|  |  |  |
| --- | --- | --- |
| 9.Cloud SQL | Understanding of Cloud SQL | Overview |
| Tables, views |
| Replication Concept |
| Data loading from GCS bucket to cloud sql tables |
| Data exporting to GCS bucket using python |
| **Composer/Cloud SQL Lab** |

Cloud SQL is a fully-managed database service that helps you set up, maintain, manage, and administer your relational databases on Google Cloud Platform.

You can use Cloud SQL with [MySQL](https://cloud.google.com/sql/docs/mysql), [PostgreSQL](https://cloud.google.com/sql/docs/postgres), or [SQL Server](https://cloud.google.com/sql/docs/sqlserver).

This frees you from database administration tasks so that you have more time to manage your data.

**Use cases for Cloud SQL**

Cloud SQL provides a cloud-based alternative to local MySQL, PostgreSQL, and SQL Server databases. You should use Cloud SQL if you want to spend less time managing your database and more time using it.

Many applications running on Compute Engine, App Engine and other services in Google Cloud use Cloud SQL for database storage

**What Cloud SQL provides**

Cloud SQL offers many services so you don't have to build and maintain them yourself. You can focus on your data and let Cloud SQL handle the following operations:

* [Backups](https://cloud.google.com/sql/docs/mysql/backup-recovery/backups)
* [High availability and failover](https://cloud.google.com/sql/docs/mysql/high-availability)
* [Network connectivity](https://cloud.google.com/sql/docs/mysql/connect-overview)
* [Export and import](https://cloud.google.com/sql/docs/mysql/import-export)
* [Maintenance and updates](https://cloud.google.com/sql/docs/mysql/maintenance)
* [Monitoring](https://cloud.google.com/sql/docs/mysql/monitor-instance)
* [Logging](https://cloud.google.com/sql/docs/mysql/logging)

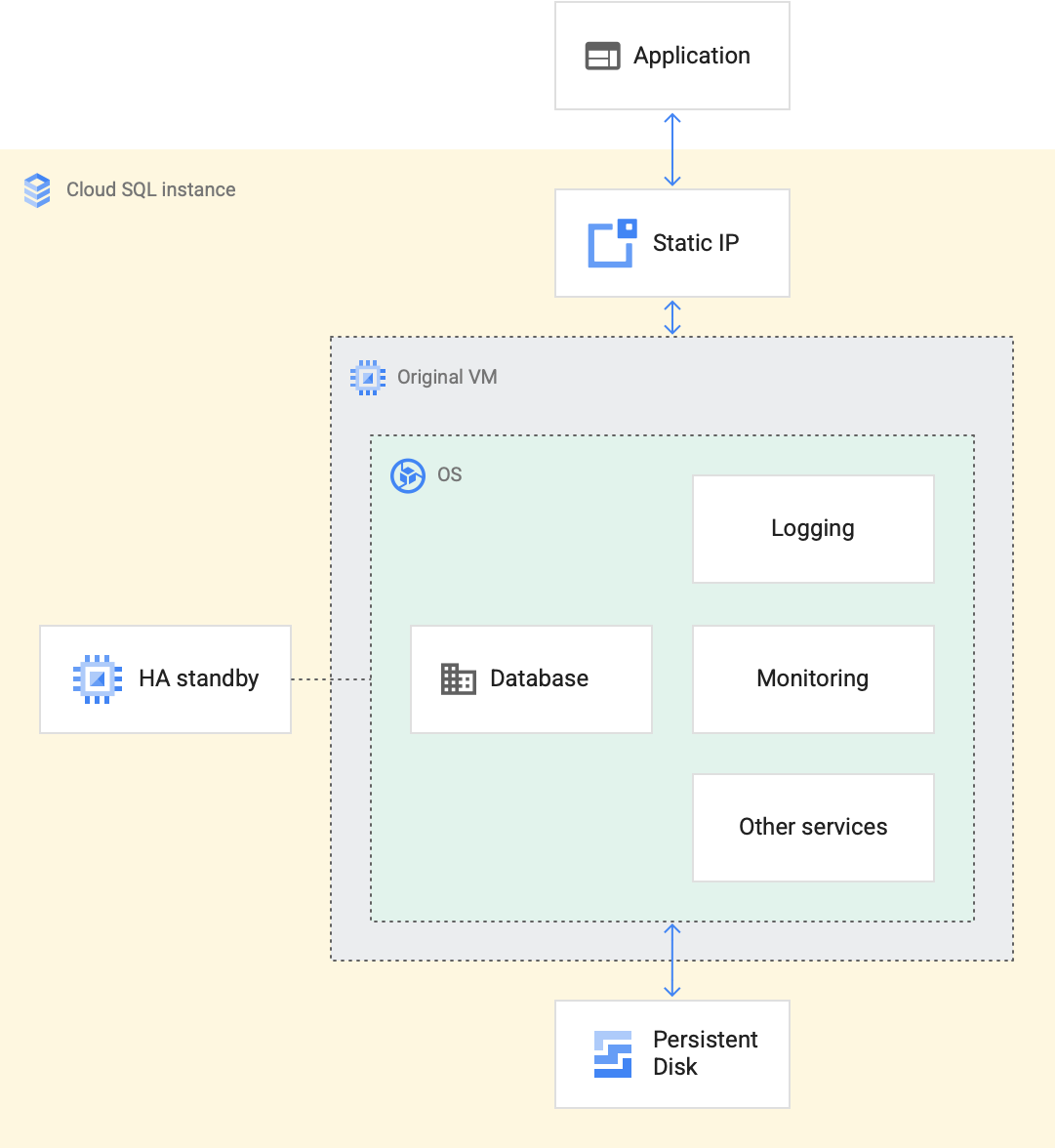
## What is a Cloud SQL instance?

