RavenDB

ANA TERESA CRUZ, ANTÓNIO BEZERRA, and MARIANA TRUTA, Faculdade de Engenharia da Universidade do Porto, Portugal

In the present paper, the object of study was the RavenDB database. The first section describes extensively the database, presenting its history, features, use cases, and a comparison with other databases. The last section presents the developed prototype to showcase RavenDB's main features.

CCS Concepts: • **Information systems → Database design and models**.

Additional Key Words and Phrases: NoSQL, RavenDB, document database, e-commerce

ACM Reference Format:
Ana Teresa Cruz, António Bezerra, and Mariana Truta. 2022. RavenDB. 1, 1 (July 2022), 13 pages.

THE TECHNOLOGY

1 INTRODUCTION

1.1 History

In June 2008, Oren Eini initiated his work on designing the next-gen document database, being tired of the problem brought by relational databases. After about a year, coding for RavenDB [10] began. In May 2010, RavenDB 1.0 became the pioneer Document Database to offer fully transactional ACID guarantees over multiple documents and throughout the entire database. It is the first Document Database on the .NET platform. Afterward, RavenDB started securing its first clients, a small team of developers, and spreading all around the world.

RavenDB continued to be explored and, the version released in October 2016 takes database clustering to a whole new level. A multiple nodes database lets you replicate your database to several places, creating a fault-tolerant application. The version released in 2018 offered a robust set of features making it easy for a developer to integrate a document database into his application. The features developed keep on increasing and now RavenDB counts on more than 3 million developers' downloads.

1.2 Licensing model

This technology allows to experiment with a free license [13] letting a user first have an opportunity to understand whether RavenDB is adequate for their needs. A person "*can take out a free on-premise database cluster or try out a cloud instance of RavenDB Cloud*", the Managed Database service. At no cost, Raven handles "*the installation, configuration, security, patching, updates, maintaining high availability, and monitoring of the user's database so they can focus more on their application*".

Authors' address: Ana Teresa Cruz, up201806460@up.pt; António Bezerra, up201806854@up.pt; Mariana Truta, up201806543@up.pt, Faculdade de Engenharia da Universidade do Porto, Portugal.

XXXX-XXXX/2022/7-ART $15.00
https://doi.org/

1.3 Pricing model

In complement to the free licenses, **community free**, **developer** and **cloud**, there are two paid licenses: **professional**, and **enterprise** [4]. The main differences between all are the price per core, the maximum number of cores, the maximum cluster memory usage, and the nodes' number. The adequacy also changes, for example, the developer license is as described by the name only suitable for developing purposes since said license must be renewed every six months.

1.4 Community and Support

RavenDB has a large community started by them in GitHub Discussions [19]. This is an active community where people from all around the world can ask questions, give ideas, make polls and even give a show and tell and demonstrate to others their implementations with RavenDB. In stack-overflow [17], Raven is a topic present with 2,474 questions, being the most recent from five hours ago, which proves this community is also active.

Besides these communities, RavenDB also offers other forms of support [12]. There is a free option where, on the website, the user fills a form or two paid options for support, **professional**, and **production**.

Lastly, the CEO also started a blog in 2004 where he posts, among others, RavenDB-related content [2]. In those posts, people can comment, and he is prompt in responding. He even adds posts related to previously asked questions.

1.5 Documentation

RavenDB has extensive documentation [7] to help users with their implementation. The website provides the following sections: What's New, Getting Started, "Inside RavenDB' book, RavenDB Cloud Documentation, and About the Examples. On What's new, one can see what features were recently added or are being worked on. The **Getting Started** section instructs a user on the installation. **"Inside RavenDB" book** written by the CEO, Oren Eini, is designed not only to give full knowledge of what RavenDB does but also all the reasoning behind each feature. The **RavenDB Cloud Documentation** has documentation about the cloud hosting. The **About Examples** section explains how the examples throughout the documentation are made. All the documentation exists for all versions of RavenDB and for the languages C#, Java, Python, and NodeJS.

At first sight, the documentation seems really helpful and well explained, but once a person begins to implement it, challenges emerge. This technology is much more robust for C# than for other languages. While exploring the website, there are links for NodeJS that are wrong and do not lead to the right documentation. And even for some versions of RavenDB, there is no documentation for NodeJS, so to implement with that language, one has to see older versions of RavenDB and adapt. Another problem emerges in the examples, although they are useful, they are not always complete. For instance, the example uses something that should be imported but does not show how to import it.

