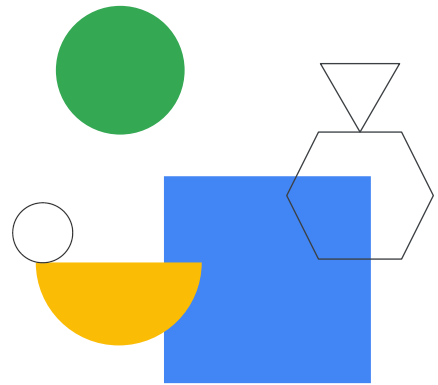


Choosing Storage and Data Solutions



In this module, we discuss Google Cloud storage and data solutions and how to select the most suitable one to meet business and technical requirements.

Learning objectives

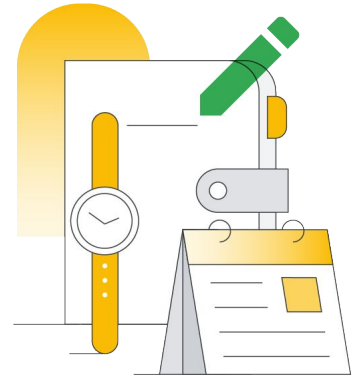
- 01 Choose the appropriate Google Cloud data storage service based on use case, durability, availability, scalability, and cost.
- 02 Store binary data with Cloud Storage.
- 03 Store relational data using Cloud SQL and Spanner.
- 04 Store NoSQL data using Firestore and Bigtable.
- 05 Cache data for fast access using Memorystore.
- 06 Aggregate data for queries and reports using BigQuery as a data warehouse.



This module examines the different storage options available through Google Cloud and provides guidelines on how to select a storage solution to meet requirements. Google Cloud provides a rich set of different storage options that cater to different types of data, sizes of data, lifecycles, and also data access patterns. This section details all and provides guidelines on how to select a suitable solution to meet requirements.

Agenda

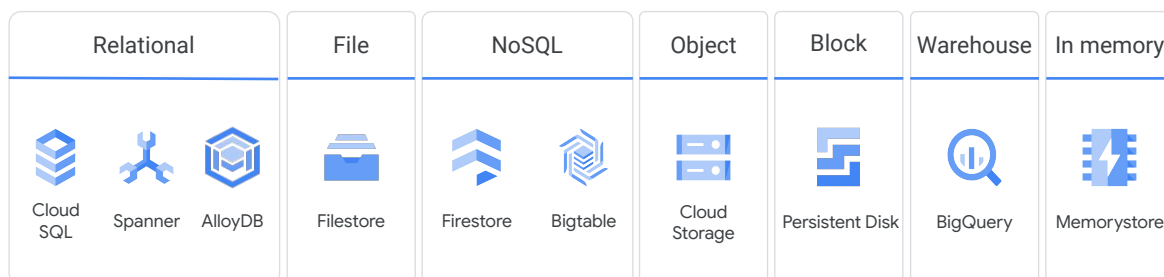
- | | |
|----|--|
| 01 | Key Storage Characteristics |
| 02 | Choosing Google Cloud Storage and Data Solutions |
| 03 | Quiz |
| 04 | Review |





Key Storage Characteristics

Google Cloud–managed storage and database portfolio



Google Cloud has a wide range of managed storage and database options. Knowing the characteristics of each and being able to select a suitable solution is vital as an architect. From a high level, the services range from in-memory and relational through to data warehouse options. These services are fully managed, scalable, and backed by industry-leading SLAs.

Making a decision on which storage solution is right for your requirements is a balance of a number of characteristics, including: type of data, scale, durability, availability, and location requirements. We discuss ways in which you can make the best decision based on your requirements.

Different data storage services have different availability SLAs

Storage Choice	Availability SLA %
Cloud Storage (multi-region bucket)	>=99.95
Cloud Storage (regional bucket)	99.9
Cloud Storage (coldline)	99.0
Spanner (multi-region)	99.999
Spanner (single region)	99.99
Firestore (multi-region)	99.999
Firestore (single region)	99.99

Google Cloud

When it comes to availability SLAs, much depends on the configuration of the service. Multi-region will increase the SLA over single region.

The availability SLAs are typically defined per month and calculated as follows.

