AWS DynamoDB Secondary Indexes

* DynamoDB provides fast access to items in a table by specifying primary key values
* DynamoDB Secondary indexes on a table allow efficient access to data with attributes other than the primary key
* DynamoDB Secondary indexes
  + is a data structure that contains a subset of attributes from a table
  + is associated with exactly one table, from which it obtains its data
  + requires an alternate key for the index partition key and sort key
  + additionally can define projected attributes which are copied from the base table into the index along with the primary key attributes
  + is automatically maintained by DynamoDB
  + any addition, modification, or deletion of items in the base table, any indexes on that table are also updated to reflect these changes.
  + helps reduce the size of the data as compared to the main table, depending upon the project attributes and hence helps improve provisioned throughput performance
  + are automatically maintained as **sparse** objects. Items will only appear in an index if they exist in the table on which the index is defined, making queries an index very efficient
* DynamoDB Secondary indexes supports two types
  + **Global secondary index –**an index with a **partition key and a sort key that can be different from those on the base table**
  + **Local secondary index –**an index that **has the same partition key as the base table, but a different sort key**

## DynamoDB Secondary Indexes - GSI vs LSI

## Global Secondary Indexes (GSI)

* DynamoDB creates and maintains indexes for the primary key attributes for efficient access of data in the table, which allows applications to quickly retrieve data by specifying primary key values.
* Global Secondary Indexes (GSI) are indexes that contain partition or composite partition-and-sort keys that can be different from the keys in the table on which the index is based.
* Global secondary index is considered “global” because queries on the index can span all items in a table, across all partitions.
* Multiple secondary indexes can be created on a table, and queries issued against these indexes.
* Applications benefit from having one or more secondary keys available to allow efficient access to data with attributes other than the primary key.
* GSIs support non-unique attributes, which increases query flexibility by enabling queries against any non-key attribute in the table
* GSIs support **eventual consistency**. DynamoDB automatically handles item additions, updates and deletes in a GSI when corresponding changes are made to the table **asynchronously**
* Data in a secondary index consists of GSI alternate key, primary key and  attributes that are projected, or copied, from the table into the index.
* Attributes that are part of an item in a table, but not part of the GSI key, primary key of the table, or projected attributes are not returned on querying the GSI index
* GSIs manage throughput independently of the table they are based on and the provisioned throughput for the table and each associated GSI needs to be specified at creation time
  + Read provisioned throughput
    - provides one Read Capacity Unit with two eventually consistent reads per second for items < 4KB in size.
    - provides one Write Capacity Unit with one write per second for items < 1KB in size.
  + Write provisioned throughput
    - consumes 1 write capacity unit if,
      * new item is inserted into table
      * existing item is deleted from table
      * existing items is updated for project attributes
    - consumes 2 write capacity units if
      * existing item is updated for key attributes, which results in deletion and addition of the new item into the index
* Throttling on a GSI affects the base table depending on whether the throttling is for read or write activity:
  + When a GSI has insufficient read capacity, the base table isn’t affected.
  + **When a GSI has insufficient write capacity, write operations won’t succeed on the base table or any of its GSIs.**

## Local Secondary Indexes (LSI)

* Local secondary index are indexes that has the same partition key as the table, but a different sort key.
* Local secondary index is “local” cause every partition of a local secondary index is scoped to a table partition that has the same partition key.
* LSI allows search using a secondary index in place of the sort key, thus expanding the number of attributes that can be used for queries which can be conducted efficiently
* LSI are updated automatically when the primary index is updated and reads support **strong, eventual and transactional consistenty options**
* LSIs can only be queried via the Query API
* LSIs cannot be added to existing tables at this time
* LSIs cannot be modified once it is created at this time
* LSI cannot be removed from a table once they are created at this time
* LSI consumes provisioned throughput capacity as part of the table with which it is associated
  + Read Provisioned throughput
    - if data read is index and projected attributes
      * provides one Read Capacity Unit with one strongly consistent read (or two eventually consistent reads) per second for items < 4KB
      * data size includes the index and projected attributes only
    - if data read is index and a non projected attribute
      * consumes double the read capacity, with one to read from the index and one to read from the table with the entire data and not just the non projected attribute
  + Write provisioned throughput
    - consumes 1 write capacity unit if,
      * new item is inserted into table
      * existing item is deleted from table
      * existing items is updated for project attributes
    - consumes 2 write capacity units if
      * existing item is updated for key attributes, which results in deletion and addition of the new item into the index

## AWS Certification Exam Practice Questions

1. In DynamoDB, a secondary index is a data structure that contains a subset of attributes from a table, along with an alternate key to support \_\_\_\_ operations.
   1. None of the above
   2. Both
   3. **Query**
   4. Scan
2. In regard to DynamoDB, what is the Global secondary index?
   1. **An index with a hash and range key that can be different from those on the table**
   2. An index that has the same range key as the table, but a different hash key
   3. An index that has the same hash key and range key as the table
   4. An index that has the same hash key as the table, but a different range key
3. In regard to DynamoDB, can I modify the index once it is created?
   1. Yes, if it is a primary hash key index
   2. **Yes, if it is a Global secondary index**(AWS now allows you to modify global secondary indexes after creation)
   3. No
   4. Yes, if it is a local secondary index
4. When thinking of DynamoDB, what are true of Global Secondary Key properties?
   1. **The partition key and sort key can be different from the table.**
   2. Only the partition key can be different from the table.
   3. Either the partition key or the sort key can be different from the table, but not both.
   4. Only the sort key can be different from the table.