Google Docs System Design

⇒ How are documents Stored?

↳ Refer Part-I for starting with Google Docs

↳ subscribe @MsDeepSingh on YouTube.

Storage Requirements

① Store document content

② User information like credentials, preferences

③ Document metadata like documentId, createdBy, etc.

④ User Sessions Database for websocket connection management

⑤ Document Collaboration Data — user actions or comments.

NOTE → I'm talking in terms of database choices from AWS context, Google Cloud has similar services — check link in video description.

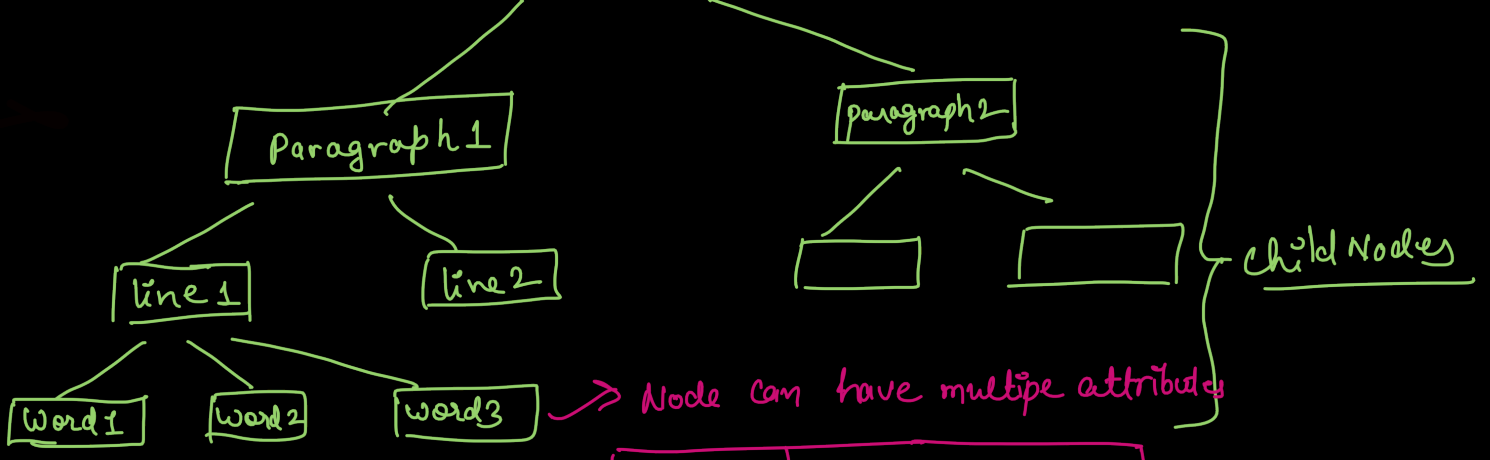
① Document Content Storage

- → Strong Consistency
- → High availability and scalability
- → Content stored in chunks for easy change management.

↓

One feasible way is to maintain a tree structure.

Complete doc - rootnode



Key	Value
Content	Word 3
is Bold	True
hyperlink	Https://@MsDeepSingh

Database Choice 1 → Relational DB Aurora → Google Cloud Spanner

① Document Table → documentId, title, document rootNode.

② Nodes Table → nodeId, parentNodeId, NodeType (→ paragraph, line etc), AttributesMap

③ Collab Table → documentId, collaboratorEmail, permissions

Database Choice 2 → DynamoDB → Google Big Table

partition Key - documentId

Sort Key - creation Time

```
{
  "documentId" : "12345",
  "title" : "subscribe to YT channel",
  "creationTime" : "2020",
  "Author" : "@MsDeepSingh",
  "Content" :
    [
      { "ContentSectionId" : "1",
        "type" : "text"
        "content" : "Hello",
      },
```

As document size grows, it will become difficult in continuous update for real-time collaborations.

```
{ ...
  ]
 }
}
```

Other DDB schema →

NodeId → Partition Key

parentNodeId → Sort Key

ChildNodeIds → NodeIds along with attributes

NodeId	Parent	ChildNodes
doc1	root	[section1, section2]
section1	doc1	[para1, para2]
para1	section1	[line1, line2]
line1	para1	[word1, word2]
word1	line1	[]

Database choice 3 – Document DB

Similar kind of structure

→ more complex query support.

Key consideration → RDBMS can provide just specific node entry update instead of querying for entire document.

the information around figuring out nodeId when multiple users are contributing can be derived with combination of users input + conflict handling by system.

(2) User Information → Aurora or RDS.

- (a) User email
- (b) Credentials
- (c) preferences

(3) Document Metadata → Dynamo or Aurora both are feasible.

- (a) document Id
- (b) created By
- (c) creation Date

ⓓ Access Permissions = {userId1, email, permissions. ...}

④ User sessions → Redis with persistent storage.

⑤ Collab Data → This will work in sync with Document Storage DB

ⓐ Comments

ⓑ Suggestions

ⓒ document Update Info → who updated specific document Node

Comments Storage

DocumentId	CommentId	NodeId → section/paragraph/line/word	Comment { author, date, content, }
partitionKey	SortKey		

Will this scale for maintaining comment replies?

↳ Each comment should have parent as well

Share your thoughts on how you'll think of databases and their schema in comments.

Subscribe for more - @msDeepSingh

@msdeep14

Happy Learning ☺