"Monthly Uptime Percentage" means total number of minutes in a month, minus the number of minutes of downtime suffered from all downtime periods in a month, divided by the total number of minutes in a month.

Cloud SQL <https://cloud.google.com/sql/sla>

Spanner <https://cloud.google.com/spanner/sla>

Cloud Firestore <https://cloud.google.com/firestore/sla>

Bigtable: <https://cloud.google.com/bigtable/sla>

Cloud Storage <https://cloud.google.com/storage/sla>

BigQuery <https://cloud.google.com/bigquery/sla>

Memorystore <https://cloud.google.com/memorystore/sla>

Durability represents the odds of losing data

Preventing data loss is a shared responsibility.

Storage choice	Google Cloud provides	What you should do
Cloud Storage	11 9's durability Versioning (optional)	Turn versioning on
Disks	Snapshots	Schedule snapshot jobs
Cloud SQL	On-demand backups Automated backups Point-in-time recovery Failover server (optional)	Create at any time Run SQL database backups
Spanner	Automatic replication	Run export jobs
Firestore	Automatic replication	Run export jobs

Google Cloud

Durability of data represents the odds of losing the data. Depending on the storage solution, the durability is a shared responsibility. Google Cloud's responsibility is to ensure that data is durable in the event of a hardware failure. Your responsibility is performing backups of your data.

For example, Cloud Storage provides you with 11 9's durability, and versioning is a feature. However, it's your responsibility to determine when to use versioning. It is recommended that you turn versioning on and have older versions archived as part of an object lifetime management policy.

For other storage services to achieve durability, it usually means taking backups of data. For disks, this means snapshots, so snapshot jobs should be scheduled. For Cloud SQL, you can create a backup at any time (on-demand). This could be useful if you are about to perform a risky operation on your database, or if you need a backup and you do not want to wait for the backup window. Google Cloud also provides automated backups, point in time recovery, and optionally a failover server. You can create on-demand backups for any instance, whether the instance has automatic backups enabled or not. To improve durability, SQL database backups should also be run.

Spanner and Firestore provide automatic replication, and you should run export jobs with the data being exported to Cloud Storage.

The amount of data and number of reads and writes is important when selecting a data storage service

01

Some services scale **horizontally** by adding nodes.

- Bigtable
- Spanner

02

Some services scale **vertically** by making machines larger.

- Cloud SQL
- Memorystore

03

Some services scale **automatically** with no limits.

- Cloud Storage
- BigQuery
- Firestore

This slide discusses the scaling characteristics of the different storage solutions. It is also useful to consider the read and write patterns when selecting a solution.

Some applications, for example a retail sales application, will read and write data frequently. This style of application also has frequent updates. With the need for structured data, this type of storage need is best served by Cloud SQL. However, for a global database with relational read/writes, Spanner can be a better solution.

For data that is written in consistently high volumes and high rates, Bigtable may be the most suitable solution. For data that is written in files that are then downloaded in their entirety, Cloud Storage is a good option.

Do you need strong consistency?

Strongly consistent databases update all copies of data within a transaction.

Ensures everyone gets the latest copy of the data on reads.

- Storage
- Cloud SQL
- Spanner
- Firestore
- Bigtable (no instance replication)

Eventually consistent databases update one copy of the data and the rest asynchronously.

Can handle a large volume of writes.

- Bigtable (default behavior)
- Memorystore replicas

Strong consistency is another important characteristic to consider when designing data solutions. A strongly consistent database will update all copies of data within a transaction and ensure that everybody gets the latest copy of committed data on reads. Google Cloud services providing strong consistency include Cloud Storage, Cloud SQL, Spanner, and Firestore. If an instance does not use replication, Bigtable provides strong consistency, because all reads and writes are sent to the same cluster.

Eventually consistent databases typically have multiple copies of the same data for performance and scalability. They support handling large volumes of writes. They operate by updating one copy of the data synchronously and all copies asynchronously, which means that not all readers are guaranteed to read the same values at a given point in time. The data will eventually become consistent, but not immediately. Bigtable and Memorystore are examples of Google Cloud data services that have eventual consistency.