Each Cloud SQL instance is powered by a virtual machine (VM) running on a host Google Cloud server.

Each VM operates the database program, such as MySQL Server, PostgreSQL, or SQL Server, and service agents that provide supporting services, such as logging and monitoring. The high availability option also provides a standby VM in another zone with a configuration that's identical to the primary VM.

The database is stored on a scalable, durable network storage device called a persistent disk that attaches to the VM.

A static IP address sits in front of each VM to ensure that the IP address an application connects to persists throughout the lifetime of the Cloud SQL instance.



**Database administration**

Cloud SQL lets you create and delete databases and database users, but it isn't a database administration tool. There are many database administration tools you can choose from, depending on your database engine, including the following:

* [phpMyAdmin](https://www.phpmyadmin.net/) for MySQL
* [MySQL Workbench](https://www.mysql.com/products/workbench/) for MySQL
* [Toad Edge](https://www.quest.com/products/toad-edge/) for MySQL and PostgreSQL
* [pgAdmin.org](https://www.pgadmin.org/) for PostgreSQL
* [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15) for SQL Server
* [Visual Studio Code](https://code.visualstudio.com/) for SQL Server

**Cloud SQL pricing**

Cloud SQL pricing varies with your configuration settings, and depends on:

* How much storage you provision, in GiB per month
* How many CPUs you select for your Cloud SQL instance
* How much memory you select for your Cloud SQL instance
* Where you choose to host your data
* How much network traffic leaves your instance
* How many IP addresses you assign and use

I apologize for the oversight. Let’s dive into a more detailed explanation of each topic related to **Google Cloud SQL (Cloud SQL)**:

