Kiva Crowdfunding Dataset Analysis

Poojitha Katta
Department of Applied Data Science
San Jose State University
San Jose, United States
poojitha.katta@sjsu.edu

Purnima Bhukya
Department of Applied Data Science
San Jose State University
San Jose, United States
purnima.bhukya@sjsu.edu

Deepali Zutshi
Department of Applied Data Science
San Jose State University
San Jose, United States
deepali.zutshi@sjsu.edu

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjsu.edu

Abstract—The main aim of this project is to design a database for the crowdfunding platform-Kiva, suggest the best solution based on the trade-offs between SQL and NoSQL schemas for Kiva as a crowdfunding database. The final solution would be decided based on the performance of each approach and the requirements of the project.

Keywords—SQL, NoSQL, Crowdfunding, Database, Kiva

I. INTRODUCTION

Crowdfunding is a system that enables individuals or ventures to seek small investments, contributions, or loans from a variety of funders online. Kiva is a crowdfunding platform that offers a new financing channel for small and micro-businesses as well as individuals. The aim of the project is to offer the best solution as a database management system for Kiva platform to analyze the factors that influence crowdfunded projects by estimating the welfare level of partners in specific regions, based on shared financial and demographic aspects. In this study, the comparison between MySQL as a Relational Database Management System (RDBMS) model and MongoDB as NoSQL approach was investigated and various conclusions were drawn.

II. DATASET

A. Source

The data was obtained from Kaggle, an online community of data scientists and machine learning practitioners. It was made available by Kiva, an online crowdfunding platform, for the "Data Science for Good" Challenge and invited people to help them build a localized model to estimate various metrics in regions where Kiva has active loans.

B. Dataset Description

The dataset contains 4 csv files with 54 attributes total, which includes 30 string, 7 decimal, 4 datetime, and 13 other data types. The first table consists of 20 columns, detailing the id, funded amount, loan amount, country code, country, currency, region, etc. Similarly, the second, third and fourth table outlines data snapshots and can be matched to the loan theme regions to get a loan's location and provides details for id, loan theme id, loan theme type, partner id, and MPI (Multidimensional Poverty Index). Extracting several insights from the historical micro-loans over a period of time and correlating the regional averages by gender, sector, or borrowing behavior to estimate the welfare rate is to be followed.

C. Data Cleaning

The data needed to be cleaned and corrected before it could be processed and stored in the database. Many of the tables had missing data which was replaced by the mode of the data in that column. Attributes with most of the data missing were removed from the tables altogether. Many of the tables also contained attributes that were duplicated in the same table as well as across multiple tables. These were removed and the data was pre-processed to remove redundancies.

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [ ]: df = pd.read_csv('kiva_loans.csv')

In [ ]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [ ]: df.info()

In [ ]: df.describe()

In [ ]: null  = df.isnull().sum().sort_values(ascending = False).reset_index()
        null.columns = ['Column', 'Frequency']
        null

In [ ]: ## tags column consists of 10000 null values so remove the colums.
        df.drop('tags', axis=1,inplace=True)

In [ ]: ## this column null values are replaced with the mode
        df['funded_time'].mode()

In [ ]: df['borrower_genders'].mode()

In [ ]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [ ]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [ ]: df.dropna(inplace = True)

In [ ]: df.isna().sum()

