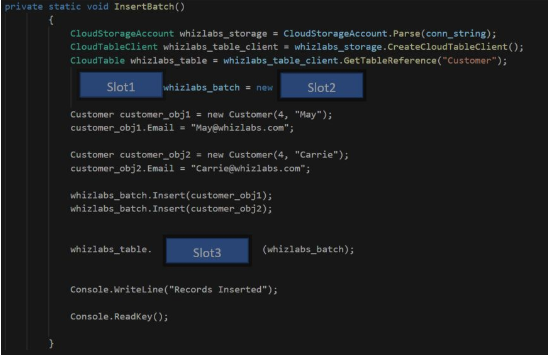


No:

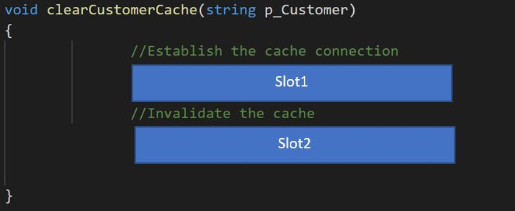
* Create SearchINDEXClient
* Create IndexBatch
* Call Documents.Index of SearchIndexClient, passing IndexBatch as parameter

Create Kubernetes cluster:

* az group create
* az aks create
* az aks get-credentials
* kubectl apply



Slot 1: TableBatchOperation  
Slot 2: TableBatchOperation  
Slot 3: ExecuteBatch



Redis cache  
Slot 1: IDatabase cache = Connection.GetDatabase();  
Slot 2: cache.KeyDelete(p\_Customer);

Create Dockerfile, with: application skillcertlabs.dll runs at startup, run a powershell script (both files are in the same directory as Dockerfile)

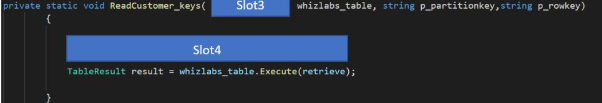
FROM microsoft/dotnet:2.-aspnetcore-runtime  
ENTRYPOINT[„dotnet”, „skillcertlabs.dll”]  
RUN powershell „skillcertlabsscript.ps1”

Querying from Azure storage (table Customers, partitioned by column firstname)  
Get all „Dave”s:

TableQuery.GenerateFilterCondition(„PartitionKey”, QueryComparions.Equal, „Dave”)

Table Storage:

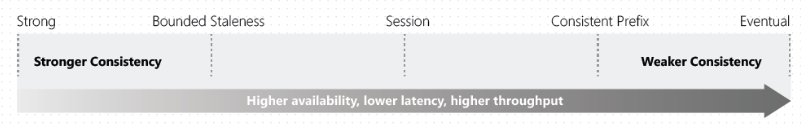
* What is the parition key: The field with which we want to load balance the data
* What is the row key: Must have value



Slot 3: CloudTable  
Slot 4: TableOperation retrieve = TableOperation.Retrieve(p\_partitionKey, p\_rowKey)

CosmosDB for hospital patients, requirements:

1. Status of patient must be the most recent (even if multiple users in different locations update the same patient’s records) 🡪 STRONG CONSISTENCY
2. Health of patient is recorded by one module, here it needs to be unsured that the data must be either the current version, or the previous version 🡪 BOUNDED STALENESS
3. If patient discharged, all charges to be processed, final bill processed 🡪 EVENTUAL CONSISTENCY



1. **Strong:** Reads return the most recent version (pricey)
2. **Bounded staleness:** Reads consistent with a **preconfigured lag**
   1. Lag can consist of a number of the most recent (K) versions or time interval (T)
3. **Session:** client session scope, best balance between strong consistency and performance by eventual consistency (best where writes occur in the context of a single user)
4. **Consistent Prefix:** Always read in the same order as I write (no guarantee that I can read all the data)
   1. If I write A, B, C then I can read: A or A,B or A,B,C (but never A,C or B,A,C)
5. **Eventual Consistency:** no guarantee for order

Azure Batch Service: create compute nodes 🡪 BatchClient.PoolOperations.CreatePool()