Besides the documentation, the website also provides videos, made by the CEO, on how to implement features [9]. The only problem is that those videos only use the language C#.

There is also an offer of demos for multiple features, for all the languages mentioned above [11]. In these demos, there is a code walk-through, where the code is explained step by step, line by line. A person can follow these demos like classes since once you finish a demo, it is stored in cache and the website lets you know which ones you already saw.

2 MAIN FEATURES

When it comes to features, they can be divided into thirteen. The main features are querying, indexes, clusters, and monitoring.

2.1 Querying

RavenDB has its language, RQL (Raven Query Language), that "*allows to execute all available types of queries and is a part of the JavaScript patching API*". One can perform "*full-text search operations over one or more fields at once with the ability to supply a custom analyzer*". The search is performed by matching "*with the terms in the index being queried, being index's terms derived from the values of the documents' textual fields. These values were converted into one or more terms depending on which Lucene analyzer the index used*".

2.2 Indexes

Indexes may be user defined or automatically generated and have advanced features for special data types. It offers support for full text search indexes. Auto indexes can be created dynamically and are designed for "*when there is no need to customize the capabilities extensively*".

2.3 Clusters

RavenDB's clustering provides redundancy and increased data availability that is consistent across a fault-tolerant, High-Availability cluster. These allow for maintaining operation in case of node failures and also for balancing load across clusters. Data sharding is not supported.

2.4 Monitoring

RavenDB has a very intuitive and appealing GUI and when it comes to monitoring offers some dashboards for this purpose. There is a Server Dashboard where one "*can monitor in real-time aggregate crucial information like CPU usage, memory consumption, state of the databases, server traffic, storage information, and much more*". Additionally, there also exists a Cluster Dashboard which displays "*data related to all cluster nodes in this view*", and also allows personalization.

3 DATA MODEL AND SUPPORTED DATA OPERATIONS

One of the primary distinctions between a relational database and a document database is the way the data is modeled. RavenDB uses a Document Database model where a lot of the features handle many of the challenges relational databases force. There are many ways to model the data with RavenDB using attachments, counters, and Time Series. RavenDB is built so even modeling data using indexes is possible. "*Every RavenDB document is an aggregate, and every aggregate is a document*". Where an "*aggregate is a cluster of domain objects that can be treated as a single unit. Modeling techniques for aggregates work well for document-oriented design, which gives a resource for modeling in the real world*".

As for the operations, they are a "*set of low-level commands used to manipulate data, execute administrative tasks, and change the configuration on a server. Common operations include set-based operations for the Patching or removal of documents by using queries*".

4 APPLICATION PROGRAMMING INTERFACES AND CLIENT LIBRARIES

RavenDB handles all interaction with the server and databases over a REST client API. It also provides client libraries for C#, Java, Node.JS, Python and Go. These are the preferred way of interacting with the database. The C# library is the most complete implementation and by far the most well documented.

The RavenDB Studio is also a client side interaction feature worth highlighting. Working as the GUI for managing RavenDB servers, it provides quite powerful tools for building and visualising queries and indexes, as well as for tracking performance and traffic metrics.

5 CONSISTENCY FEATURES

RavenDB ensures both ACID [3] (atomic, consistent, isolated, and durable) and BASE (basically available, soft state, eventually consistent) "*in different parts of its operations. In general, ACID is the ideal end goal. A fully consistent model makes it easy to build upon and reason about, but it also can be expensive to build and maintain. In a distributed system, ensuring atomicity can be quite expensive, since you need to talk to multiple machines for each operation, for example. On the other hand, BASE gives a lot more freedom, and that translates into a lot more optimization opportunities. In RavenDB, all operations on a document or attachment using its ID (put, modify, delete) are always consistent and run in an ACID transaction*".

Improving performance while keeping ACID consistency throughout the system has been a challenge. One of the latest improvements to RavenDB is the development of Voron, a custom-made storage engine, designed specifically to maximize the performance and eliminate integration issues.

6 REPLICATION

RavenDB offers extensive support for data replication and distributed database access. RavenDB cluster consensus is assured by the Rachis protocol, based on Raft consensus. This strong consistency guarantee is used for cluster-wide operations, such as creating databases, adding new nodes to a cluster and electing a new cluster leader. In order for the protocol to function correctly, at least 3 nodes should be clustered, to ensure a majority in the case of node failure. Most common cluster configurations are 3 or 5 clusters.