In [ ]: data.to_csv('kiva_mpi_region_locations', encoding='utf-8', index = False)
```

Fig. 1 Jupyter Notebook for Data Cleaning

III. DATA MODEL

A NoSQL database system such as MongoDB performs differently compared to a traditional relational database system. Where an RDBMS is incapable of handling and storing unstructured and semi-structured data, NoSQL can store, process, and visualize such data with ease. MongoDB represents the information as a JSON document instead of the row and column format adopted by an RDBMS system. MongoDB stores data in structures called 'collections' and the data entries are called 'documents'. The documents contain key-value pairs of various types and can also consist of arrays, nested documents,

etc. Each document can have its own structure and are self-describing in nature.

A. Data Model in MongoDB

The following diagram represents the various collections created by importing the data into MongoDB. Each attribute in the collections represents the various documents keys and is of variable types (string, integer, date, timestamp, etc.).

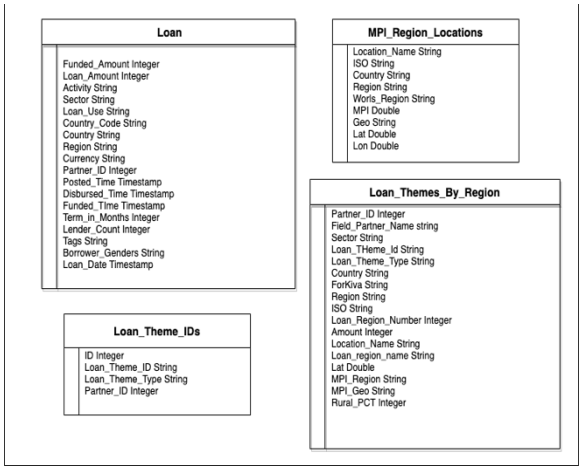


Fig. 2 Data Model - NoSQL

Each collection in the database contains documents with a JSON-like structure having key-value pairs for each of the entries in the collection. A total of four collections were created initially and imported into MongoDB for this project. These were then aggregated to create a single collection named 'all_kiva'.

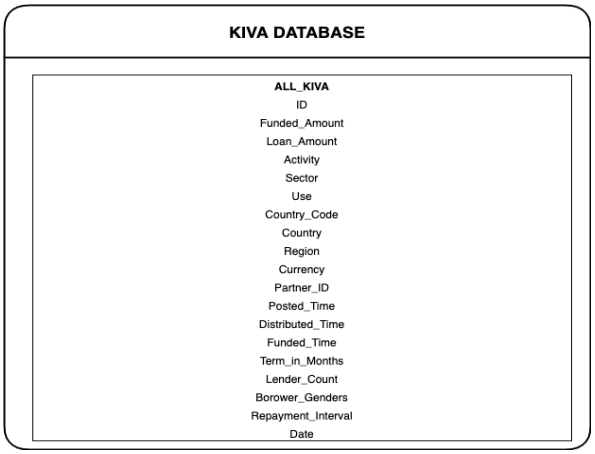


Fig. 3 Document Structure in MongoDB

B. Data Model in MySQL

An RDBMS model such as MySQL represents the data as a collection of relations. Each relation is represented by a table structure with rows and columns of records. In a relation thought of as a table, every row indicates a collection of data values related to each other. The name of the table and the columns help users interpret the meaning of the values stored in the rows.

Data in a relational model is normalized, which is the process of structuring the data in order to reduce redundancy as well as improve the integrity of the database.

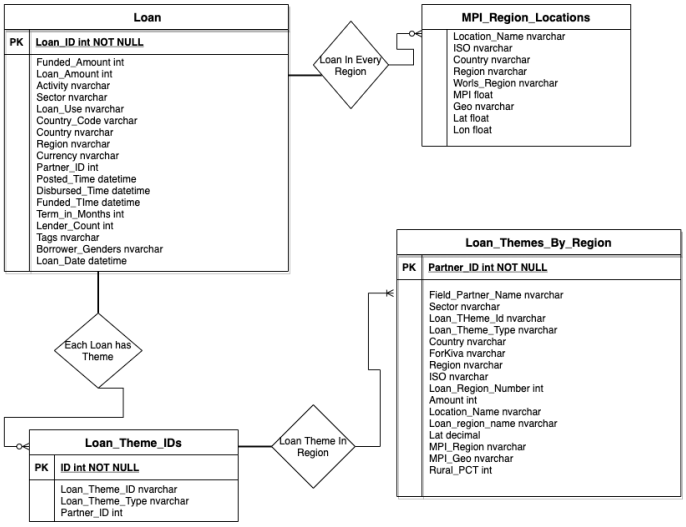


Fig. 4 Denormalized Data in SQL

The figure above represents the raw data available to us in a denormalized manner. It can be seen that various columns have been repeated in several tables which introduces redundancy in the database.

Normalization of this data generated 7 tables from the existing 4 and improved the overall quality of the data. For example, attributes such as 'ISO ', 'country', 'latitude', 'longitude', 'world_region', 'location' were common in many of the entities. These were extracted to create a separate entity table called 'Region' which was then linked to the table 'Loan' to define the loan details for each region. The following schema was created after the process of normalization was performed.

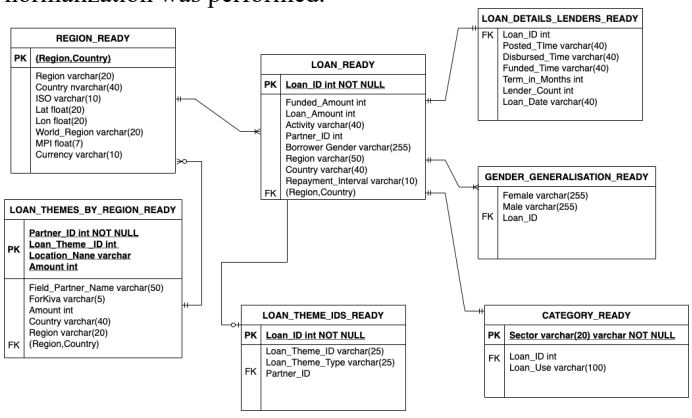


Fig. 5 Normalized Data Model in SQL

IV. QUERIES AND PERFORMANCE

