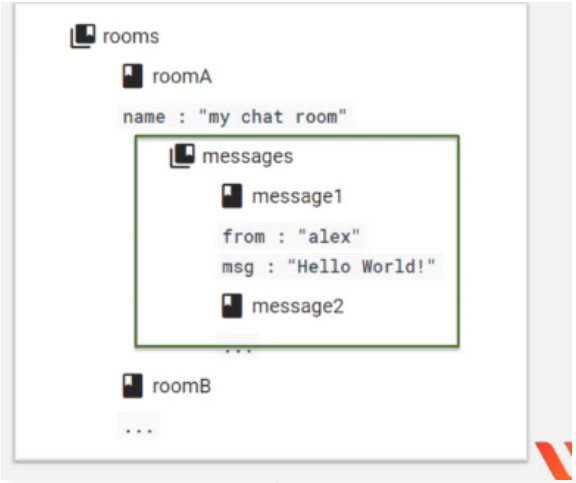
Datastore/Firestore

25 July 2021 20:46

Key Points Datastore/Firestore

1. Document and collection
2. Firetore Support nested document with each having document id autogenerated and sub-collection. Primitive data type like string and complex data type like list supported.
3. In firestore all fields are by default indexed (Single-field index -> a sorted mapping of field value)
4. We can exempt a field from single-field index by using single-field-index exemption. (Firestore)
5. Firestore support Composit index can be used to based on field order.
6. Previously eventual consistent but now strong consistency with firestore storage layer.
7. Contention can be resolved by failing one transaction at mobile sdk or by placing a lock in document at server side.
8. Support millions of concurrent client
9. Firestore support max 10,000 req per sec for datastore no limit.
10. Offline sync support present in firestore
11. Entity and Kind are used for datastore.
12. Scale automatically
13. Used for gaming and user profile data, indexing for objects in cloud storage
14. Batch operation are more suited
15. Choice of firestore is permanent
16. With multi-region 99.999% and with single region 99.99% availibility
17. All data by default encrypted at rest.
18. Datastore provide sql like query with no join support
19. In datastore mode we can have a root entity with related entity under it knows as entity group. For example a product entity and all its info
20. With firestore storage layer 1 transaction/sec per entity group limitation removed. Transaction are not limited to 25 entity group.
21. Firestore is real-time database for realtime notification.

Sub-collection structure



	Native mode	Datastore mode
	Enable all of Cloud Firestore's features, with offline support and real-time synchronization. SELECT NATIVE MODE	Leverage Cloud Datastore's system behavior on top of Cloud Firestore's powerful storage layer. SELECT DATASTORE MODE
API	Firestore	Datastore
Scalability	Automatically scales to millions of concurrent clients	Automatically scales to millions of writes per second
App engine support	Not supported in the App Engine standard Python 2.7 and PHP 5.5 runtimes	All runtimes
Max writes per second	10,000	No limit
Real-time updates	✓	✗

Mobile/web client libraries with offline data persistence	✓	✗
Query consistency	Strong	Strong
Data model	Documents / collections	Entities / kinds
Web console	Firestore page in Google Cloud Platform and Firebase	Datastore page in Google Cloud Platform

Limit and Costing

Datastore limits

If you have not yet upgraded from Datastore to Firestore in Datastore mode, the following limits also apply to your database instance:

Limit	Amount
Maximum number of entity groups that can be accessed in a transaction	25
Maximum rate of transactions reading from or writing to an entity group	1 per sec
Maximum write rate to an entity group. Note you can batch writes together for an entity group. This allows you to write multiple entities to an entity group within this limit.	1 per second
No of write per second 10,000	

