CLOUD COMPUTING

Object Storage

Dr. Prafullata Kiran Auradkar

Department of Computer Science and Engineering

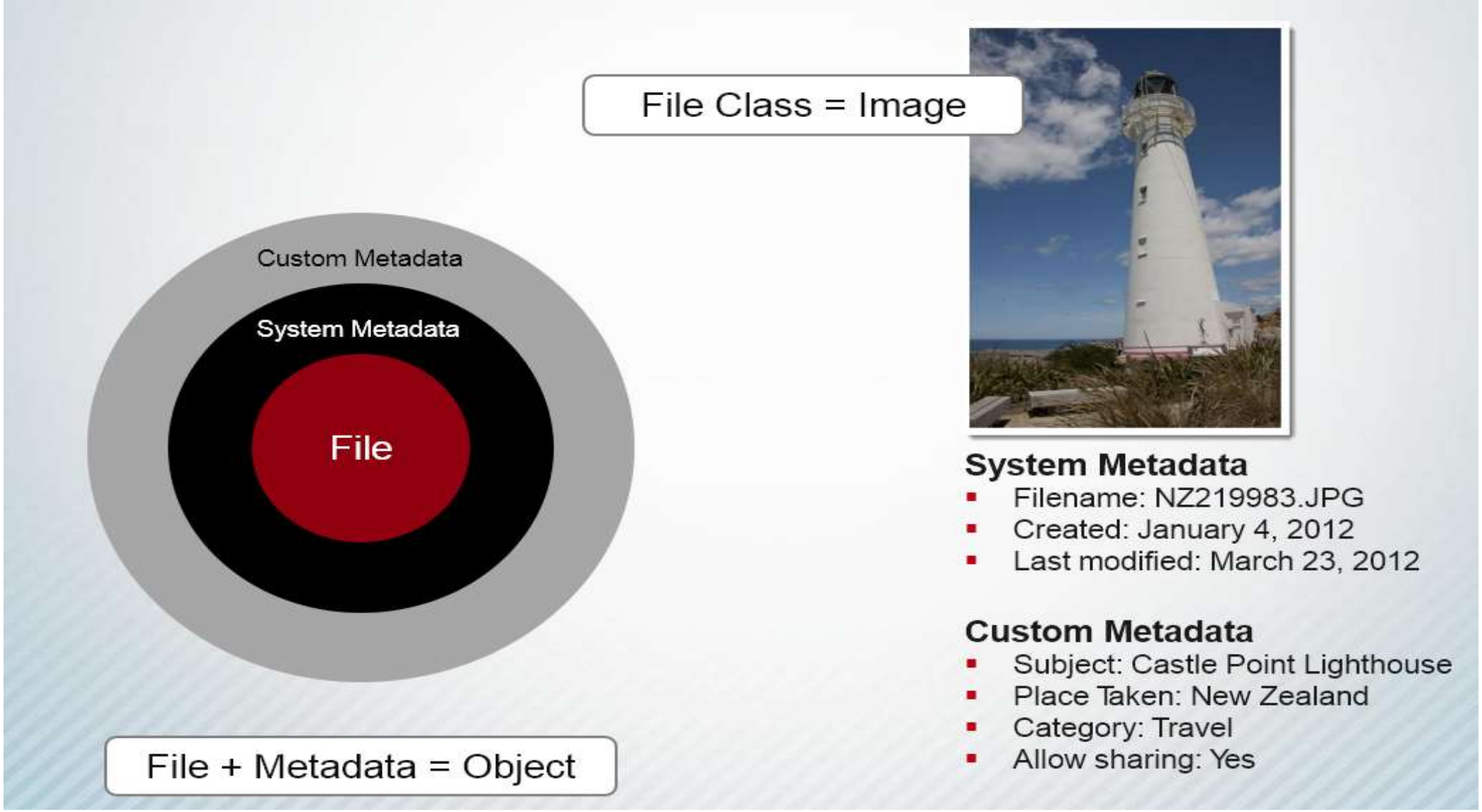
Acknowledgements:

Significant information in the slide deck presented through the Unit 3 of the course have been created by **Dr. H.L. Phalachandra** and would like to acknowledge and thank him for the same. There have been some information which I might have leveraged from the content of **Dr. K.V. Subramaniam's** lecture contents too. I may have supplemented the same with contents from books and other sources from Internet and would like to sincerely thank, acknowledge and reiterate that the credit/rights for the same remain with the original authors/publishers only. These are intended for classroom presentation only.