In this section all the operations include create, read, update and insert (CRUD) were performed on both a SQL-based database and a NoSQL database and thoroughly investigated. Each operation has 3 related figures in this section. The first figure for each operation shows the SQL query and the time taken when the query or operation command is performed on MySQL workbench. In the same figure, below the results of the query, the execution time is visible. The second figure for the related operation displays the command on MongoDB and the third figure shows the statistics of that operation on MongoDB. A table that shows the command and its performance is placed right after the figures for each operation separately.

A. Read Operations

In this section, 7 read operations have been performed on both RDBMS (MySQL) and NoSQL (MongoDB). The queries and the execution time for operations have been monitored and displayed in the table below screenshots of the queries and their performances.

a) Read operation - 1

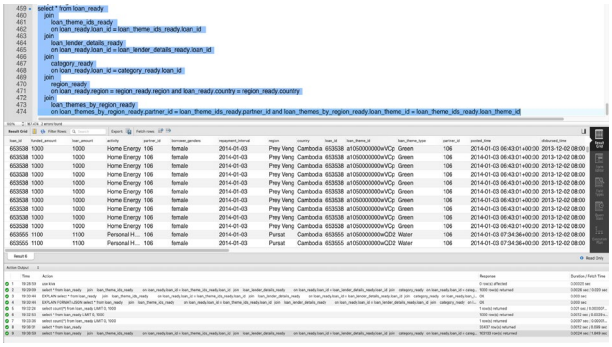


Fig. 6.a. Retrieve all data from joining all tables in MySQL

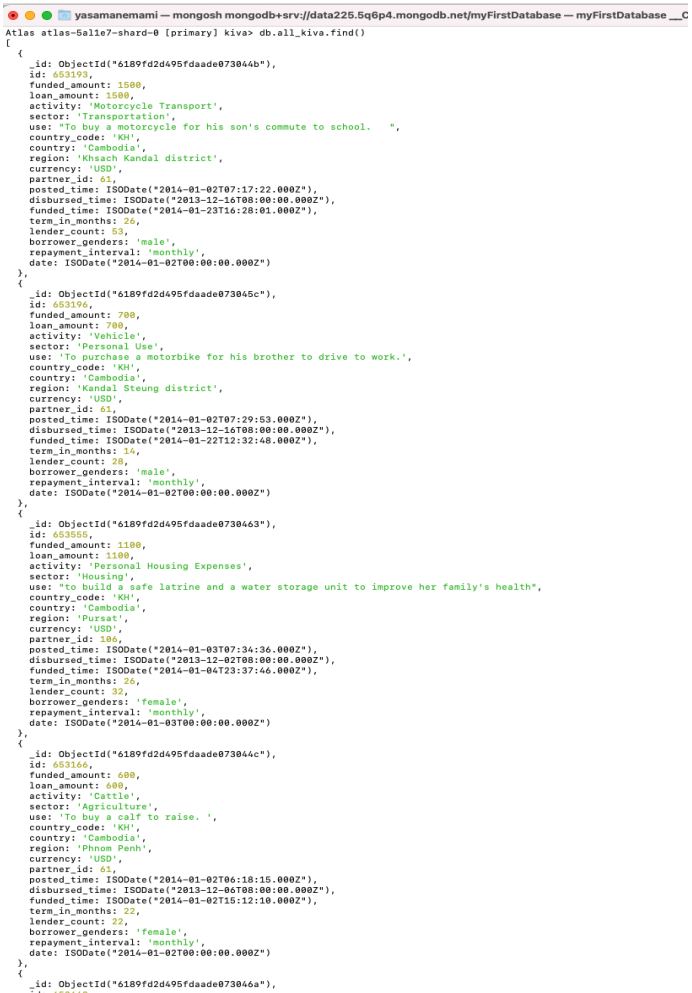


Fig. 6.b. Retrieve all the data in all_kiva collection on MongoDB

Fig. 6.c. Performance of retrieving all data on MongoDB

Read - Operation 1		
DB	Command	Execution Time(ms)
SQL	select * from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id join category_ready on loan_ready.loan_id = category_ready.loan_id join region_ready on loan_ready.region = region_ready.region and loan_ready.country = region_ready.country join loan_themes_by_region_ready on loan_themes_by_region_ready.partner_id = loan_theme_ids_ready.partner_id and loan_themes_by_region_ready.loan_theme_id = loan_theme_ids_ready.loan_theme_id;	1837
NoSQL	db.all_kiva.find()	105

Table 1. Performance and Commands for the first read operation

Fig. 7.b loan_amount between 500$,1500$ on MongoDB

b) Read operation - 2

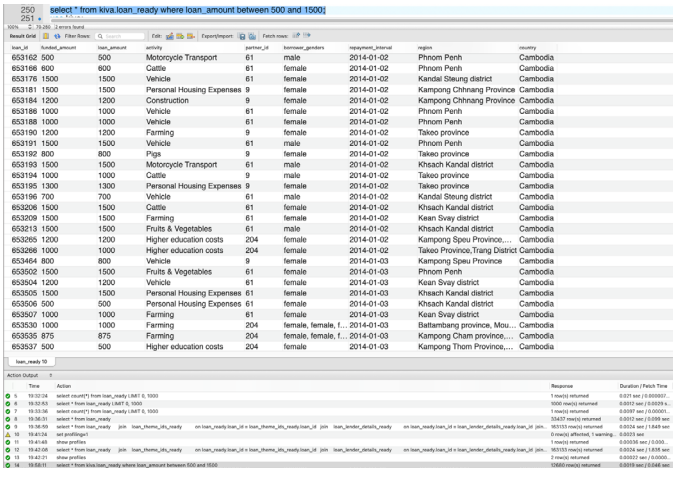


Fig. 7.a loan_amount between 500$,1500$ on MySQL

Fig. 7.c Performance of loan_amount between 500$,1500$ on MongoDB

Read - Operation 2		
DB	Command	Execution Time(ms)
SQL	select * from kiva.loan_ready where loan_amount between 500 and 1500;	47
NoSQL	db.all_kiva.find({loan_amount:{$gt:500, $lt:1500}})	317

Table 2. Performance and Commands for the first read operation

c) Read operation - 3

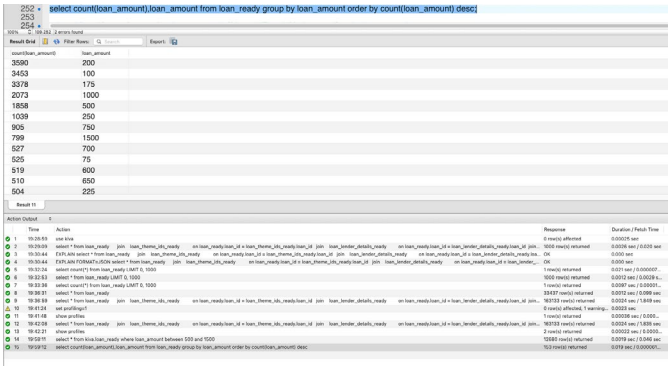


Fig. 8.a. Count of each loan

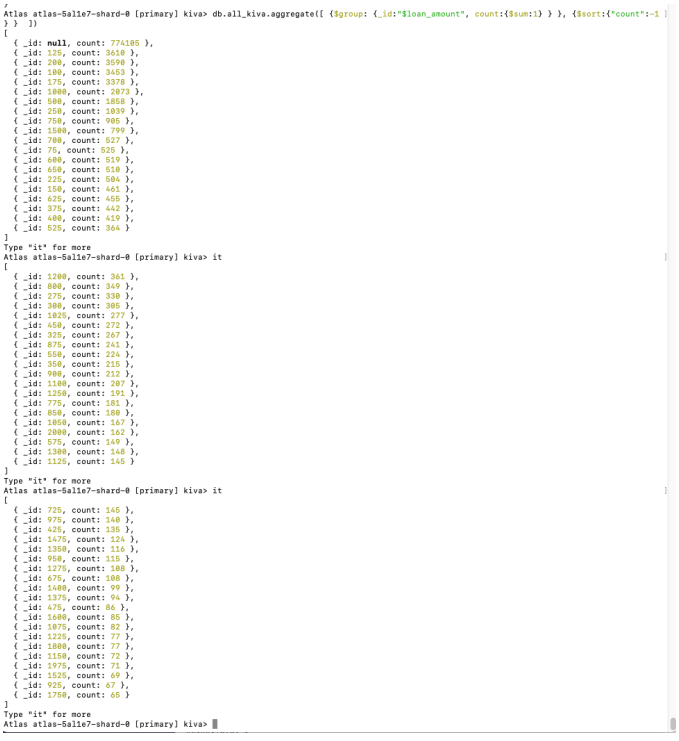


Fig. 8.b Count of Each Loan

Fig. 8.c Count of Each Loan NoSQL Performance

Read - Operation 3		
DB	Command	Execution Time(ms)
SQL	select count(loan_amount),loan_amount from loan_ready group by loan_amount order by count(loan_amount) desc;	18
NoSQL	db.all_kiva.aggregate([{ $group: { _id:"$loan_amount", count:{$sum:1}}}, {$sort:{"count":-1}}])	133

Table 3. Performance and Commands for the second read operation

a) Read operation - 4

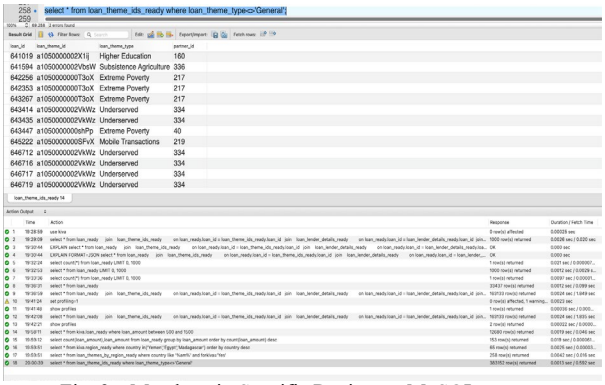


Fig. 9.a Members in Specific Region on MySQL

Fig. 9.b Members in Specific Region on MongoDB

Fig. 9.c Performance of Members in Specific Region in MongoDB

Read - Operation 4		
DB	Command	Execution Time(ms)
