1.- SQS->Lambda->RDS

**Use Case - Log processing**

Millions of logs are generated and putting SQS before lambda provides us to manage number of database queries in t time according to our needs. It is a "queue" and we are consuming that queue. In some business cases it may not be useful (banking transactions etc.) but in some cases (analytic calculations) it may be helpful. Instead of making a single insert whenever lambda is invoked, we can set batch size and insert in [batch](https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html)(10 records at once) which reduces the number of queries.

Amazon SQS offload queue management tasks related to the messages and AWS Lambda performs message transformation, queues the message, and transmits the message with massive scale, fault-tolerance, and efficient message distribution to the RDS.

Lambda polls the queue and invokes your Lambda function synchronously with an event that contains queue messages. Lambda reads messages in batches and invokes the function once for each batch then load the processed data to RDS for consumption to other teams.

2.- SQS->Lambda->RDS proxy->RDS

**Use Case – Improvement on Log Processing architecture**

Above architecture have limitations where the RDS system is flooded with database requests from Lambda which can be overcome by the RDS proxy.

The proxy handles all database traffic that normally flows from our Lambda functions directly to the database. Lambda functions interact with RDS Proxy instead of the database instance. It handles the connection pooling necessary for scaling many simultaneous connections created by concurrent Lambda functions. This allows Lambda applications to reuse existing connections, rather than creating new connections for every function invocation.

A Lambda function, instead of connecting directly to an RDS instance, connects to RDS Proxy, which is a managed resource sitting in between Lambda and RDS. It serves as a DB connection pool and is responsible for efficient creation and management of database connections.

RDS Proxy’s benefits are not limited to Lambda. It also helps to increase application resilience and security. In the event of database failure, Proxy can quickly switch to a new RDS instance, preserving active connections. This is a much faster and user-friendly process than the standard DNS update.

Also, with RDS Proxy, DB authorization can be done with IAM roles rather than DB credentials in the application code. This helps to reduce the risk of leaking credentials.

3.- SQS->Lambda->Kinesis->Lambda->RDS

**Use Case- Sensors in automobiles detect need for a spare part and automatically place order**

Vehicle’s sensors send data to Kinesis through a lambda function which polls the data from an SQS Queue which will refine the data after which Kinesis data streams will ingest and stores sensor data streams for processing. AWS Lambda is triggered and runs code to detect trends in sensor data, identify anomalies, and order replacement faulty parts. Meanwhile it will store the data to the RDS and maintain the order history.

4.- SQS->Lambda->Kinesis->Lambda->RDS proxy->RDS

**Use Case - Analysis of streaming social media data**

Petabytes of data are getting generated every second ex – clicks, views on the webpage and from the social media stream which get dumped every second. Leveraging SQS it will queue the messages and trigger the lambda function which in result load the streaming data to kinesis – which will ingest and stores the social media feeds for real time processing and generate hashtag data in real time after which AWS Lambda will perform post process and write to AWS RDS but to maintain the traffic will make use of RDS proxy to balance to connection pools.

RDS Proxy will handle spikes in Lambda requests and make sure that the DB instance does not get overwhelmed by them. Requests will either be sequenced in the queue, waiting for the idle connection in the pool, or rejected if they start exceeding the limit. While this may result in increased request times or even cause some requests to be rejected, it protects the DB instance from stability issues and increases overall application resiliency.