Firestore in Datastore mode limits

Maximum size for a transaction: 10MB
Maximum depth of nested entity values: 20

Datastore/Firestore structure

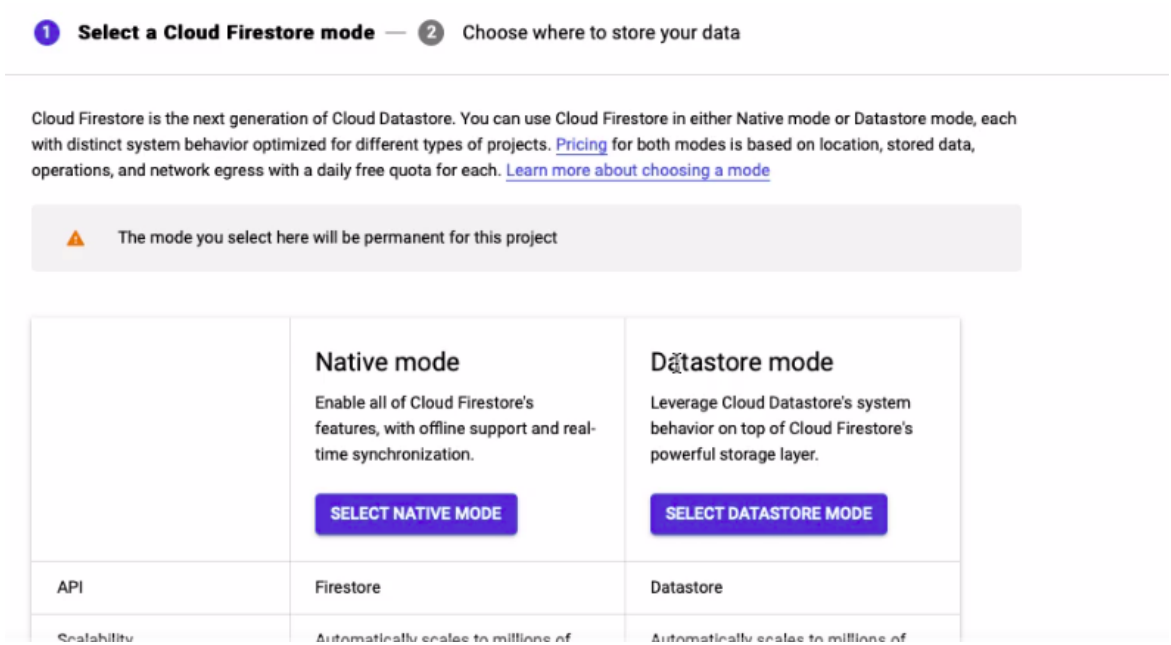
Concept	Cloud Datastore	Cloud Firestore	Relational database
Category of object	Kind	Collection group	Table
One object	Entity	Document	Row
Individual data for an object	Property	Field	Column
Unique ID for an object	Key	Document ID	Primary key

Different kind of query in datastore

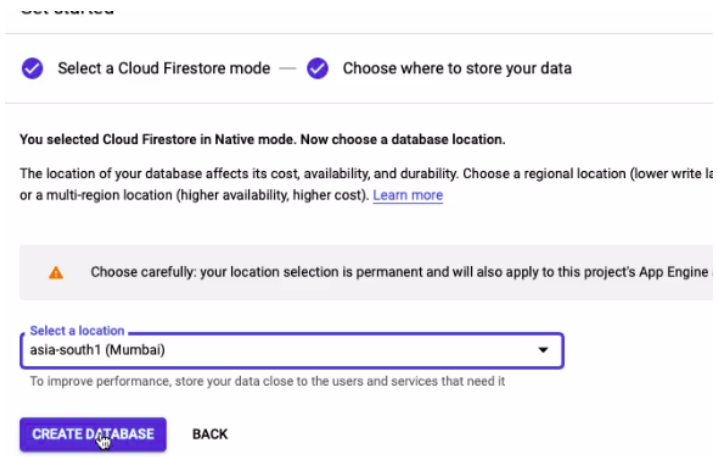
- **Ancestor queries: An ancestor query limits its results to the specified entity and its descendants**
- **Kindless queries: A query with no kind and no ancestor retrieves all of the entities of an application from Datastore mode**
- **Projection queries: Projection queries allow you to query for specific properties of an entity that you actually need**
- **Keys-only queries: A keys-only query returns just the keys of the result entities instead of the entities**