Replication for Bigtable is eventually consistent by default. This term means that when you write a change to one cluster, you will eventually be able to read that change from the other clusters in the instance, but only after the change is replicated between the clusters.

Calculate the total cost per GB when choosing a storage service

- Bigtable and Spanner would be too expensive for storing smaller amounts of data
- Firestore is less expensive per GB, but you also pay for reads and writes
- Cloud Storage is relatively cheap, but you can't run a database in storage
- BigQuery storage is relatively cheap, but doesn't provide fast access to records and you have to pay for running queries

You need to choose the right storage solutions for each of your microservices based on their requirements.

The cost of a storage service will depend on the amount of data stored, the amount of data retrieved or scanned, and the per-unit charges of the storage service. Realistically, selecting a storage solution is primarily based on type of data and consistency requirements, followed by evaluating cost if there are similar services.

The high-level summary for selecting a storage service is:

- For full SQL support for an online transaction processing (OLTP) system, consider Spanner or Cloud SQL.
- For interactive querying in an online analytical processing (OLAP) system, consider BigQuery.
- To store highly structured objects in a document database, with support for ACID transactions and SQL-like queries, consider Firestore.
- For in-memory data storage with low latency, consider Memorystore.
- To sync data between users in real time, consider the Firebase Realtime Database.
- For very high throughput and scalability for non-structured key-value data with minimally < 1TB data, consider Bigtable.

Activity 6

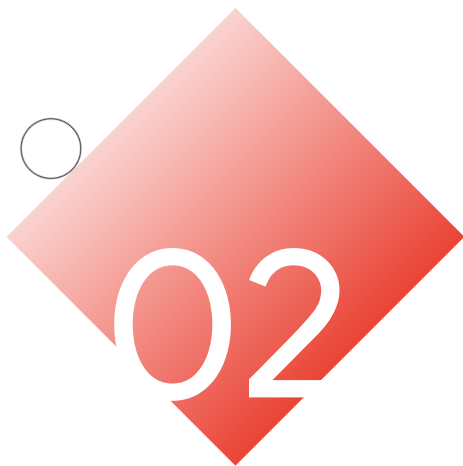
🕒 15 min

Defining storage characteristics

Refer to your Design and Process Workbook.











- Determine the storage characteristics for each of your case-study services.





Choosing Google Cloud Storage and Data Solutions

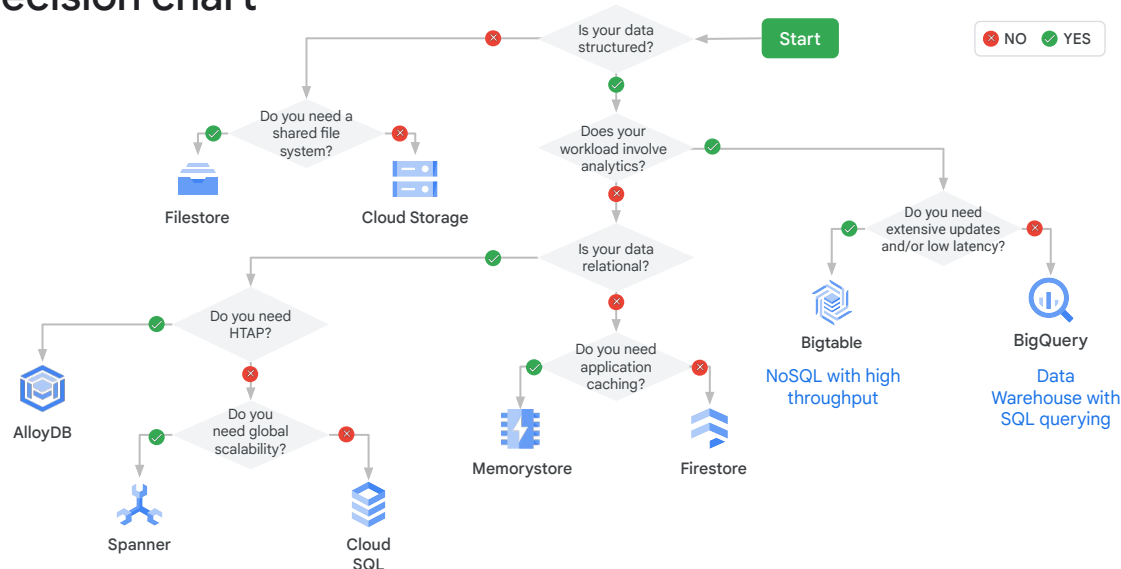
Google Cloud storage and database portfolio

