*Data Architecture for loan broker*

## Problem statement

We need to provide a data model for loan application data processing.

Design a data architecture which can support -
1. Loan offer application
2. Reporting analysis of the applied and approved loans

## Key Requirements

Design a Logical Model to -

- Store customer information like name, email and address info
- Store credit report info
- Store information about the application (amounts and dates)
- Store information about the bids (amounts, interest rates, repayment time)
- Store information about the banks

Also serve the reporting needs –

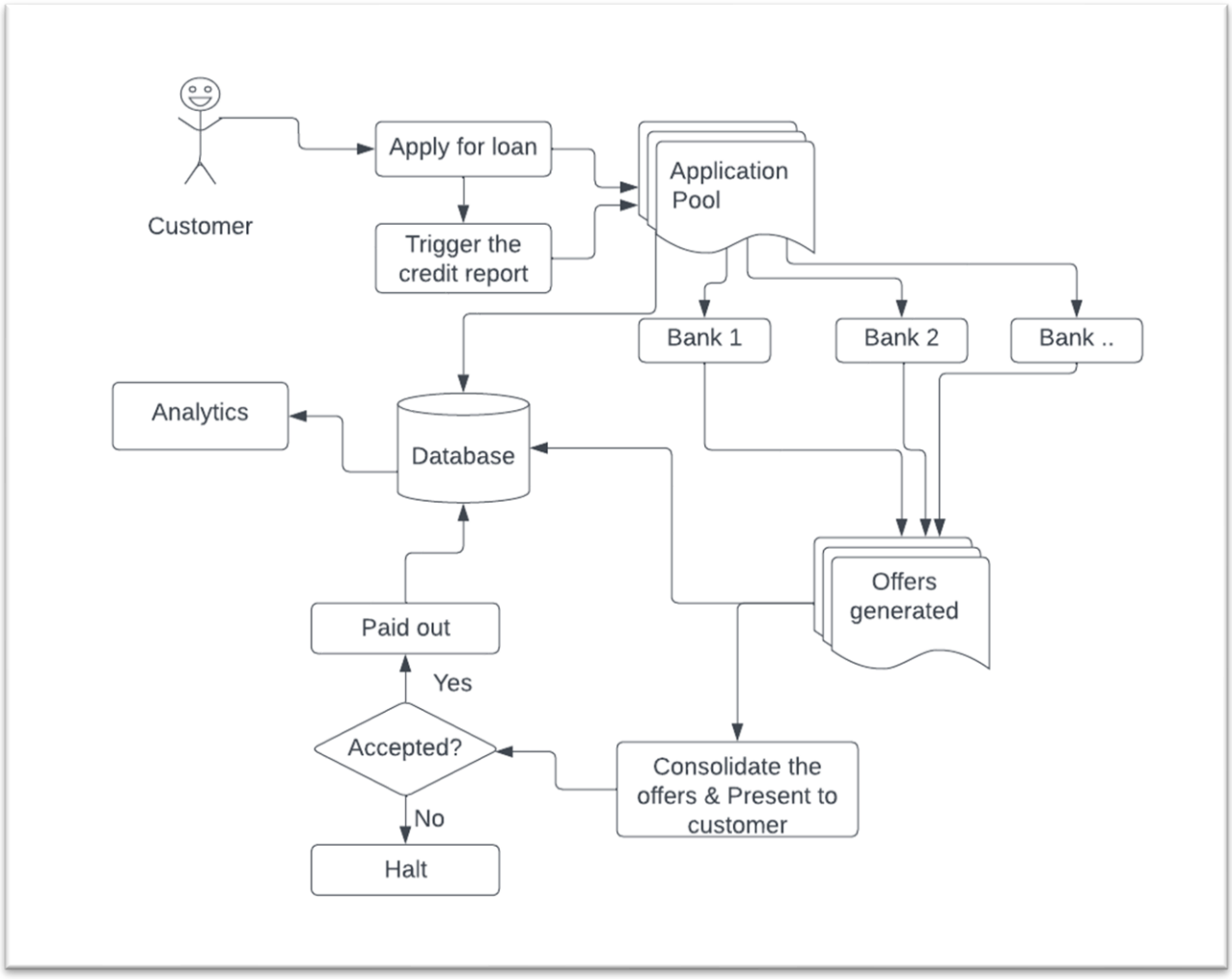
- User should be able to create the reports from the given table.