1. **Overview**:
   * **Purpose and Use Case**:
     + Cloud SQL is a fully managed relational database service provided by Google Cloud.
     + It’s designed for applications that require a relational database (e.g., web applications, analytics, reporting).
   * **Managed Service**:
     + Google handles infrastructure, backups, and maintenance, allowing you to focus on your data and application.
     + No need to worry about server provisioning, patching, or scaling.
   * **Database Engines Supported**:
     + **MySQL**: Open-source relational database known for its performance, reliability, and ease of use.
     + **PostgreSQL**: Powerful, extensible open-source database with advanced features.
     + **SQL Server**: Microsoft’s relational database system.
2. **Features and Benefits**:
   * **High Availability and Failover**:
     + Cloud SQL provides automatic failover and replication for data durability.
     + If the primary instance fails, traffic is automatically redirected to a standby instance.
   * **Scalability**:
     + Easily scale your database vertically (more resources) or horizontally (read replicas).
     + Vertical scaling: Adjust CPU and memory resources.
     + Horizontal scaling: Create read replicas for improved read performance.
   * **Security**:
     + Data is encrypted at rest and in transit.
     + IAM-based access control for managing permissions.
     + Private IP connectivity for secure communication.
   * **Monitoring and Logging**:
     + Built-in monitoring and logging for performance insights.
     + View query performance, resource utilization, and more.
   * **Backups and Point-in-Time Recovery**:
     + Automated backups with configurable retention periods.
     + Restore to any point in time within the retention window.
   * **Network Connectivity**:
     + Accessible from GCP services, Compute Engine, and on-premises via VPN or Interconnect.
     + Supports private IP connectivity within VPCs.
   * **Maintenance and Updates**:
     + Google handles patching, maintenance, and version upgrades.
     + Scheduled maintenance windows for minimal impact.
3. **Cloud SQL Instance**:
   * Each Cloud SQL instance runs on a virtual machine (VM) within GCP.
   * Components:
     + **VM**: Hosts the database engine (e.g., MySQL, PostgreSQL) and supporting services (logging, monitoring).
     + **Persistent Disk**: Scalable, durable storage for the database.
     + **Static IP Address**: Ensures a consistent IP for connecting to the instance.
     + **High Availability Option**: Includes a standby VM in another zone for failover.
4. **Database Administration**:
   * While Cloud SQL simplifies many tasks, it’s not a full-fledged administration tool.
   * Use tools like phpMyAdmin (for MySQL) or MySQL Workbench for advanced administration tasks.
5. **Use Cases**:
   * **Application Databases**: Many apps on Compute Engine, App Engine, and other GCP services use Cloud SQL.
   * **Web Applications**: Ideal for web apps requiring a reliable and managed database.
   * **Analytics and Reporting**: Store data for reporting and analytics.
6. **Pricing**:
   * Pay for the resources (CPU, memory, storage) allocated to your instance.
   * Costs vary based on instance type, storage, and usage.

In summary, Cloud SQL simplifies database management, provides high availability, and allows you to focus on your data. Whether you’re building web apps, running analytics, or managing application databases, Cloud SQL offers a robust solution within GCP. If you have any specific questions or need further details, feel free to ask! 😊

### Cloud SQL for MySQL

Guides

Create instances

Connection Overview

Enable and disable high availability on an instance

Create and manage MySQL databases

Create and manage MySQL users

Export and import using SQL dump file

Export and import using CSV files

Create backups

Create read replicas

A newly-created instance has four system databases:

* [information\_schema](https://dev.mysql.com/doc/refman/5.7/en/information-schema.html): Provides access to database metadata, information about the MySQL server.
* [mysql](https://dev.mysql.com/doc/refman/5.7/en/system-schema.html): The system schema. It contains tables that store information required by the MySQL server as it runs.
* [performance\_schema](https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html): A feature for monitoring MySQL Server execution at a low level.
* [sys](https://dev.mysql.com/doc/refman/5.7/en/sys-schema.html): Contains a set of objects that helps DBAs and developers interpret data collected by the performance schema.
* Use the [gcloud sql instances create](https://cloud.google.com/sdk/gcloud/reference/sql/instances/create) command to create the instance:  
  For Cloud SQL Enterprise Plus edition instances:
* **gcloud sql instances create *INSTANCE\_NAME* \  
  --region=*REGION* \  
  --tier=*TIER* \  
  --database-version=*MYSQL\_8\_0\_31* \  
  --edition=ENTERPRISE\_PLUS**

### Cloud SQL Overview:

Google Cloud SQL is a fully-managed relational database service for MySQL, PostgreSQL, and SQL Server. It offers automated backups, built-in security, and high availability.

### Tables and Views:

#### 1. Creating Tables:

```sql

CREATE TABLE employees (

employee\_id INT NOT NULL,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

PRIMARY KEY (employee\_id)

);

```

#### 2. Creating Views:

```sql

CREATE VIEW employee\_names AS

SELECT first\_name, last\_name FROM employees;

```

### Replication Concept:

Cloud SQL replication allows you to create and maintain multiple copies of your data in different locations, providing high availability, disaster recovery, and read scaling. There are two types of replication in Cloud SQL:

1. \*\*Asynchronous replication:\*\*

- In asynchronous replication, transactions are committed on the primary instance and then asynchronously replicated to the standby instances.

- Suitable for disaster recovery and read scaling.

- May result in some data loss if the primary instance fails before data is replicated to the standby instance.

2. \*\*Synchronous replication:\*\*

- In synchronous replication, a transaction is not considered committed on the primary instance until it has been replicated to all standby instances.

- Ensures no data loss if the primary instance fails.

- More restrictive than asynchronous replication because it waits for acknowledgment from all standby instances before committing transactions.

- Suitable for regional instances.