Object Storage

- Object storage is prominently used approach while building cloud storage systems
- Object storage is different from block or file storage as it allows a user to store data in the form of objects (essentially files in a logical view) by virtualizing the physical implementation in a flat namespace eliminating name collisions using REST HTTP APIs.

Object Storage Example of the Object



Object Storage

- Object storage :
  - Data manipulated using GET, PUT, DELETE, UPDATE
  - Use REpresentational State Transfer (REST) APIs
  - A distillation of the way the Web already works
    - Resources are identified by uniform resource identifiers (URIs)
    - Resources are manipulated through their representations
    - Messages are self-descriptive and stateless (XML)
    - Multiple representations are accepted or sent
- Objects contain additional descriptive properties which can be used for better indexing or management.
- Object storage also allows the addressing and identification of individual objects by more than just file name and file path.

Object Storage

- Object storage is built using scale-out distributed systems where each node, most often, actually runs on a local filesystem.
- Object storage does not need specialized & expensive hardware and the architecture allows for the use of commodity hardware
- The most critical tasks of an object storage system are :
  - Data placement
  - Automating management tasks, including durability and availability
- Typically, a user sends their HTTP GET, PUT, POST, HEAD, or DELETE request to any one node out from a set of nodes, and the request is translated to physical nodes by the object storage software.
- The object storage software also takes care of the durability model by doing any one of the following:
  - creating multiple copies of the object and chunking it
  - creating erasure codes (*Erasure coding (EC) is a method of data protection in which data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations or storage media*

  or a combination of these.
- Supports activities towards Management, such as periodic health checks, self-healing, and data migration.
- Management is also made easy by using a single flat namespace, which means that a storage administrator can manage the entire cluster as a single entity.

Object Storage : Illustration – Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (S3)

- Amazon S3 is a highly reliable, highly available, scalable and fast storage in the cloud for storing and retrieving large amounts of data just through simple web services.
- There are three ways of using S3. Most common operations can be performed via the AWS console (GUI interface to AWS)
- For use of S3 within applications, Amazon provides a REST-ful API with familiar HTTP operations such as GET, PUT, DELETE, and HEAD.
- There are libraries and SDKs for various languages that abstract these operations
- Since S3 is a storage service, several S3 browsers exist that allow users to explore their S3 account as if it were a directory (or a folder). There are also file system implementations that let users treat their S3 account as just another directory on their local disk.

Object Storage - Amazon S3 – Organizing Data in S3

Organizing Data In S3: Buckets, Objects and Keys

- Data is stored as **objects** in S3.
- These objects in S3 are stored in resources called **buckets**
- S3 objects can be up to 5 Terabytes in size and there are no limits on the number of objects that can be stored.
- Objects in S3 are replicated across multiple geographic locations to make it resilient to several types of failures.
- Objects are referred to with **keys** – basically an optional directory path name followed by the name of the object.
- If object versioning is enabled, recovery from inadvertent deletions and modifications is possible.

PES UNIVERSITY

Organizing Data In S3: Buckets, Objects and Keys (Cont.)

- Buckets provide a way to keep related objects in one place and separate them from others. There can be up to 100 buckets per account and an unlimited number of objects in a bucket.
- Each object has a key, which can be used as the path to the resource in an HTTP URL.
- **Example**: if the bucket is named johndoe and the key to an object is resume.doc, then its HTTP URL is http://s3.amazonaws.com/johndoe/resume.doc or alternatively, http://johndoe.s3.amazonaws.com/resume.doc
- By convention, slash-separated keys are used to establish a directory-like naming scheme for convenient browsing in S3

Object Storage - Amazon Simple Storage Service (S3)