SQL	select * from kiva.region_ready where country in('Yemen','Egypt','Madagascar') order by country desc;	1.3
NoSQL	db.all_kiva.find({ country:{ $in:["Yeman", "Egypt", "Madagascar"]}}).sort({country:-1})	54

Table 4. Performance and Commands for the third read operation

b) Read operation - 5

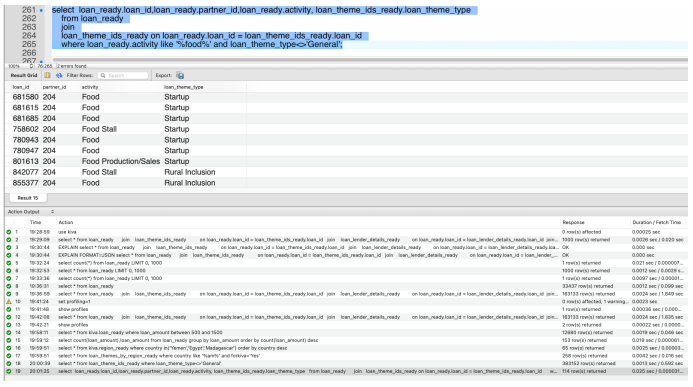


Fig. 10.a Countries Like '_am%' on MySQL

Fig. 10.b Countries Like '/.*am.*/'

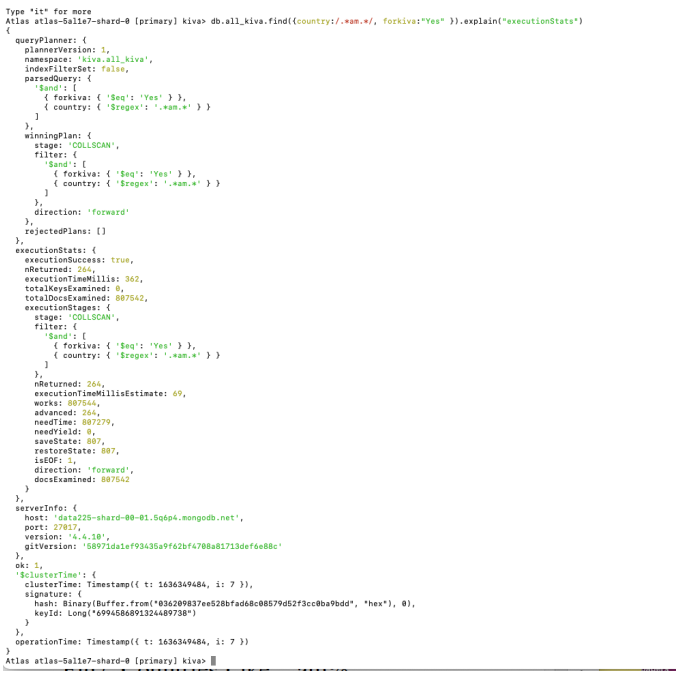
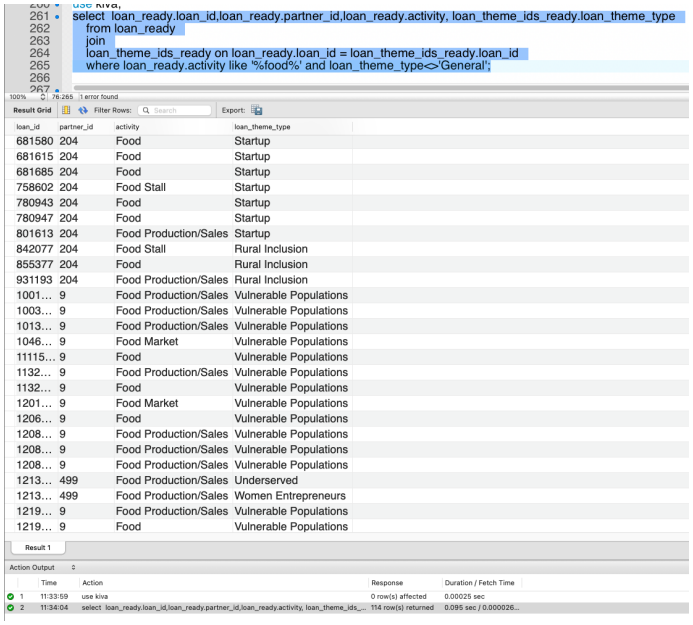


Fig. 10.c . Countries Like '/.*am.*/' Performance on MongoDB

Read - Operation 5		
DB	Command	Execution Time(ms)
SQL	select * from loan_themes_by_region_ready where country like '%am%' and forkiva='Yes';	20
NoSQL	db.all_kiva.find({country:/.*am.*/, forkiva:"Yes" })	362

Table 5. Performance and Commands for the fourth read operation

e) Read Operation - 6



loan_id	partner_id	activity	loan_theme_type
681580	204	Food	Startup
681615	204	Food	Startup
681685	204	Food	Startup
758602	204	Food Stall	Startup
780943	204	Food	Startup
780947	204	Food	Startup
801613	204	Food Production/Sales	Startup
842077	204	Food Stall	Rural Inclusion
855377	204	Food	Rural Inclusion
931193	204	Food Production/Sales	Rural Inclusion
1001…	9	Food Production/Sales	Vulnerable Populations
1003…	9	Food Production/Sales	Vulnerable Populations
1013…	9	Food Production/Sales	Vulnerable Populations
1046…	9	Food Market	Vulnerable Populations
11115…	9	Food	Vulnerable Populations
1132…	9	Food Production/Sales	Vulnerable Populations
1132…	9	Food	Vulnerable Populations
1201…	9	Food Market	Vulnerable Populations
1206…	9	Food	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1213…	499	Food Production/Sales	Underserved
1213…	499	Food Production/Sales	Women Entrepreneurs
1219…	9	Food Production/Sales	Vulnerable Populations
1219…	9	Food	Vulnerable Populations

Fig.11.a Records from join of two tables to show the loans in food industry specific type (exclude general types) on MySQL

Fig. 11.b Show the loans in food industry-specific type (exclude general types)

Fig.11.c Performance the loans in food industry-specific type (exclude general types) on MongoDB

Read - Operation 6		
DB	Command	Execution Time(ms)
SQL	select loan_ready.loan_id,loan_ready.partner_id, loan_ready.activity, loan_theme_ids_ready.loan_theme_type from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id where loan_ready.activity like '%food%' and loan_theme_type<>'General';	24
NoSQL	db.all_kiva.find({activity:/.*Food.*/, loan_theme_type:{$ne:"General"} })	305

Table 6. Performance and Commands for the fifth read operation

f) Read Operation - 7

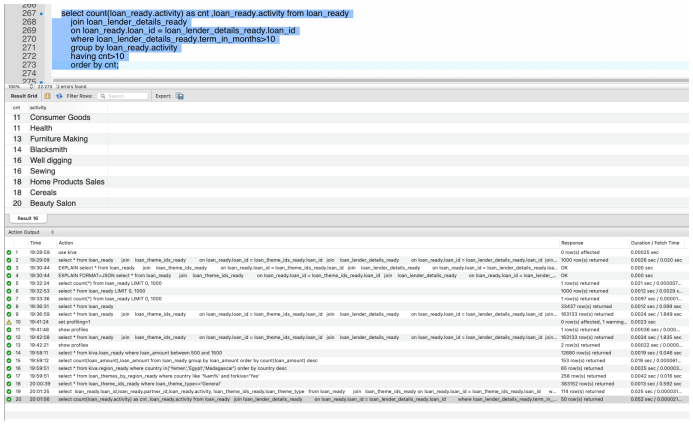


Fig.12.a # of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

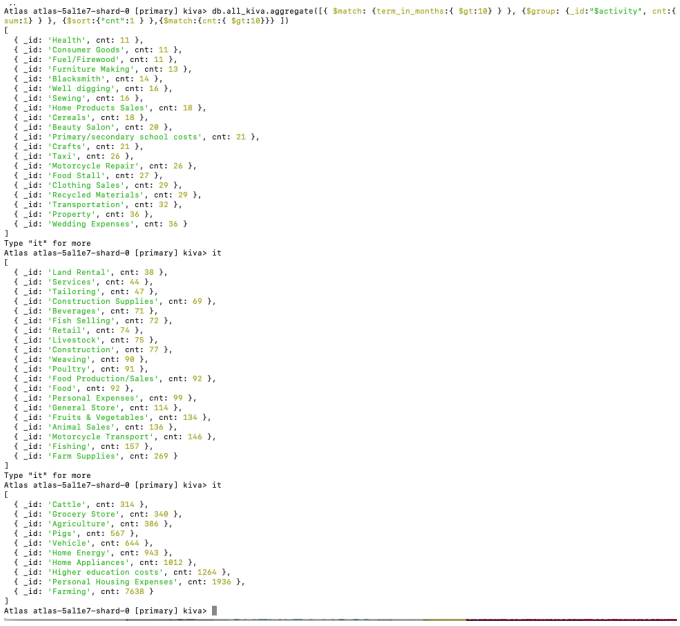


Fig. 12.b Display number of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

Fig. 12.c Performance of # of times each activity gets funded with terms of bigger10 months and if they are funded more than 10 times

Read - Operation 7		
DB	Command	Execution Time(millisec)
SQL	select count(loan_ready.activity) as cnt ,loan_ready.activity from loan_ready join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id where loan_lender_details_ready.term_in_months >10 group by loan_ready.activity having cnt>10 order by cnt;	52