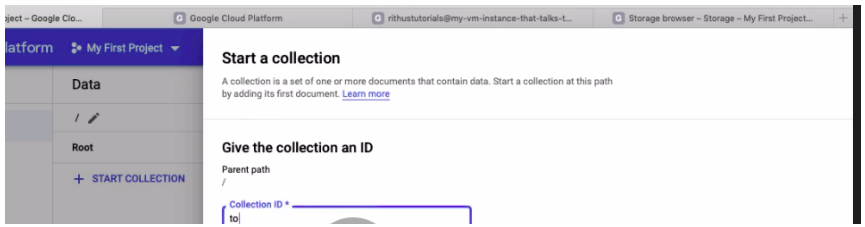
	themselves
Stackdriver logging supported	• Requests: Basic traffic monitoring for volume and any changes • Read Size: Size of entity reads grouped by type • Write Size: Size of entity writes grouped by type • Index Writes: Count of index writes from inbound writes and volume/changes Stackdriver
Spported data type firestore	Supported data types • Array • Boolean • Bytes • Date and time • Floating point number • Geographical point • Integer • Map – e.g. {a: "foo", b: "bar", c: "qux"}. • Null • Reference – e.g. projects/[PROJECT_ID]/databases/[DATABASE_ID] • Text string
Cloud Datastore Data Contention Data contention When two or more operations compete to control the same document. For example, one transaction might require a document to remain consistent while a concurrent operation tries to update that document's field values. Firestore resolves data contention by delaying or failing one of the operations. The Firestore client libraries automatically retry transactions that fail due to data contention. After a finite number of retries, the transaction operation fails and returns an error message: ABORTED: Too much contention on these documents. Please try again.	Cloud Datastore and Firestore • **Datastore** - Highly scalable NoSQL Document Database ▪ Automatically scales and partitions data as it grows ▪ Recommended for upto a few TBs of data ◦ For bigger volumes, BigTable is recommended ▪ Supports Transactions, Indexes and SQL like queries (GQL) ◦ Does NOT support Joins or Aggregate (sum or count) operations ▪ For use cases needing flexible schema with transactions ◦ Examples: User Profile and Product Catalogs ▪ Structure: Kind > Entity (Use namespaces to group entities) ▪ You can export data ONLY from gcloud (NOT from cloud console) ◦ Export contains a metadata file and a folder with the data • **Firestore** = Datastore++ : Optimized for multi device access ▪ Offline mode and data synchronization across multiple devices - mobile, IOT etc ▪ Provides client side libraries - Web, iOS, Android and more ▪ Offers Datastore and Native modes
Play with firestore	Select Firestore Mode

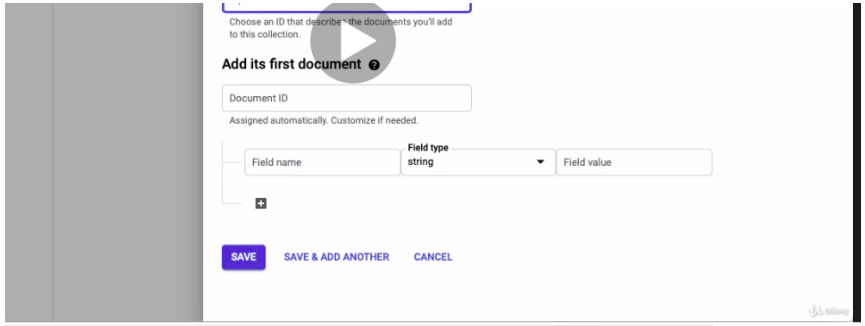
- Choice of firestore mode is permanent for the project. We can not change later.
- In Firestore a collection is a table and a document is a row
- Inside a document we can add field or a nested collection
- Indexing is by default added to all the fields , but we can create composite index