## High Level Solution flow

**Comment** - *To get requirements clarified and manage the overall understanding of the application flow, I have created the high level solution flow diagram. This is quite useful to validate my understanding with business stakeholders and can help in removing the ambiguities.*

In the below mentioned diagram the overall application flow is shown to understand the interaction of the system.

1. Customer starts the application.
2. Credit report is triggered and added with the application.
3. Banks get the notification and prepare the offers.
4. Offers are added and consolidated to the one request.
5. Customer looks on the offer bids and decide.
6. If accepted, amount paid, and added to the loan list, else halt.
7. Reporting user is able to access the database to create different reports.

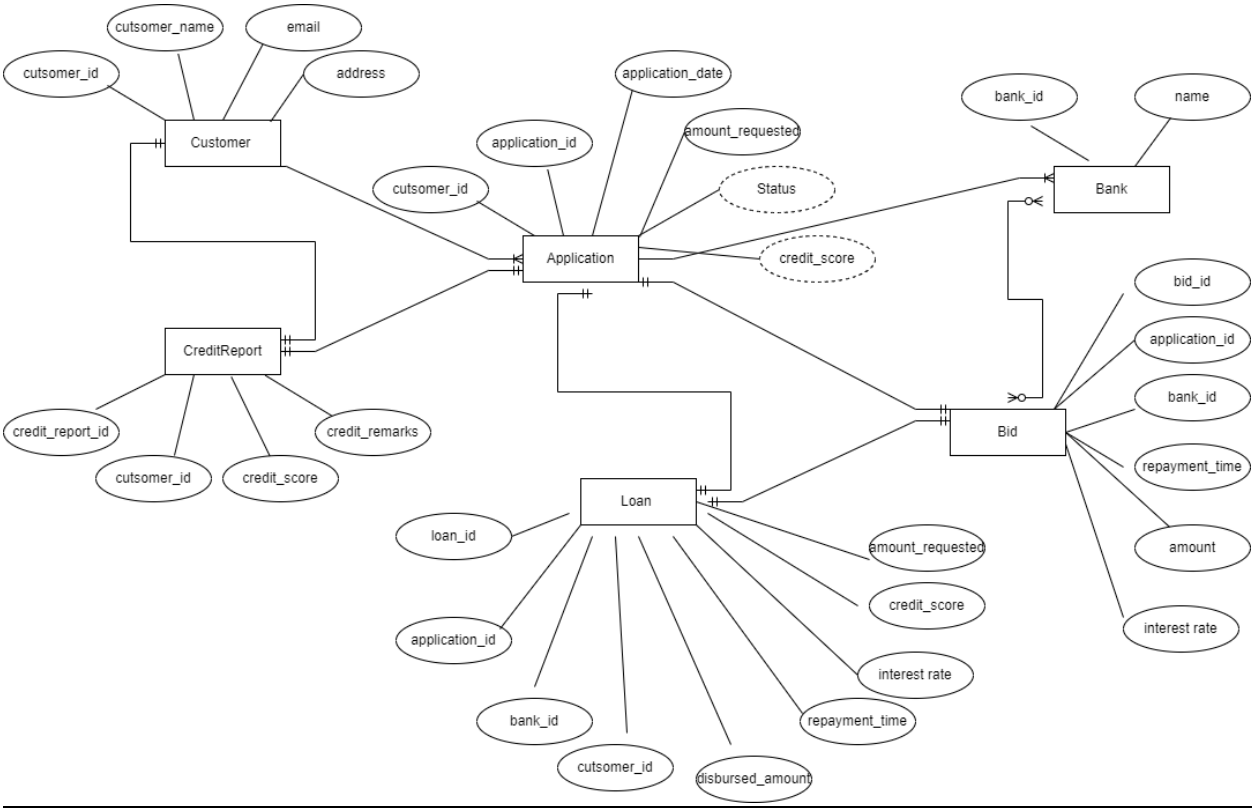



## Logical data model

**Comment**

*My recommendation would be that we design the model by considering the various entities involved in the loan application process, including customers, credit reports, loan applications, bank bids, and banks themselves.*

1. *A customer can make multiple loan applications, but a loan application belongs to only one customer (One-to-Many Relationship).*
2. *A loan application can receive multiple bids, but a bid belongs to only one loan application (One-to-Many Relationship).*
3. *All banks can make multiple bids (Many-to-Many Relationship).*
4. *Finally, we have loan ledger, one customer can have only one loan mapped from one bank [ Assuming that customer only take loan from one bank regardless of partial or full] (One-to-One Relationship)*