NoSQL	db.all_kiva.aggregate([{ $match: { term_in_months:{ $gt:10 } } }, {$group: { _id: "$activity", cnt: {$sum:1} } }, { $sort:{ "cnt":1 } },{ $match:{cnt:{ $gt:10}}}])	305

Table 7. Performance and Commands for the sixth read operation

From all of the above experiments, the read execution time for MySQL is faster than MongoDB if we are using where clause and filtering the data. However, retrieving all the data is faster in MongoDB almost 18 times compared with MySQL.

B. *Update Operation*

In this section, update operation on MySQL and MongoDB is investigated.

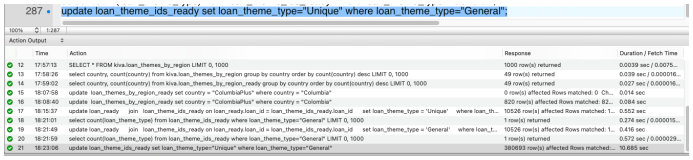


Fig. 13.a Update on Mysql

Fig. 13.b Update documents set a value on condition

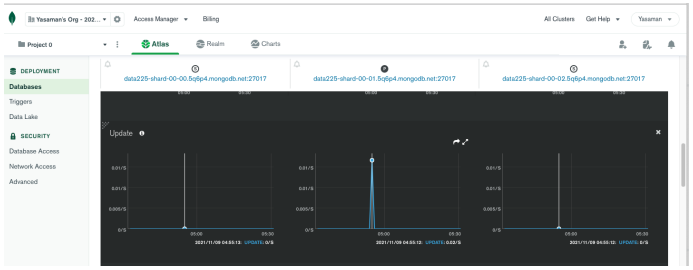


Fig. 13.c Update Performance on MongoDB

Update - Operation		
DB	Command	Execution Time(ms)
SQL	update loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id set loan_theme_type = 'General' where loan_theme_type = 'Unique';	10685
NoSQL	db.all_kiva.updateMany({loan_theme_type: "General"}, {$set:{loan_theme_type: "Unique" }})	20

Table 8. Performance and Commands for update operation

The results show that for Kiva database **update** operation on MongoDB is faster than MySQL.

i)C. Insert Operations

This section focuses on insert operation and performances.

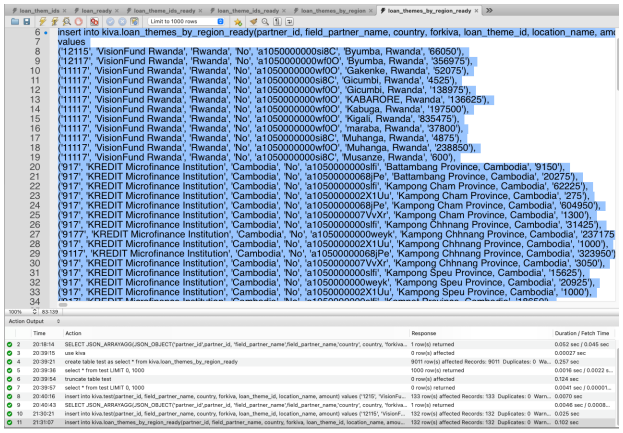


Fig. 14.a Insert 133 records to MySQL

Fig. 14.b Insert 133 documents to all_kiva collection

Fig. 14.c Performance of Insert on MongoDB

Insert - Operation		
DB	Command	Execution Time(ms)
SQL	insert into loan_ready(loan_id, funded_amount, loan_amount, activity, partner_id, borrower_genders, repayment_interval, region, country) values (1653162, 50, 500, 'Motorcycle Transport', 61, 'male', '2014-01-02', 'Phnom Penh', 'Cambodia'), 132 other records	102
NoSQL	db.all_kiva.insertMany([{ "amount": 66050, "country": "Rwanda", "forkiva": "No", "partner_id": 1215, "loan_theme_id": "a1050000000si8C", "location_name": "Byumba, Rwanda", "field_partner_name": "VisionFund Rwanda" }, 132 more records])	20

Table 9. Performance and Commands for insert operation

Results of insert operation show that Kiva database insert operation on MongoDB is faster than MySQL.

ii)D. Delete Operations

This section shows experiments on delete operation on MySQL and MongoDB.

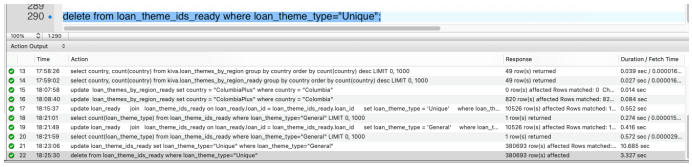


Fig. 15.a Delete records based on condition on MySQL

Fig. 15.b Delete documents based on conditions on MongoDB

Fig. 15.c Performance of delete documents on MongoDB

Delete - Operation		
DB	Command	Execution Time(ms)
SQL	delete from loan_theme_ids_ready where loan_theme_type="Unique";	3300
NoSQL	db.all_kiva.deleteMany({loan_theme_type:"Unique"})	3

Table 10. Performance and Commands for delete operation

The results of this section show that the delete operation on MongoDB is faster than MySQL.

V. DESIGN CHOICE

