Resumen 4
Estudiante: Celina Madrigal Murillo
Carné: 2020059364

Schema-Agnostic Indexing with Azure DocumentDB

1. INTRODUCTION

Azure DocumentDB is Microsoft's multi-tenant distributed database service for managing JSON documents at Internet scale.
The indexing subsystem needs to support automatic indexing of documents without requiring a schema or secondary indices, DocumentDB's query language, real-time, consistent queries in the face of sustained high document ingestion rates, and multitenancy under extremely frugal resource budgets while still providing predictable performance guarantees and remaining cost effective.

1.1 Overview of the Capabilities

DocumentDB is based on the JSON data model and JavaScript language directly within its database engine. We believe this is crucial for eliminating the "impedance mismatch" between the application programming languages/type-systems and the database schema. Specifically, this approach enables the following DocumentDB capabilities:

- The query language supports rich relational and hierarchical queries.
- The database engine is optimized to serve consistent queries in the face of sustained high volume document writes.
- Transactional execution of application logic provided via stored procedures and triggers, authored entirely in JavaScript and executed directly inside DocumentDB's database engine.
- As a geo-distributed database system, DocumentDB offers well-defined and tunable consistency levels for developers to choose from and corresponding performance guarantees.
- As a fully-managed, multi-tenant cloud database service, all machine and resource management is abstracted from users.

1.2 Resource Model

A tenant of DocumentDB starts by provisioning a database account. A database account manages one or more DocumentDB databases. A DocumentDB database in-turn manages a set of entities: users, permissions and collections. A DocumentDB collection is a schema-agnostic container of arbitrary user generated documents. In addition to documents, a DocumentDB collection also manages stored procedures, triggers, user defined functions (UDFs) and attachments. Entities under the tenant's database account – databases, users, collections, documents etc. are referred to as resources. Each resource is uniquely identified by a stable and logical URI and is represented as a JSON document.

1.3 System Topology

The DocumentDB service is deployed worldwide across multiple Azure regions. We deploy and manage DocumentDB service on clusters of machines each with dedicated local SSDs. Upon deployment, the

DocumentDB service manifests itself as an overlay network of machines, referred to as a federation which spans one or more clusters.

1.4 Design Goals for Indexing

We designed the indexing subsystem of DocumentDB's database engine with the following goals:

- **Automatic indexing:** Documents within a DocumentDB collection could be based on arbitrary schemas.
- **Configurable storage/performance tradeoffs:** Although documents are automatically indexed by default, developers should be able to make fine grained tradeoffs between the storage overhead of index, query consistency and write/query performance using a custom indexing policy.
- **Efficient, rich hierarchical and relational queries:** The index should efficiently support the richness of DocumentDB's query APIs
- **Consistent queries in face of sustained volume of document writes:** For high write throughput workloads requiring consistent queries, the index needs to be updated efficiently and synchronously with the document writes.
- **Multi-tenancy:** Multi-tenancy requires careful resource governance.

2. SCHEMA AGNOSTIC INDEXING

2.1 No Schema, No Problem!

The schema of a document describes the structure and the type system of the document independent of the document instance. With a goal to eliminate the impedance mismatch between the database and the application programming models, DocumentDB exploits the simplicity of JSON and its lack of a schema specification. It makes no assumptions about the documents and allows documents within a DocumentDB collection to vary in schema, in addition to the instance specific values.

2.2 Documents as Trees

The technique which helps blurring the boundary between the schema of JSON documents and their instance values, is representing documents as trees.

2.3 Index as a Document

With automatic indexing, every path in a document tree is indexed. Each update of a document to a DocumentDB collection leads to update of the structure of the index.

2.4 DocumentDB Queries

Developers can query DocumentDB collections using queries written in SQL and JavaScript. Both SQL and JavaScript queries get translated to an internal intermediate query language called DocumentDB Query IL.