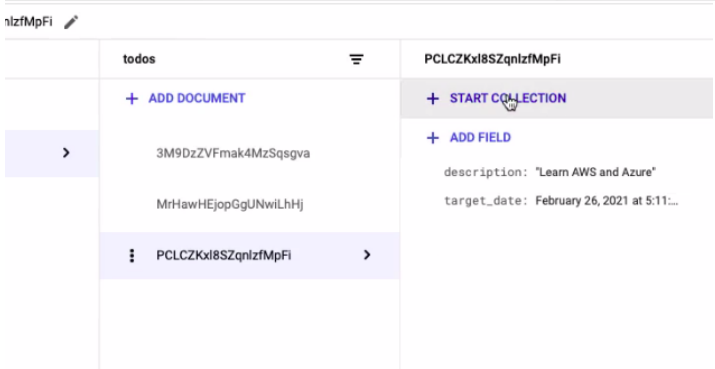


Select Location

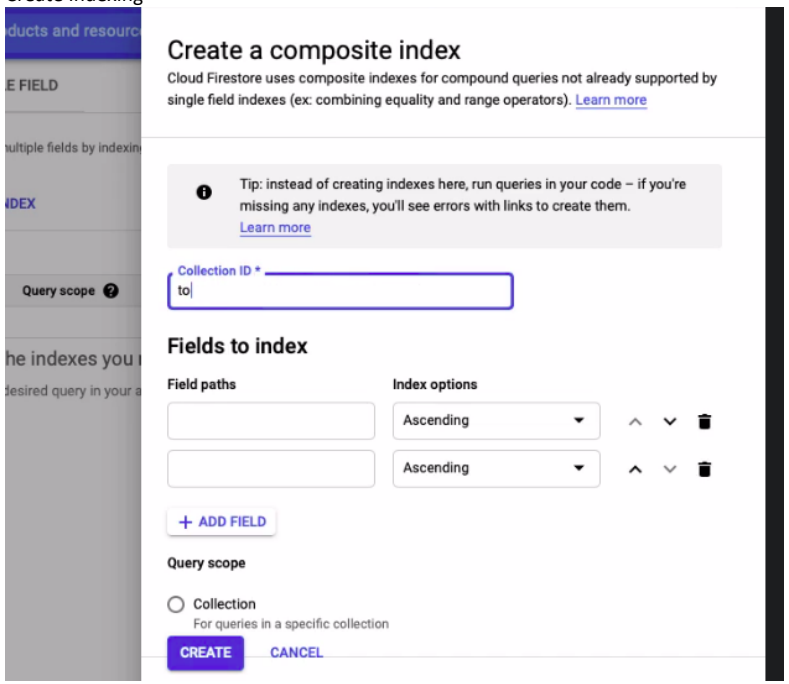




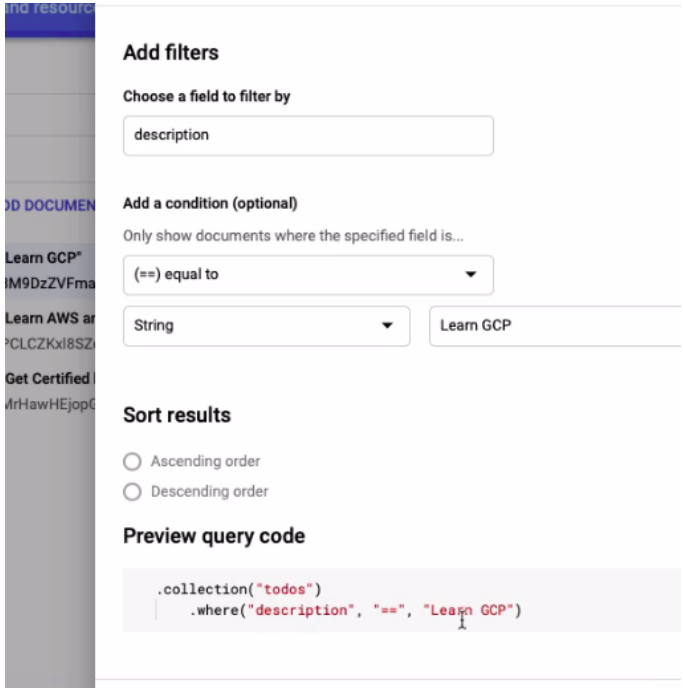




Create Indexing

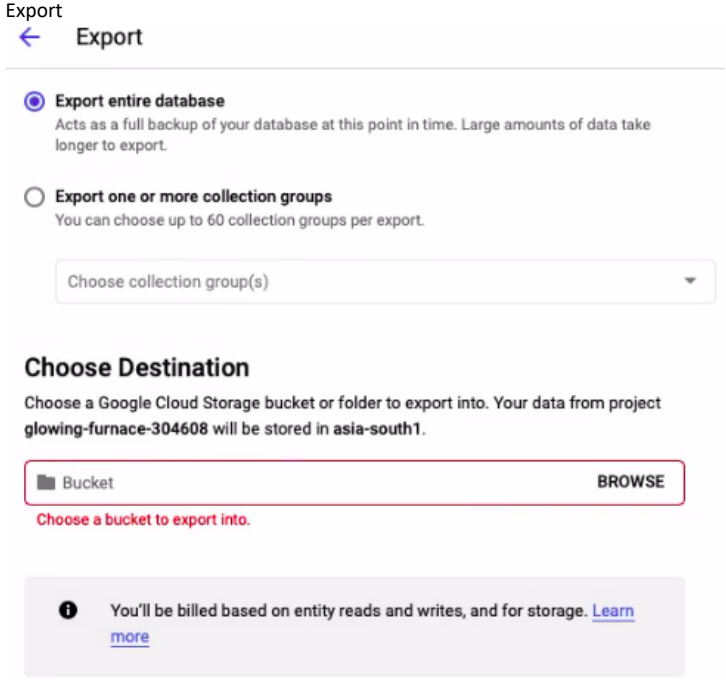


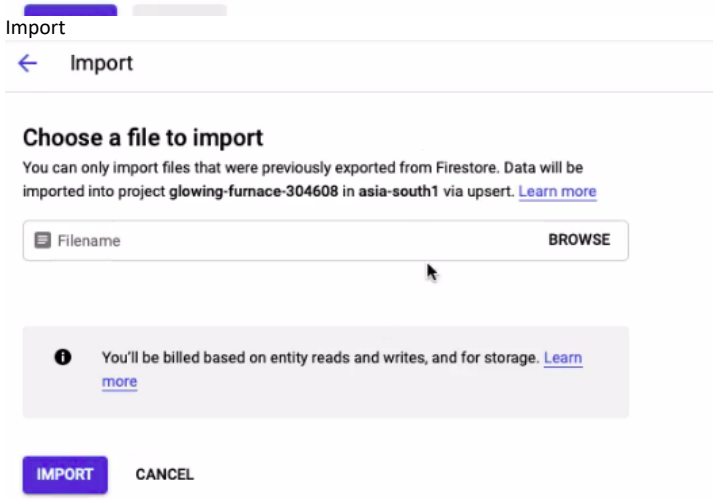
Add filter on doument to query



Import/ Export Firestore

Export



	Import 
Datastore best practices	**Understanding Cloud Datastore Best Practices** • Cloud Datastore is a **document store with flexible schema** ▪ Recommended for storing things like user profiles ▪ Another Use Case: Index for objects stored in Cloud Storage ○ You want to allow users to upload their profile pictures: ○ Store objects (pictures) in Cloud Storage ○ Enable quick search by storing metadata (like ids and cloud storage bucket, object details) in Cloud Datastore • Design **your keys and indexes** carefully: ▪ Avoid monotonically increasing values as keys ○ NOT RECOMMENDED - 1, 2, 3, …, OR "Customer1", "Customer2", "Customer3", … or timestamps ○ RECOMMENDED - Use allocateIds() for well-distributed numeric IDs ▪ Create indexes only if they would be used in queries ○ For ad hoc queries on large datasets without pre-defined indexes, BigQuery is recommended! • **Prefer batch operations** (to single read, write or delete operations): ▪ More efficient as multiple operations are performed with same overhead as one operation