Relational			File	NoSQL		Object	Block	Warehouse	In-memory
 Cloud SQL	 Spanner	 AlloyDB	 Filestore	 Firestore	 Bigtable	 Cloud Storage	 Persistent Disk	 BigQuery	 Memorystore
Good for: Web frameworks	Good for: RDBMS+scale, HA, HTAP	Good for: Hybrid transactional and analytical processing	Good for: Network Attached Storage (NAS)	Good for: Hierarchical, mobile, web	Good for: Heavy read + write, events	Good for: Binary object data	Good for: Applications requiring high performance, scalability and availability	Good for: Enterprise data warehouse	Good for: Caching for Web/Mobile apps
Such as: CMS, eCommerce	Such as: User metadata, Ad/Fin/MarTech	Such as: Machine learning, Generative AI	Such as: Latency sensitive workloads	Such as: User profiles, Game State	Such as: AdTech, financial, IoT	Such as: Images, media serving, backups	Such as: Data warehousing, Big data analytics	Such as: Analytics, dashboards	Such as: Game state, user sessions
Scales to 64 TB MySQL PostgreSQL SQL Server	Scales infinitely Regional or multi-regional	Secure, AI-powered performance	Fully managed, enterprise grade file storage service	Completely managed Document database	Scales infinitely Wide-column NoSQL	Completely managed Infinitely scalable	Powerful and versatile storage service	Completely Managed SQL analysis	Managed Redis DB
Fixed schema	Fixed schema	Fixed schema	Schemaless	Schemaless	Schemaless	Schemaless	Schemaless	Fixed schema	Schemaless

The above table provides a good summary of the characteristics of each storage class. A starting point in selecting a storage solution is often the type of data, which at a high level can be categorized as structured or unstructured, where the structure is defined by a schema and all records adhere to the schema. After this categorization, the size or amount of data to be stored needs to be considered.

So for example, if a requirement was for customer order information in a retail application, a fixed schema would be a likely choice. The amount of data to be stored is unlikely to be in the 10s of terabytes, and transactions are required. In this case, Cloud SQL seems a clear choice.

Decision chart



Google Cloud

Let's summarize the services in this module with this decision chart:

- First, ask yourself: Is your data structured, and will it need to be accessed using its structured data format? If the answer is no, then ask yourself if you need a shared file system. If you do, then choose Filestore.
- If you don't, then choose Cloud Storage.
- If your data is structured and needs to be accessed in this way, then ask yourself, does your workload focus on analytics? If it does, you will want to choose Bigtable or BigQuery, depending on your latency and update needs.
 - BigQuery is recommended as a data warehouse, is the default storage for tabular data, and is optimized for large-scale, ad-hoc SQL-based analysis and reporting. While BigQuery data manipulation language (DML) enables you to update, insert, and delete data from your BigQuery tables, because it has a built-in cache BigQuery works really well in cases where the data does not change often.
 - Bigtable is a NoSQL wide-column database. It's optimized for low latency, large numbers of reads and writes, and maintaining performance at scale.
 - In addition to analytics, Bigtable is also suited as a 'fast lookup' non-relational database for datasets too large to store in memory, with use cases in areas such as IoT, AdTech and FinTech.
- If your workload doesn't involve analytics, check whether your data is relational. If it's not relational, do you need application caching?
 - If caching is a requirement, choose Memorystore, an in-memory

- database.
 - Otherwise choose Firestore, a document database.
- If your data is relational and you need Hybrid transaction/analytical processing (HTAP) choose AlloyDB.
 - If you don't need HTAP and don't need global scalability, choose Cloud SQL.
 - If you don't need HTAP and need global scalability, choose Spanner.

