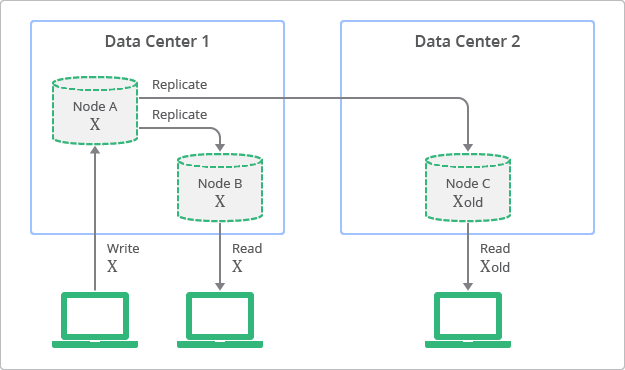
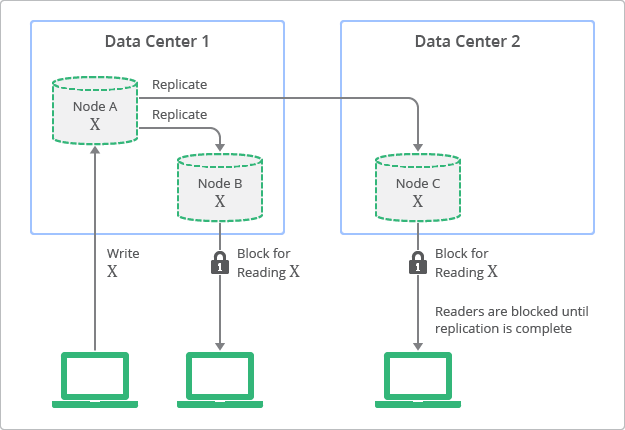
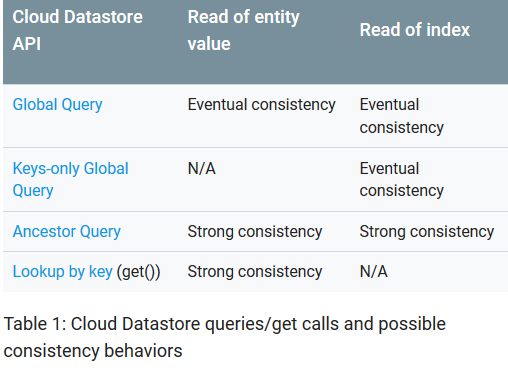
Project: Alice Speaks  
Topic: Notes on Designing a Datastore DB  
Created by: David Ramirez  
Date: 7/13/18  
Copyright © 2018 LeapWithAlice,LLC. All rights reserved

***Balancing Strong and Eventual Consistency with Google Cloud Datastore***

<https://cloud.google.com/datastore/docs/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/>

* Eventual Consistency
  + Eventual consistency is the theoretical guarantee that (provided no new updates to an entity are made) all reads of the entity will eventually return the last updated value
  + Depiction of Replication with Eventual Consitency
  + Although replicas are always available to read, some replicas may be inconsistent with the latest write on the originating node
    - This is in contrast to relational DBs that have strong consistency (immediate consistency) but at the cost of scalability
  + Depiction of replication with strong consistency
  + No-SQL better for high scalability, performance with high availability
* Balancing Eventual and Strong Consistency
  + In general, use cases with very large number of entities often suggest that eventual consistency is the best model
    - On the other hand, use cases with a small number of entities and a narrow context suggest that strong consistency is required
* Eventual Consistency in Cloud Datastore
  + 
  + Queries without an ancestor are known as **global queries**
  + **Keys-only global query** is a global query that returns only the keys of entities matching the query, not the attribute values of the entities
  + **Ancestor query** scopes the query based on an ancestor entity
* Eventual Consistency when Reading Entity Values
  + An updated entity value may not be immediately visible when executing a query (with the exception of ancestor queries)
    - This behavior is caused by the replication between Cloud Datastore servers
  + Usually, updates reach all replicas very quickly
    - Datacenter-wide incidents that involve switching over a large amount of servers between datacenters may negatively affect the replication time
  + Apps that use global queries should be carefully designed to handle cases in which replication time is long
  + Eventual consistency on **reading entity values** can be avoided by using a **keys-only query**, an **ancestor query**, or **looking up by key (the get() method)**
* Eventual Consistency on Reading an Index
  + An index may not yet be updated when a global query is executed
    - May be able to read the latest property values of the entities, but the “list of entities” included in the query result may be filtered based on old index values
  + Every entity update is essentially made up of 2 phases
    - 1) **commit phase**: write to the transaction log is performed
    - 2) **write phase**: data written and indexes are updated
    - If the **commit phase** succeeds, the **write phase** is guaranteed to succeed (but it may not happen immediately)
      * If an entity is queried before the indexes are updated, you may end up viewing old data
    - Time delay as a result of this 2 phase process is usually small, but may be longer (even minutes or more in exceptional circumstances)
  + Eventual consistency on **reading an index** can only be avoided by using an **ancestor query** or **lookup by key method** (performing a **keys-only query** will not avoid eventual consistency for reading an index)