#### Practical Example for Asynchronous Replication:

```bash

# Create the primary instance

gcloud sql instances create primary-instance \

--region=us-central1 \

--database-version=MYSQL\_5\_7 \

--tier=db-n1-standard-2 \

--storage-type=SSD \

--storage-size=10GB \

--backup-start-time=23:00

# Enable binary log on the primary instance

gcloud sql instances patch primary-instance \

--enable-bin-log \

--backup-start-time=23:00

# Create the read replica (asynchronous)

gcloud sql instances create replica-instance \

--region=us-central1 \

--master-instance-name=primary-instance \

--tier=db-n1-standard-2 \

--storage-type=SSD \

--storage-size=10GB

# Promote the read replica to a standalone instance if needed

gcloud sql instances promote-replica replica-instance

```

### Data Loading from GCS Bucket to Cloud SQL Tables:

#### 1. First, ensure you have the required CSV file in your Google Cloud Storage (GCS) bucket.

#### 2. Create a Cloud SQL table:

```sql

CREATE TABLE employees (

employee\_id INT NOT NULL,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

PRIMARY KEY (employee\_id)

);

```

#### 3. Loading data from a CSV file in a GCS bucket into the Cloud SQL table:

you can use the `gcloud sql import csv` command to load data from a Google Cloud Storage (GCS) bucket into a Cloud SQL table. Here's how you can do it:

```bash

**gcloud sql import csv [INSTANCE\_NAME] gs://[BUCKET\_NAME]/[FILE\_NAME] --database=[DATABASE\_NAME] --table=[TABLE\_NAME] --user=[USERNAME]**

```

Replace the following placeholders with your actual information:

- `[INSTANCE\_NAME]`: Replace this with the name of your Cloud SQL instance.

- `[BUCKET\_NAME]`: Replace this with the name of your GCS bucket.

- `[FILE\_NAME]`: Replace this with the name of your CSV file.

- `[DATABASE\_NAME]`: Replace this with the name of your Cloud SQL database.

- `[TABLE\_NAME]`: Replace this with the name of your Cloud SQL table.

- `[USERNAME]`: Replace this with the username for the Cloud SQL instance.

Example:

```bash

**gcloud sql import csv my-instance gs://my-bucket/data.csv**

**--database=my\_database**

**--table=my\_table**

**--user=my\_user**

```

Ensure that the Cloud SQL instance has the necessary permissions to access the GCS bucket.

```sql

LOAD DATA INFILE 'gs://your\_bucket/your\_file.csv'

INTO TABLE employees

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

(employee\_id, first\_name, last\_name);

```

### Data Exporting to GCS Bucket using Python:

Here's how you can export data from a Cloud SQL table to a CSV file in a GCS bucket using Python:

#### Prerequisites:

- Make sure you have the necessary IAM permissions.

- Install the required libraries:

```bash

pip install google-cloud-storage mysql-connector-python

```

#### Python Script:

```python

import os

import mysql.connector

from google.cloud import storage

# Google Cloud Storage parameters

BUCKET\_NAME = "your\_bucket"

FILE\_NAME = "exported\_data.csv"

REMOTE\_FILE\_PATH = f"{BUCKET\_NAME}/{FILE\_NAME}"

# Cloud SQL parameters

INSTANCE\_NAME = "your\_instance\_name"

DB\_NAME = "your\_database\_name"

TABLE\_NAME = "your\_table\_name"

# Cloud SQL connection

DB\_USER = "your\_db\_user"

DB\_PASSWORD = "your\_db\_password"

def export\_data\_to\_gcs():

# Connect to the Cloud SQL database

cnx = mysql.connector.connect(user=DB\_USER, password=DB\_PASSWORD,

unix\_socket=f'/cloudsql/{INSTANCE\_NAME}',

database=DB\_NAME)

cursor = cnx.cursor()

# Execute the query to export data

query = f"SELECT \* FROM {TABLE\_NAME}"

cursor.execute(query)

# Export data to CSV

with open("/tmp/exported\_data.csv", "w") as csv\_file:

for row in cursor.fetchall():

csv\_file.write(",".join([str(x) for x in row]) + "\n")

# Upload the CSV file to GCS

storage\_client = storage.Client()

bucket = storage\_client.bucket(BUCKET\_NAME)

blob = bucket.blob(FILE\_NAME)

blob.upload\_from\_filename("/tmp/exported\_data.csv")

print(f"Data exported to gs://{BUCKET\_NAME}/{FILE\_NAME}")

cursor.close()

cnx.close()

if \_\_name\_\_ == "\_\_main\_\_":

export\_data\_to\_gcs()

```

### Run the Script:

Run this Python script to export the data to the specified GCS bucket.

```bash

python export\_data\_to\_gcs.py

```

This script will export the data from the Cloud SQL table to a CSV file in the specified GCS bucket.