Depending on your application, you might use one or several of these services to get the job done. For more information on how to choose between these different services, please refer to the following two links:

<https://cloud.google.com/storage-options/>

<https://cloud.google.com/products/databases/>

Transferring data into Google Cloud can be challenging

	1 Mbps	10 Mbps	100 Mbps	1 Gbps	10 Gbps	100 Gbps
1 GB	3 hrs	18 mins	2 mins	11 secs	1 sec	0.1 sec
10 GB	30 hrs	3 hrs	18 mins	2 mins	11 secs	1 sec
100 GB	12 days	30 hrs	3 hrs	18 mins	2 mins	11 secs
1 TB	124 days	12 days	30 hrs	3 hrs	18 mins	2 mins
10 TB	3 years	124 days	12 days	30 hrs	3 hrs	18 mins
100 TB	34 years	3 years	124 days	12 days	30 hrs	3 hrs
1 PB	340 years	34 years	3 years	124 days	12 days	30 hrs
10 PB	3,404 years	340 years	34 years	3 years	124 days	12 days
100 PB	34,048 years	3,404 years	340 years	34 years	3 years	124 days

Google Cloud

Transferring data into Google Cloud is not a trivial operation. Multiple factors must be considered, including cost, time, offline versus online transfer options, transfer tools and technologies, and security.

Cost

While transfer into Google Cloud is free, there will be costs with the storage of the data, possibly appliance costs if a transfer appliance is used, and possibly egress costs if transferring from another cloud provider.

Time

If you have huge datasets, the time required for transfer across a network may be unrealistic. Even if it is realistic, the effects on your organization's infrastructure may be damaging while the transfer is taking place. This needs to be considered. The table above shows the challenge of moving large data sets.

Offline vs. Online

The decision is whether to transfer using a network or storage hardware. Although several tools are available to support network transfer when only low bandwidth is available such as storage transfer service

(<https://cloud.google.com/storage-transfer/docs/overview>), Google also offers a hardware solution known as transfer appliance

<https://cloud.google.com/transfer-appliance/>. Here Google ships you hardware which you fill with data from your data center and ship back, where it is transferred to Cloud Storage. The data is encrypted until you choose to decrypt it.

Security and Privacy

There may be company policies that prevent transferring data over a public internet.

In that case, Direct Peering or Cloud Interconnect may be possible solutions.

Still there are other considerations such as protecting the data at rest (authorization and access to the source and destination storage system), protecting data while in transit, and protecting access to the transfer product. The general approach is to have the data encrypted and access gained only via access keys.

For smaller or scheduled data uploads, use the Cloud Storage Transfer Service

Import online data to Cloud Storage

- Amazon S3
- HTTP/HTTPS Location
- Transfer data between Cloud Storage buckets

Scheduled jobs

- One time or recurring, import at a scheduled time of day
- Options for delete objects not in source or after transfer
- Filter on file name, creation date

Google Cloud

Storage Transfer Service is a product that enables you to:

- Move or back up data to a Cloud Storage bucket either from other cloud storage providers or from your on-premises storage.
- Move data from one Cloud Storage bucket to another, so that it is available to different groups of users or applications.
- Periodically move data as part of a data processing pipeline or analytical workflow.

Storage Transfer Service provides options that make data transfers and synchronization easier. For example, you can:

- Schedule one-time transfer operations or recurring transfer operations.
- Delete existing objects in the destination bucket if they don't have a corresponding object in the source.
- Delete data source objects after transferring them.
- Schedule periodic synchronization from a data source to a data sink with advanced filters based on file creation dates, file-name filters, and the times of day you prefer to import data.

The gsutil utility also allows transfer of data between Cloud storage and other locations. In addition, Google has a Cloud Storage Transfer Service for on-premises data in beta

(<https://cloud.google.com/blog/products/storage-data-transfer/introducing-storage-transfer-service-for-on-premises-data>).

To help make the decision about which tool to use, consider the following:

- Transferring from another cloud storage provider: use Storage Transfer Service
- Transferring less than 1 TB from on-premises: use gsutil
- Transferring more than 1 TB from on-premises: use Storage Transfer Service for on-premises data (beta)

Use the Storage Transfer Service for on-premises data for large-scale uploads from your data center

- Install on-premises agent on your servers
- Agent runs in a Docker container
- Set up a connection to Google Cloud
- Requires a minimum of 300 Mbps bandwidth
- Scales to billions of files and 100s of TBs
- Secure
- Automatic retries
- Logged
- Easy to monitor via the Cloud Console

Google Cloud

The Storage Transfer Service for on-premises data allows large-scale online data transfers from on-premises storage to Cloud Storage. With this service, data validation, encryption, error retries, and fault tolerance are built in. On-premises software is installed—it comes as a Docker container—and then via the Cloud Console, directories to be transferred to Cloud Storage are selected. Once data transfer begins, the service will parallelize the transfer across many agents. Via the Cloud Console, a user can view detailed transfer logs and also the creation, management, and monitoring of transfer jobs.