---

***Customer*** *entity contains all the information about customer and forms a 1 to Many relationships with application [ Assuming that one customer can apply for multiple loan applications at a time (different amount)].*

***Credit Report*** *entity contains credit report and has 1 to 1 mapping with customer and application applied [ Assuming that previous scores are not valid hence we need only one score].*

***Bank entity*** *contains the banks info and works as a bank identifier. This has 1 to many relationships with Application entity.*

***Bid entity*** *contains the bid provided by banks for application id [ Here, we could go with two solutions – a) one single record for each application and add banks and id as column,(complex to scale), b) bid wise records so multiple records for one application id, I am going with option b, adding rows of same application id if multiple bids are received, so we will duplicate application for each bid. It has Many to many (optional) relationships with the bank and 1 to 1 with Loan entity.*

***Loan entity*** *contains the final details about the loan application. It has 1 to 1 relationship with Application and Bid entity.*

*Key points for the loan entity –*

- *Loan entity can be used for the reporting and understanding about loan amount disbursed, requested amount, interest rate etc., works as a single source of truth.*
- *If customer rejected the bid, we need to fill disbursed amount with zero.*

***Why this model?***

*My goal is to minimize the operations for reporting user so that efficiency of the model is maintained, so final entity **Loan** can serve majority of the reporting needs and gives a 360-degree view of the customer. Now, Loan table can be fed to any analytics tool such as PowerBI and you will be able to get the overall status of the system.*

- *I propose to use RDMBS such as Oracle DB, MySQL or PostgresDB for this case as these DBs are very SQL friendly and serves the purpose of the use case.*
- *We can use PowerBI or Tableau or any other visualization tool to build the analytics capabilities*

***GCP based recommendations.***

- *If you want to use GCP, you can BigQuery (as a managed service from GCP, it will be quite cost effective and helps you get connected with analytics services)*
- *We should go with batch data ingestion, save some cost.*
- *If you want to build any sort of monitoring & alerting, we can use Stackdriver from GCP as it is quite compatible with all the GCP managed services.*
- *If you need to take this data out for something else, we can use RESTAPI services or provide access to the data storage.*