The choice of database design is a critical decision and there are important factors such as data size, relationships, structure, and the necessity to enforce a schema and ensure consistency that needs to be addressed. Both the relational database model and the NoSQL database model are appropriate for certain applications. Depending on the problem that the organization is attempting to solve, it will determine whether a NoSQL database model or a relational database model should be used. In addition, some organizations may opt for a hybrid mix of NoSQL and relational databases.

Developers are particularly attracted to the document data model. Objects must be transformed back and forth between the database and objects in the programming language in the relational model used by RDBMS. This can slow down development and add complexity to the program. In a document database, the document can almost directly map to the class structure of the programming language, saving the programmer time[7].

As scale takes precedence, it becomes increasingly expensive to maintain and manage a relational database system and we start looking at non-relational models that don't require a strong schema. Also, the aim of the Kiva platform is to gather the data and perform data analytics on the data to extract information that enhances businesses.

The most frequent operations of this project are inserting new records of data. Update and delete records also might happen from time to time and the data analysis would be required to derive decisions. For this project, after creating, storing, querying both the relational and non-relational models and comparing each of their performances against each other, we conclude that a non-relational schema is the one that works best.

A. Data Updation

A NoSQL database is extremely flexible and allows changes to be made to the database easily as required. With our project, updating records regarding new projects is an operation that is prioritized over others and a comparison of the performance of these operations with MySQL reveals that they are exponentially slower than NoSQL. For example, an update operation performed in MySQL took 10685 ms whereas the same operation, when performed in NoSQL, took only 20 ms. This shows NoSQL is approximately 500x faster than MySQL.

B. Schema Enforcement

NoSQL databases tend to have an implicit schema and are not entirely schemaless. With changing requirements and new data coming frequently it becomes increasingly difficult to modify and enforce a new schema every time. It would take enormous amounts of time to keep rewriting enforcement rules and updating them daily.

C. Record Insertion

Ease of access to the internet has brought inventors and investors closer than ever before. With the means to make available their idea online and gather funding for their projects, entrepreneurs use crowdfunding requests to platforms such as Kiva that are in large demand. This means the creation and insertion of the details of these new projects in the database are much more frequent. A comparison of these operations in MySQL and NoSQL demonstrates that it is much easier and faster to do this in NoSQL with execution times being extremely lower. For instance, the insertion of a record in MongoDB took 20 ms, on the other hand, MySQL took more than 5 times that to insert the same record.