To use the Storage Transfer Service for on-premises, a Posix-compliant source is required and a network connection of at least 300Mbps. Also, a Docker-supported Linux server that can access the data to be transferred is required with ports 80 and 443 open for outbound connections.

The use case is for on-premises transfer of data whose size is > 1TB.

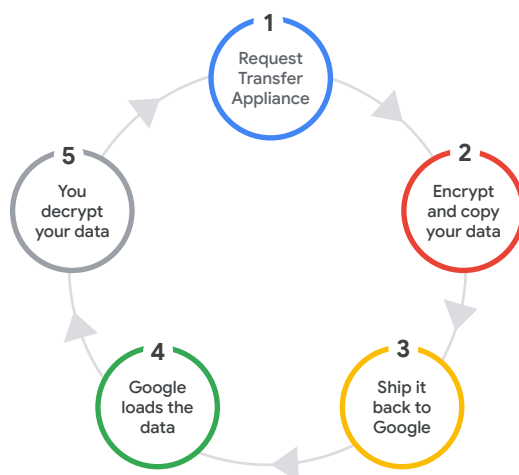
Use Transfer Appliance for large amounts of data

Rackable device up to 1PB shipped to Google.

Use Transfer Appliance if uploading your data would take too long.

Secure:

- You control the encryption key.
- Google securely erases the appliance after use.



Google Cloud

Transfer Appliance is a secure, rackable, high-capacity storage server that you set up in your data center. You fill it with data and ship it to an ingest location, where the data is uploaded to Cloud Storage. Data is encrypted automatically and remains safe until decrypted. Two sizes of appliance are available: 100 TB and 480 TB.

Transfer Appliance is easily mounted in rackspace in a data center and can be mounted as Network Attached Storage (NAS). A simple user interface is provided to guide users through local data capture, and the Cloud Console is used to decrypt and ingest data.

The process for using Transfer Appliance is that you request an appliance, and it is shipped in a tamper-evident case. Data is transferred to appliance. The appliance is shipped back to Google, data is loaded to Cloud Storage, and you are notified that it is available. Google uses tamper-evident seals on shipping cases to and from the data ingest site. Data is encrypted to AES256 standard at the moment of capture. Once the transfer is complete, the appliance is erased per NIST-800-88 standards.

There's also a transfer service for BigQuery

Source type

Choose a data source from the list below

Source *

Google Cloud Storage

This is the Google Cloud Storage configuration. [Learn more](#)

Transfer config name

Display name *

BigQuery Migration

Schedule options

☐ Start now
 ☒ Start at set time

Repeats *

Daily

Start date and run time *

11/30/19, 2:00 AM

EST

Destination settings

Select the destination for the transfer data

Destination dataset *

sales_dataset

Data source details

Destination table *

daily_sales

Cloud Storage URI *

☒ doug-rehnstrom-public/*-csv

☒ Delete source files after transfer

File format *

CSV

Transfer Options

All Formats

Number of errors allowed

0

JSON, CSV

☐ Ignore unknown values

CSV

Field delimiter

,

Header rows to skip

0

Google Cloud

The BigQuery Data Transfer Service automates data movement from SaaS applications to BigQuery on a scheduled, managed basis. The Data Transfer Service initially supports Google application sources like Google Ads, Campaign Manager, Google Ad Manager, and YouTube. There are also data connectors that allow easy data transfer from Teradata, Amazon Redshift, and Amazon S3 to BigQuery.