Datastore emulator for local debugging The Datastore emulator provides local emulation of the production Datastore environment. You can use the emulator to develop and test your application locally.	
Datastore mode	Get started

Select a Cloud Firestore mode — 2 Choose where to store your data

You selected Cloud Firestore in Datastore mode. Now choose a database location.

The location of your database affects its cost, availability, and durability. Choose a regional location (lower write latency, lower cost) or a multi-region location (higher availability, higher cost). Learn more

Choose carefully: your location selection is permanent and will also apply to this project's App Engine app

Select a location
eur3 (Europe)

To improve performance, store your data close to the users and services that need it

CREATE DATABASE BACK

Create an entity

Namespace
[default]

Kind
Task

Key identifier
Numeric ID (auto-generated)

SPECIFY PARENT

Properties

ADD PROPERTY

CREATE CANCEL

Entities CREATE ENTITY DELETE

QUERY BY KIND QUERY BY GQL

Kind

Kind
Task

FILTER ENTITIES

	Name/ID ↑
☐	id=5634161670881280

Edit entity REFRESH DELETE

Namespace	[default]
Kind	Task
Key	Task id:5634161670881280
Key literal	Key(Task, 5634161670881280)
URL-safe key	ahdlfmRhdGFzdG9yZS1kZW1vLTI0NjcxMHIRCxIEVGFzaxiAgIDo14eBCgw