Security

- Users can ensure the security of their S3 data by two methods
  1. **Access control to objects**: Users can set permissions that allow others to access their objects. This is accomplished via the AWS Management Console.
  2. **Audit logs**: S3 allows users to turn on logging for a bucket, in which case it stores complete access logs for the bucket in a different bucket. This allows users to see which AWS account accessed the objects, the time of access, the IP address from which the accesses took place and the operations that were performed. Logging can be enabled from the AWS Management Console

Object Storage - Amazon Simple Storage Service (S3)

Data Protection

1. **Replication**:
   - By default, S3 replicates data across multiple storage devices, and is designed to survive two replica failures.
   - It is also possible to request Reduced Redundancy Storage(RRS) for noncritical data. RRS data is replicated twice, and is designed to survive one replica failure.
   - S3 provides strong consistency if used in that mode and guarantees consistency among the replicas.
2. **Regions**:
   - For performance, legal, Availability and other reasons, it may be desirable to have S3 data running in specific geographic locations.
   - This can be accomplished at the bucket level by selecting the region that the bucket is stored in during its creation.

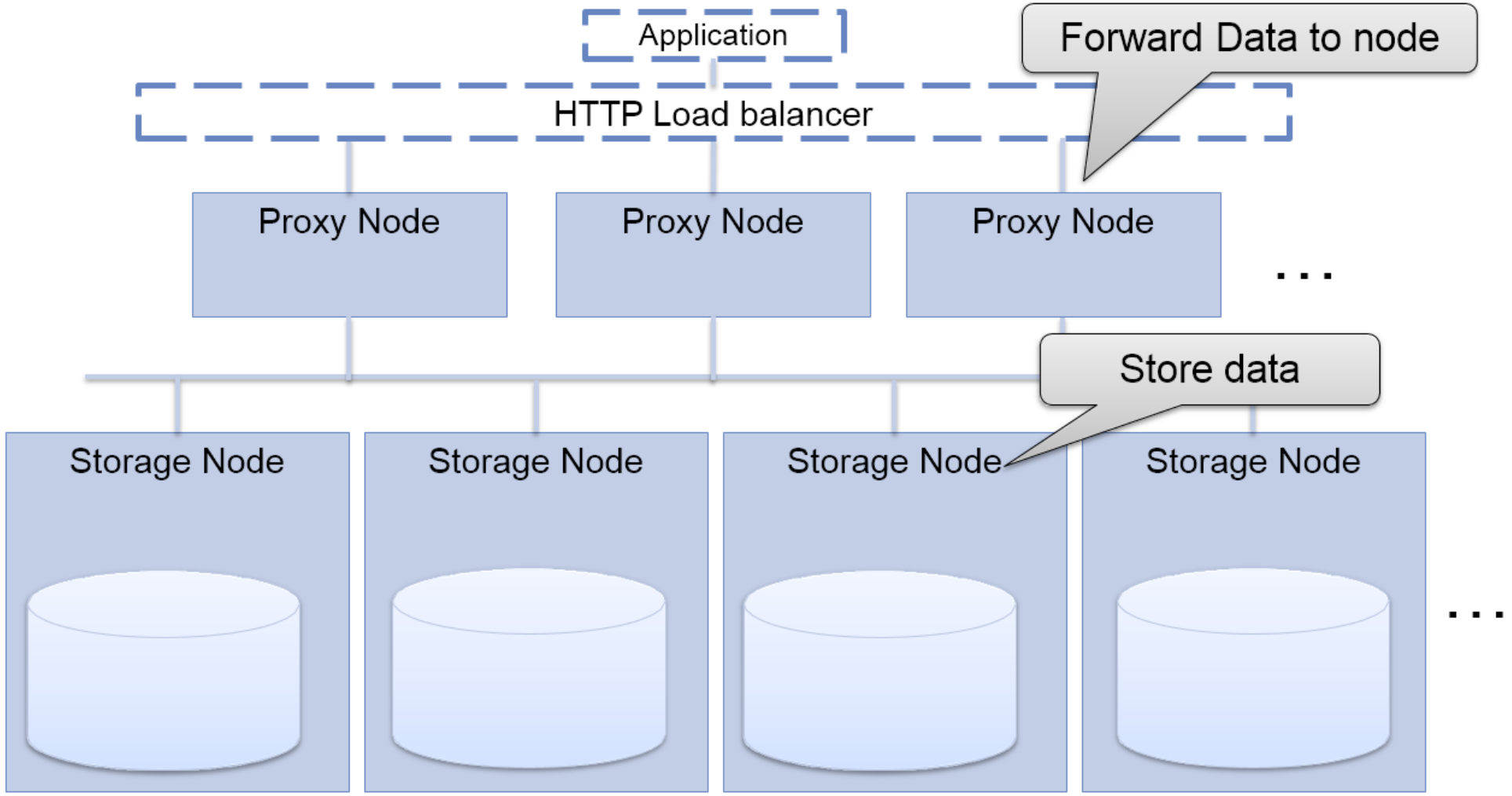
Data Protection

3. **Versioning:**
   - If versioning is enabled on a bucket, then S3 automatically stores the full history of all objects in the bucket from that time onwards.
   - The object can be restored to a prior version and even deletes can be undone. This guarantees that data is never inadvertently lost

Large Objects and Multi-part Uploads

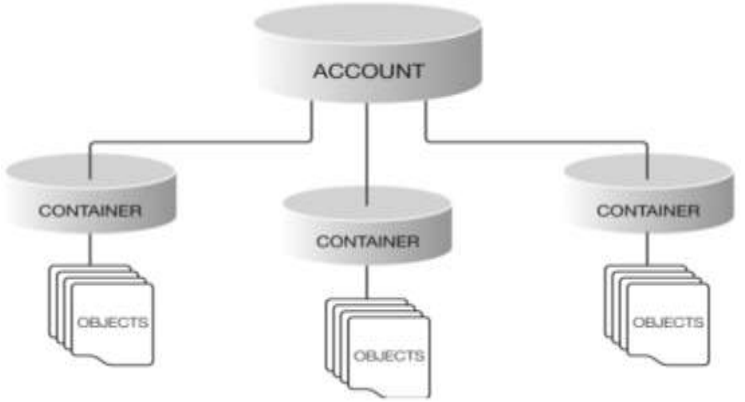
- S3 provides APIs that allow the developer to write a program that splits a large object into several parts and uploads each part independently.
- These uploads can be parallelized for greater speed to maximize the network utilization. If a part fails to upload, only that part needs to be re-tried