Activity 7

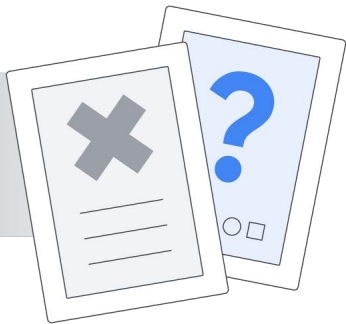
🕒 25 min

Choosing Google Cloud storage and data services

Refer to your Design and Process Workbook.

- Choose the storage services for each of your case-study services.





Quiz



Question #1

Question

Describe the difference between strong and eventual consistency.

Describe the difference between strong and eventual consistency.

Question #1

Answer

Describe the difference between strong and eventual consistency.

In a distributed database, there are multiple copies of the data. With strong consistency, when one copy changes, all the copies change before it is available for read. With eventual consistency, when data is updated, one copy changes and then the other copies are changed asynchronously.

Eventual consistency allows for faster writes, but it is possible for users to get stale data on a read.

Google Cloud

The answer provided above covers the main points. Non-relational databases, also known as NoSQL databases, have emerged in recent years as an alternative to relational databases. To achieve scalability, often these databases maintain multiple copies of the same distributed data. To developers more accustomed to relational databases, it may be challenging to design a system that leverages non-relational databases, because some characteristics and practices of non-relational databases may be relatively unfamiliar to them. Eventual consistency is one of these characteristics.

Eventual consistency is a theoretical guarantee that, provided no new updates to an entity are made, all reads of the entity will eventually return the last updated value. The internet Domain Name System (DNS) is a well-known example of a system with an eventual consistency model. DNS servers do not necessarily reflect the latest values, but rather, the values are cached and replicated across many directories over the internet. It takes a certain amount of time to replicate modified values to all DNS clients and servers.

In contrast, traditional relational databases have been designed based on the concept of strong consistency, also called immediate consistency. This means that data viewed immediately after an update will be consistent for all observers of the entity. This characteristic has been a fundamental assumption for many developers who use relational databases. However, to have strong consistency, developers must compromise on the scalability and performance of their application. Simply put, data has to be locked during the period of update or replication process to ensure that no

other processes are updating the same data.

Question #2

Question

You need to store user preferences, product information, and reviews for a website you are building. There won't be a huge amount of data. What would be a simple, cost-effective, managed solution?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

You need to store user preferences, product information, and reviews for a website you are building. There won't be a huge amount of data. What would be a simple, cost-effective, managed solution?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

Question #2

Answer

You need to store user preferences, product information, and reviews for a website you are building. There won't be a huge amount of data. What would be a simple, cost-effective, managed solution?

A. Firestore

B. Spanner

C. Cloud SQL

D. BigQuery



Google Cloud

- A. This answer is correct. Firestore provides automatic scale ACID transactions and live synchronization and is integrated with Google Cloud and Firebase. It also has a free tier.
- B. This answer is not correct. A key feature of Spanner is scale for relational data, and the requirement says there will not be a huge amount of data. Spanner would likely be more expensive than Cloud SQL without the need to use the scale Spanner provides.
- C. This answer is not correct. This could be a potential solution, but the flexibility of Firestore as a document store compared to the schema-based Cloud SQL is a better choice for the type of data, in particular for user preferences and reviews. Also the scale of Cloud SQL may be a limitation for peak loads, for instance Black Friday, or if the customer base grows. Finally, the pricing model is more expensive than Firestore.
- D. This answer is not correct. BigQuery is a data warehouse used for data analytics, and its model and pricing structure do not meet the application requirements..

Question #3

Question

You are a global financial services company with users all over the world. You need a database service that can provide low latency worldwide with strong consistency. Which service might you choose?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

You are a global financial services company with users all over the world. You need a database service that can provide low latency worldwide with strong consistency. Which service might you choose?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

Question #3

Answer

You are a global financial services company with users all over the world. You need a database service that can provide low latency worldwide with strong consistency. Which service might you choose?

- A. Firestore
- B. **Spanner**
- C. Cloud SQL
- D. BigQuery



- A. This answer is not correct. Firestore provides automatic scale ACID transactions, but the document data model is unlikely to be suitable for financial services.
- B. This answer is correct. A key feature of Spanner is scale for relational data with strong consistency, and it is globally distributed to provide low latency. The high availability and automatic replication are also strong features for financial services.
- C. This answer is not correct. Cloud SQL will not provide worldwide low latency because it is a regional service. Also, for financial services the replication and availability have to be managed, but with Spanner they are built in.
- D. This answer is not correct. BigQuery is a data warehouse used for data analytics, and its model and pricing structure do not meet the application requirements.