### Conclusion:

By following these steps, you can have a detailed understanding and practical experience with Cloud SQL, including tables, views, replication concepts, and data loading from and exporting to Google Cloud Storage (GCS) buckets.

Exporting data from Google Cloud SQL to a Google Cloud Storage (GCS) bucket using the command-line interface (CLI) involves several steps. Here's how to do it:

### Step 1: Enable the Cloud SQL Admin API:

Make sure the Cloud SQL Admin API is enabled. You can enable it using the following gcloud command:

```bash

**gcloud services enable sqladmin.googleapis.com**

```

### Step 2: Export the Data to a GCS Bucket:

Use the `gcloud sql export csv` command to export the data from the Cloud SQL instance to a CSV file in the GCS bucket.

```bash

**gcloud sql export csv [INSTANCE\_NAME] gs://[BUCKET\_NAME]/[FILE\_NAME]**

**--query="[QUERY]"**

```

Replace the placeholders:

- `[INSTANCE\_NAME]` with your Cloud SQL instance name.

- `[BUCKET\_NAME]` with the name of the GCS bucket.

- `[FILE\_NAME]` with the name of the file to export to.

- `[QUERY]` with the SQL query to export the desired data.

Example:

```bash

**gcloud sql export csv my-instance gs://my-bucket/exported\_data.csv**

**--query="SELECT \* FROM employees"**

```

### Conclusion:

By following these steps, you can export data from Google Cloud SQL to a Google Cloud Storage (GCS) bucket using the command-line interface (CLI).

In Cloud SQL, read replicas allow you to replicate data from a master Cloud SQL instance to one or more read-only replicas. Here's an explanation of the different types:

1. \*\*Read Replica\*\*:

   - A read replica is a copy of the master instance that reflects changes to the data in the master.

   - Read replicas are within the same region as the master instance.

   - These replicas are used to distribute the load of read-heavy workloads and can also be used for disaster recovery.

   - They cannot be promoted to a standalone instance.

2. \*\*Cross-Region Read Replica\*\*:

   - Cross-region read replicas are read replicas that are in a different region from the master instance.

   - These are useful for disaster recovery and serving read requests from geographically distributed users.

   - You can create cross-region read replicas for regional instances only.

3. \*\*External Read Replica\*\*:

   - External read replicas are read replicas created from an external primary database server.

   - You can use external read replicas to offload read queries from your primary database, regardless of where the primary database is located.

   - They are in different Cloud SQL instances, which can even be from different projects.

Here are examples of each type of Cloud SQL read replica:

1. \*\*Read Replica\*\*:

   - Suppose you have a master Cloud SQL instance (`my-instance`) in the `us-central1` region. You can create a read replica within the same region using the following command:

     ```bash

**gcloud sql instances create my-replica --master-instance-name=my-instance --region=us-central1**

     ```

   - This will create a read replica named `my-replica` in the `us-central1` region.

2. \*\*Cross-Region Read Replica\*\*:

   - Continuing with the previous example, let's create a cross-region read replica in the `europe-west1` region:

     ```bash

**gcloud sql instances create my-cross-region-replica**

**--master-instance-name=my-instance**

**--region=europe-west1**

     ```

   - This will create a cross-region read replica named `my-cross-region-replica` in the `europe-west1` region.

3. \*\*External Read Replica\*\*:

   - In this scenario, we assume you have a MySQL instance running on a non-Google Cloud Platform (GCP) server.

   - To create an external read replica, you need to first configure the external primary database for replication and then create an external read replica in Cloud SQL.

   - Here's an example:

     - Configure the external primary database for replication (MySQL example):

       ```sql

       CREATE USER 'replica'@'%' IDENTIFIED BY 'password';

       GRANT REPLICATION SLAVE ON \*.\* TO 'replica'@'%';

       FLUSH PRIVILEGES;

       ```

     - Then, create an external read replica in Cloud SQL:

       ```bash

       gcloud sql instances create my-external-replica --database-version=MYSQL\_5\_7 --master-instance-name=my-external-master \

       --master-instance-region=us-central1 --region=us-central1

       ```

   - This will create an external read replica named `my-external-replica` in the `us-central1` region.