D. Data Analytics

The purpose of creating a database for Kiva Crowdfunding is to analyze this data and draw conclusions about the factors that influence crowdfunding, how financial and demographic aspects affect the creators and their general welfare. The analysis of data requires powerful visualization BI tools. With time MongoDB has built BI capabilities directly into the NoSQL database and allowed users to take a deeper dive into their data. The use of visualization tools available in MongoDB Atlas has allowed us to map the distribution of crowdfunded projects across the world and helped pose meaningful questions such as why a certain part of the world sees a greater proportion of such projects than the rest.

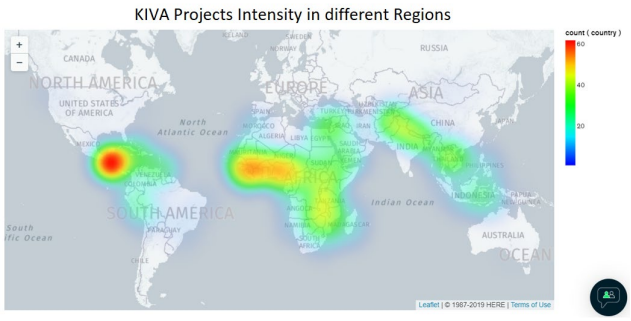


Fig. 16 Heat Map of Distribution of Crowdfunded Projects

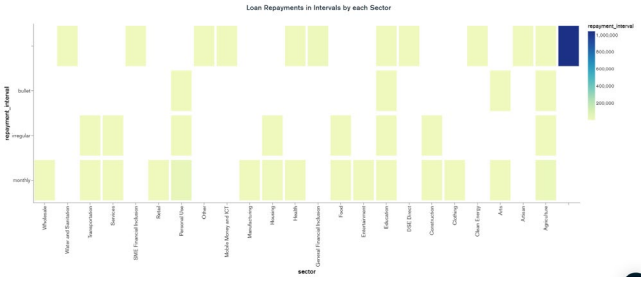


Fig. 17 Distribution of loan repayment interval over various sectors

This type of analysis serves as an answer as to which sector is gaining the most out of such crowdfunding campaigns and thus help improve upon ways in which those who can benefit from it.

Moreover, NoSQL's flexible data modeling is well suited to support dynamic scalability and improved performance for Big Data analytics[8], and it could be leveraged as a new category of data architecture coexisting with traditional SQL databases.

E. Use of Semi- and Unstructured Data

A NoSQL database such as MongoDB is very well suited to handle semi-structured and unstructured data alike. Businesses all the time are changing to best serve partners and customers. To satisfy this requirement, a schema-free database would be a significant factor to align to updated requirements. A database such as one for Kiva requires the storage of unstructured data like log files, spreadsheets, and images. The inability of a traditional relational database model to handle such data does not make it suitable for this database.

The following figure shows the creation of a non-relational database schema for Kiva and the records stored in the collections.

VI. TRADE-OFFS

The selection of either a SQL or a NoSQL database depends entirely on the type of data one has and how they aim to use it. There is no one-size-fits-all policy that applies here and both the schemas come with their own advantages and disadvantages. The selection of NoSQL for this project comes with a compromise on some of the features MySQL provides that MongoDB cannot.

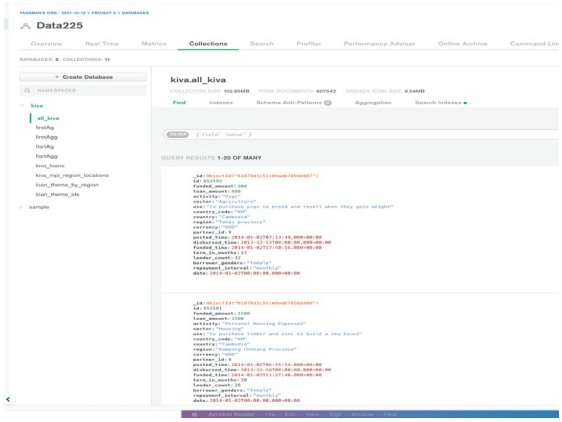


Fig. 18 Collection in MongoDb - 'all_kiva'

A. Complex Queries

Although NoSQL outperforms SQL at faster execution for queries, it is SQL that is able to execute complex queries easily and much faster than NoSQL. The availability of joins in MySQL allows data to be combined from multiple tables and compared easily which is not allowed in NoSQL.

B. Consistency