Open Stack Swift – Another Illustration of Object Storage

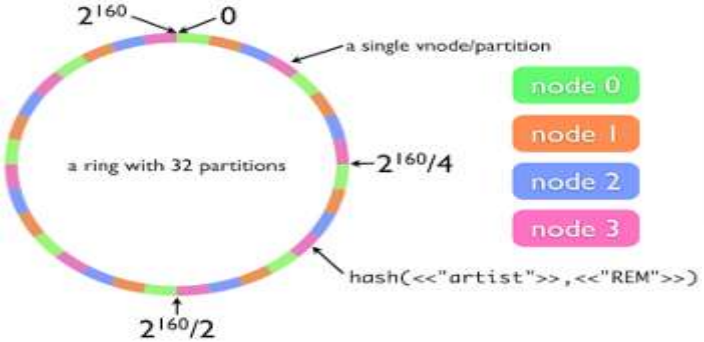


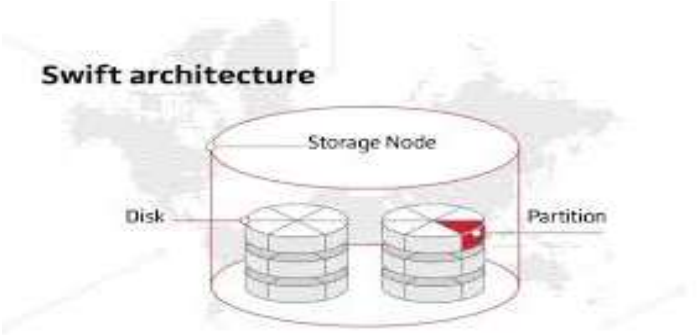
Swift

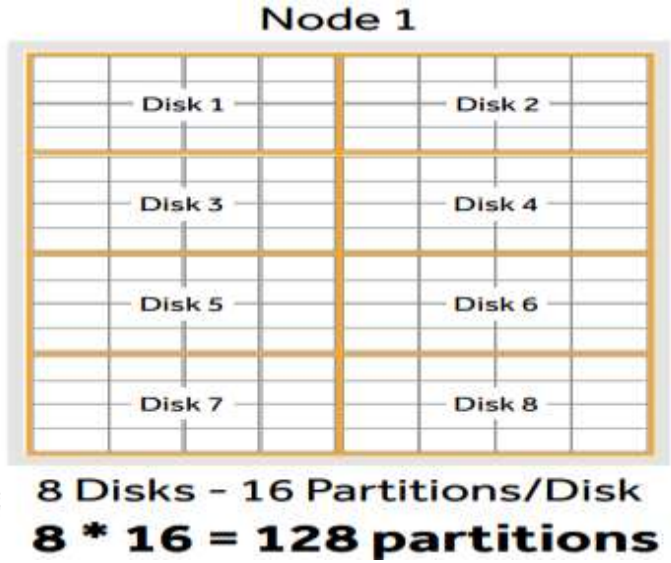
- **Swift Partitions** breaks the storage available into locations where data will be located including account databases, container databases or objects. It's the core of the replication system and distributed across all disks.
  Its like a moving bin moving in a warehouse. These are not linux partitions. Adding a node leads to reassigning of partitions



- **Account** is an user in the storage system- unlike volumes, swift creates accounts which enables multiple users and applications to access the storage system at the same time
- **Container:** Containers are where Accounts create and store data.
  Containers are name spaces used to group objects (conceptually like directori
- **Object :** Actual data is stored on the disk. This could be photos etc.
- **Ring** maps the partition space to physical locations on disk. Its like an encyclopedia, instead of letters swift used hash for searchin







NoSQL Database - DynamoDB

- DynamoDB is a NoSQL fully managed cloud-based document and a Key Value database available through Amazon Web Services (AWS)
- All data in DynamoDB is stored in tables that you have to create and define in advance, though tables have some flexible elements and can be modified later
- DynamoDB requires users to define only some aspects of tables, most importantly the structure of keys and local secondary indexes, while retaining a schema less flavor.
- DynamoDB enables users to query data based on secondary indexes too rather than solely on the basis of a primary key
- DynamoDB supports only item-level consistency, which is analogous to row-level consistency in RDBMSs
- If consistency across items is a necessity, DynamoDB is not the right choice
- DynamoDB has no concept of joins between tables. Table is the highest level at which data can be grouped and manipulated, and any join-style capabilities that you need will have to be implemented on the application side

CLOUD COMPUTING

DynamoDB

- In DynamoDB, tables, items, and attributes are the core components
- A table is a collection of items and each item is a collection of attributes
- DynamoDB uses primary keys to uniquely identify each item in a table and secondary indexes to provide more querying flexibility
- DynamoDB supports two different kinds of primary keys:
    - **Partition key** – A simple primary key, composed of one attribute known as the partition key. DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored.
    - **Partition key and sort key** – Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key and the second attribute is the sort key. All items with the same partition key value are stored together in sorted order by sort key value.

People

```
{
    "PersonID": 101,
    "LastName": "Smith",
    "FirstName": "Fred",
    "Phone": "555-4321"
}
```

```
{
    "PersonID": 102,
    "LastName": "Jones",
    "FirstName": "Mary",
    "Address": {
        "Street": "123 Main",
        "City": "Anytown",
        "State": "OH",
        "ZIPCode": 12345
    }
}
```

```
{
    "PersonID": 103,
    "LastName": "Stephens",
    "FirstName": "Howard",
    "Address": {
        "Street": "123 Main",
        "City": "London",
        "PostalCode": "ER3 5K8"
    },
    "FavoriteColor": "Blue"
}
```