Question #4

Question

You want to analyze sales trends. To help achieve this, you want to combine data from your on-premises Oracle database with Google Analytics data and your web server logs. Where might you store the data so it is both easy to query and cost-effective?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

You want to analyze sales trends. To help achieve this, you want to combine data from your on-premises Oracle database with Google Analytics data and your web server logs. Where might you store the data so it is both easy to query and cost-effective?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery

Question #4

Answer

You want to analyze sales trends. To help achieve this, you want to combine data from your on-premises Oracle database with Google Analytics data and your web server logs. Where might you store the data so it is both easy to query and cost-effective?

- A. Firestore
- B. Spanner
- C. Cloud SQL
- D. BigQuery



The answers A, B, and C are not correct. Here there is a need to combine data from three different data sources, one at least of which is streaming. Firestore, Spanner, and Cloud SQL do not provide the features to ingest so it would be a lot more work to integrate and store than BigQuery.

D. This answer is correct. BigQuery is a data warehouse used for data analytics, and so is built for this type of use case. It provides the infrastructure to ingest data from many different sources, which is a requirement too. The cost model of paying for storage and then only for queries run is attractive too.

Question #5

Question

Currently, you are using Firestore to store information about products, reviews, and user sessions. You'd like to speed up data access in a simple, cost-effective way. What would you recommend?

- A. Move the data to Spanner.
- B. Move the data to BigQuery.
- C. Move the data to Bigtable.
- D. Cache the data using Memorystore.

Currently, you are using Firestore to store information about products, reviews, and user sessions. You'd like to speed up data access in a simple, cost-effective way. What would you recommend?

- A. Move the data to Spanner.
- B. Move the data to BigQuery.
- C. Move the data to Bigtable.
- D. Cache the data using Memorystore.

Question #5

Answer

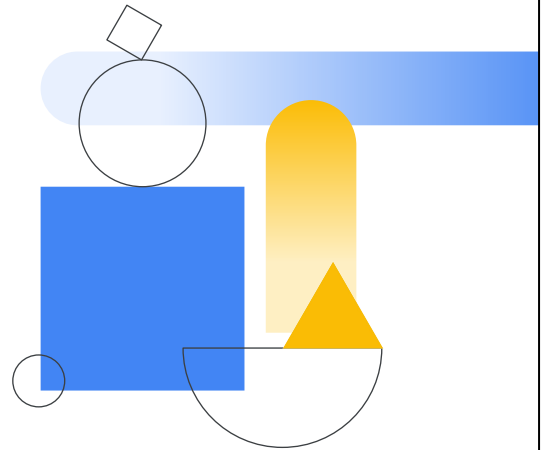
Currently, you are using Firestore to store information about products, reviews, and user sessions. You'd like to speed up data access in a simple, cost-effective way. What would you recommend?

- A. Move the data to Spanner.
- B. Move the data to BigQuery.
- C. Move the data to Bigtable.
- D. Cache the data using Memorystore.



- A. This answer is not correct. Spanner is for relational data at global scale. The data model and cost model are not the best fit for this requirement.
- B. This answer is not correct. BigQuery is for analytics of data rather than a datastore.
- C. This answer is not correct. The core requirement here is speed. Although Bigtable is low latency, it is designed for large-scale data, which is not the requirement here.
- D. This answer is correct. Memorystore provides the best fit when considering data model, performance, scale, cost, and availability.

Review: Choosing Storage and Data Solutions



In this module we covered the various storage services available in Google Cloud. These include Cloud Storage for binary data, Cloud SQL and Spanner for relational databases, Bigtable and Firestore for NoSQL databases, and BigQuery for data warehouses. We also talked about different storage characteristics and how they can be used to help you choose where to store your data.

More resources

Google Cloud Storage Options

<https://cloud.google.com/products/storage/>

Google Cloud Data Options

<https://cloud.google.com/products/databases/>



Here is a list of useful resources to find out more details on subjects discussed in this section.