Unlike SQL databases, NoSQL databases do not follow ACID properties and hence have a latency period before consistency is achieved across all nodes in the database but it is not guaranteed when this will happen. It instead follows BASE properties which describes consistency as a developer's problem, one that should not be handled by the database.

C. An Established Platform

Relational databases are well established and mature technology. SQL standards are well defined and commonly accepted whereas NoSQL is still developing and still has a long way to go.

VII. CONCLUSION

It is not an option to use only SQL or NoSQL for all business practices. Every business has a unique requirement for how they intend to use a database, and these requirements are determined by what the business expects from its database. Both the SQL and NoSQL models have their own set of advantages and disadvantages that each business must weigh before deciding which is best for them. There is no absolute single solution for choosing a database for all the businesses. The database design should be based on the purpose of the business and the cost-performance factor. In this study, we investigated and designed a relational model as well as NoSQL document-based model for Kiva crowdfunding platform. Based on the performance of each approach and the requirement of the business for big data analytics on the data, we decided to choose NoSQL design over the relational data model for querying and storing the data and performing analytics.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Professor Simon Shim for his guidance that aided in the completion of this project and helped us understand database concepts using practical examples. We would also like to thank Shiva Abhishek Varma Penmetsa and Rushikesh Jagtap for their continued help and support throughout the development of the project.

REFERENCES

[1] S. Yu, Crowdfunding and regional entrepreneurial investment: an application of the CrowdBerkeley database, Research Policy, Volume 46, Issue 10, 2017, Pages 1723-1737, ISSN 0048-7333

[2] Burtch, G., Ghose, A., Wattal, S., 2014. Cultural Differences and geography as determinants of online prosocial lending. MIS Q. 38 (3), 773–794.

[3] Mollick, E., Nanda, R., 2015. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. Manage. Sci. 62 (6), 1533–1553.Mollick, E.R., 2014. The Dynamics of Crowdfunding: An Exploratory Study. J. Bus.Venturing 29 (January (1)), 1–16.

[4] Smith, Tim. “Crowdfunding.” Investopedia, Investopedia, 13Sept.2021,www.investopedia.com/terms/c/crowdfunding

[5] Tzoof Avny Broosh. “ How to Choose the Right Database”,Sept. 2019, https://towardsdatascience.com/how-to-choose-the-right-database-afcf95541741.

[6] Jatin Raisinghani. “Should You Use MongoDB or SQL Databases for Analytics?”, Aug. 2018, https://www.holistics.io/blog/should-you-use-mongodb-or-sql-databases-for-analytics/.

[7] Nance, Cory; Losser, Travis; Iype, Reenu; and Harmon, Gary, "NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH" (2013). SAIS 2013Proceedings. 27.

[8] Venkatraman, Sitalakshmi, et al. "SQL versus NoSQL movement with big data analytics." International Journal of Information Technology and Computer Science 8.12 (2016): 59-66.