Properties

New property

Name *
description

Type
String

Value
Studying for the exam

☑ Index this property

CANCEL DONE

ADD PROPERTY

SAVE CANCEL

Edit entity REFRESH DELETE

Namespace	[default]
Kind	Task
Key	Task id:5634161670881280

Key literal Key(Task, 5634161670881280)

URL-safe key ahdlfmRhdGFzdG9yZS1kZW1vLTI0NjcxMHIRCxIEVGFzaxiAgIDo14eBCgw

Properties

description: Studying for the exam
Indexed

date: 2019-07-14 (11:49:00.000) UTC+1
Indexed
(Not saved)

New property

Name *
done

Type
Boolean

Value
False

Index this property

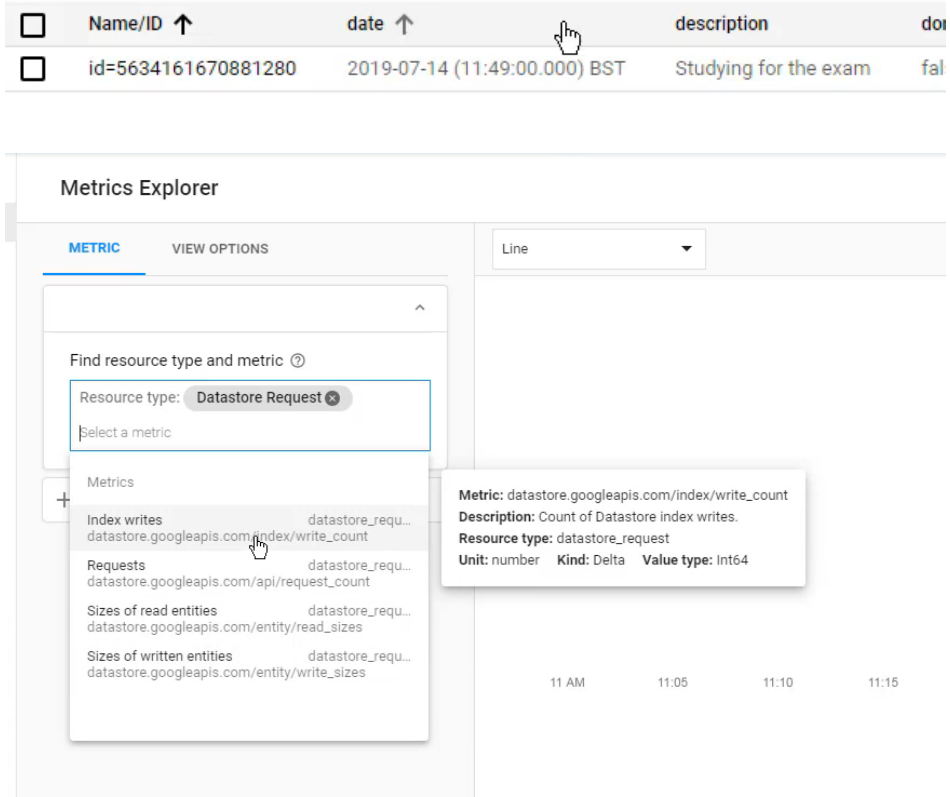
CANCEL DONE

ADD PROPERTY

SAVE CANCEL

```
1 select * from Task where done=false
```

RUN QUERY CLEAR QUERY GQL query help


Name/ID
date
description
id=5634161670881280
2019-07-14 (11:49:00.000) BST
Studying for the exam
Metrics Explorer
METRIC
VIEW OPTIONS
Line
Find resource type and metric
Resource type: Datastore Request
Select a metric
Metrics
Index writes
datastore.googleapis.com/index/write_count
Requests
datastore.googleapis.com/api/request_count
Sizes of read entities
datastore.googleapis.com/entity/read_sizes
Sizes of written entities
datastore.googleapis.com/entity/write_sizes
Metric: datastore.googleapis.com/index/write_count
Description: Count of Datastore index writes.
Resource type: datastore_request
Unit: number Kind: Delta Value type: Int64
11 AM
11:05
11:10
11:15