***Key points to remember.***

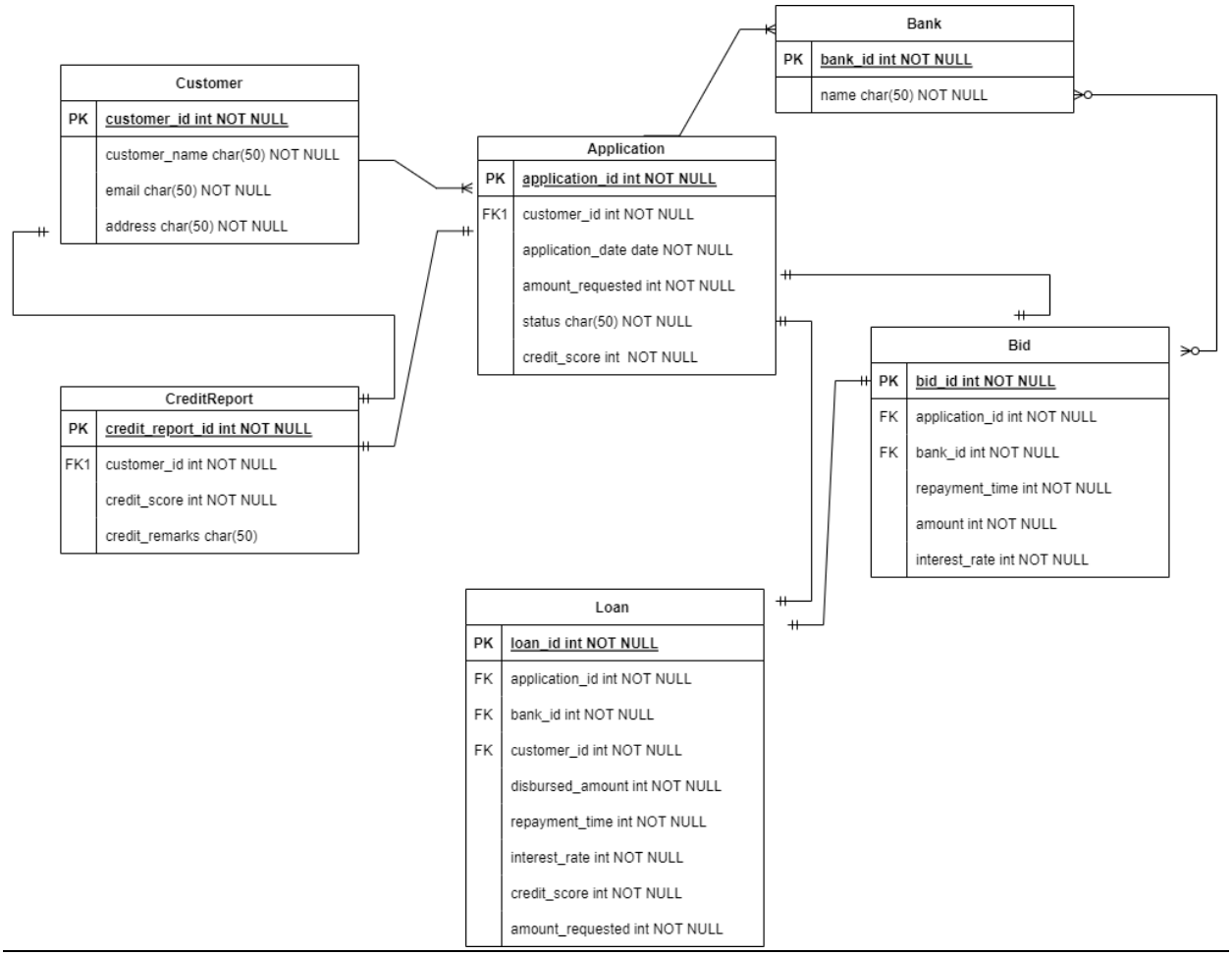
- *This data model is flexible enough to accommodate any future needs. Example if you add another customer attributes in future such as likelihood of paying capabilities, you can just add another attribute to the customer table and it will flow to the application further. So, we should **always design for future flexibility.***
- ***Efficient** – the majority of the operations are done at the end (lazy) so that we do what we need, and considering the use case, Loan entity should be able to handle majority of the needs without additional operations*

***Next steps***

*We can add timestamp to the data model so that **archiving become easy in future, and we can build data policies on this attribute** – ex. Application with zero disbursed amount, older than 365 days are archived. We will save some money on cloud resources if using cloud.*

***Design physical data model***

An overview how it will look like if we have to implement it




**- A description of the dataflow design proposal and tools selection rationale**

To support the dataflow requirements, I propose the use of an Extract, Transform, Load (ETL) system that will extract data from the two different databases, transform the data into a consistent format, and load the data into the database.

I recommend using.

- Data connector to transfer the data from other data sources -, sometimes we don't have data connector readily available so we need to create them. RESTAPI based connectors are well equipped to do these jobs.
- DataFlow from GCP to handle the ETL tasks,
- BigQuery and BigTable for the 360-view storage

**A short description of how this model could fit into a data warehouse landscape with multiple sources and users of data.**

The logical data model can fit into a data warehouse landscape with multiple sources and users of data. We need to build connectors if not readily available and then we can transfer the data to the **staging area based on Cloud storage** in GCP. We need to do data checks, validation, processing etc. before moving it to final data storage/data

warehouse, in this case Big Query. The data warehouse can be designed to support analytical queries that enable users to gain insights into loan application trends, bank bidding behavior, and customer creditworthiness.

Ex. – you have a dataset coming from Salesforce, so, we need to build a RESTAPI to connect to GCP and once connected, we can move the data (batch movement is recommended to save cost) to staging area where we perform all the data quality and processing activities before moving it to data warehouse. Similarly for other databases.

Usually, the challenge comes when we have different data format, so staging area would help us to convert it to desired data format.

**A description of important aspects you see in relation to data governance and security.**

I propose the data monitoring and alerting services for the governance and security. For the data governance, I recommend implementing a data quality program that includes data profiling, validation, cleansing, and enrichment. This project should be designed to ensure that the data is accurate, complete, and consistent. This could be achieved using Stackdriver and DataFlow in GCP.

I recommend that we follow the least privileged access methods and have group access policies to ensure that right audience have access to only needed data. Hence security measures are in place.