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/dynamodb-dg.pdf#HowItWorks

DynamoDB

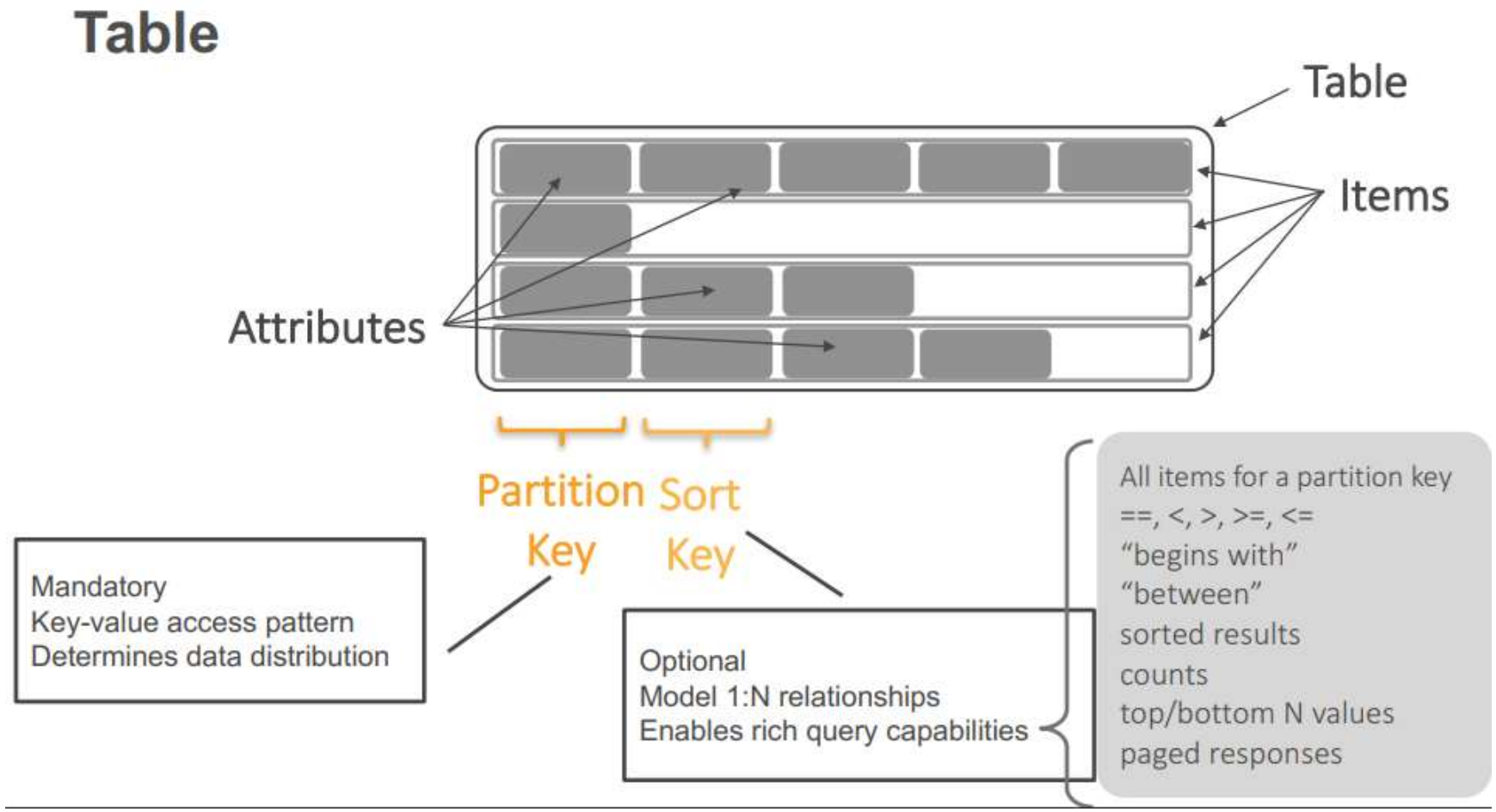
Secondary Indexes

- Users can create one or more secondary indexes on a table.
- A secondary index lets users query the data in the table using an alternate key, in addition to queries against the primary key.

Partitions and Data Distribution

- DynamoDB stores data in partitions.
- A partition is an allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across multiple Availability Zones within an AWS Region.
- Partition management is handled entirely by DynamoDB

DynamoDB Architecture



THANK YOU

Prafullata Kiran Auradkar

Department of Computer Science and Engineering

prafullatak@pes.edu