Databases are handled by clusters differently. When creating a database in a cluster, you may chose it's replication degree and the nodes that should keep it. Most commonly, you keep a database in all nodes. RavenDB prioritizes database availability for writes, therefore, it does not wait for consensus at every database operation. In fact, database operations are completely agnostic from cluster organization and status, from the client side. The job of replicating data through the cluster is the node's, that sends the transactions it receives to the other nodes in the cluster through TCP connections.

Ensuring write availability means that conflicts are possible when concurrent transactions operate on the same document in different nodes. The database has a conflict resolution algorithm that can be customized to better suit application needs. Behaviour can be set to accept the latest document, to accept the document ftom an authoritative node or to merge documents based on custom functions.

7 DATA PROCESSING FEATURES

RavenDB provides a lot of data processing features. It comes with its storage engine, full-text search engine, Map-Reduce indexing, Graphical User Interface (GUI), built-in security in transit and at rest, ETL Replication, Pull Replication, and even a memory management system for caching and garbage collection. There is no need to pay extra for the safety of data or any of the other features or even add-on parts like Elasticsearch or Hadoop. The advantages of having everything you need in one place are huge: homegrown features can tweak the database to create a perfect fit; no need for outside integration which means less work.

8 USE CASES

Just like any other database, RavenDB has appropriate use cases, as well as some problematic scenarios.

8.1 Adequate

"*RavenDB is focused on being a good database for software programs capable of supporting transaction-oriented applications on the internet (OLTP) and as a persistent view model store (usually for the performance-critical pages)*" [15].

Recurrent use cases are e-commerce, management apps, and distribution systems. This is because of the RavenDB's high performance, managed cloud option, document data model, and so on.

On the website, there are loads of examples of real-case scenarios where RavenDB was used [8].

8.2 Problematic Scenarios

"*Although RavenDB is suitable for a whole range of problems, there is one specific scenario that is not that adequate*". The CEO does not recommend its use for reporting since reporting databases need to answer queries that would be uneasy or slow on a RavenDB system [14]. Despite RavenDB being able to do reporting, what to report must be known apriori, which is not the desired behavior in most reporting systems where users are given a lot of freedom on what to do.

Regardless of e-commerce being one recommended use case for RavenDB, the methods provided aren't much robust for this kind of problem, especially in terms of security.

A feature that also seems to be lacking is Range Queries which appears to have some bugs in recent versions, mainly the manipulation of the numbers.

9 COMPARISON TO OTHER TECHNOLOGIES

"*In RavenDB, the database schema isn't fixed for long. Instead, data is stored without schema in the form of JSON documents. Thus the documents have many arbitrary structures as well as attributes that are associated with them. RavenDB makes the best use of Indexes, which are automatically created, which depends on the user's usage, or are created explicitly by the clients.*" Besides offering a high number of features, the on-premise license is also cheaper.

The website has a section comparing RavenDB with other databases such as MongoDB, CosmosDB, and Postgresql [6]. "*RavenDB tops known databases in many areas and features like concurrency control, single-node transactions, distributed transactions, querying, and indexing.*"

9.1 Compare to SQL solutions

Document based solutions such as RavenDB have different characteristics, stated by MongoDB [1], that allow it to:

- "*Handle large volumes of data at high speed with a scale-out architecture*"
- "*Store unstructured, semi-structured, or structured data*"
- "*Enable easy updates to schemas and fields*"
- "*Be developer-friendly*"
- "*Take full advantage of the cloud to deliver zero downtime*"

Such is too difficult or even impossible with a SQL solution.

SQL has advantages as well, but when compared with RavenDB they are hard to find. For example, one advantage of SQL is being a standardized language due to documentation and long establishment over the years. Although RavenDB uses its own language RQL (Raven Query Language), it is similar to SQL, being simple. So language simplicity is no longer a reason why people should use SQL.

9.2 Compare to NoSQL solutions

A drawback of using RavenDB is that it does not support sharding while MongoDB, for example, does. However, when it comes to performance, none can compare with RavenDB.

A study was made comparing RavenDB and Couchbase [5] and the key findings were:

- RavenDB outperforms Couchbase by orders of magnitude when using queries under load
- Couchbase proved fragile in production, with high overhead on failure conditions. On the other hand, ravenDB's failure model proved resilient and allowed higher operational flexibility and peace of mind

- RavenDB stores the data at 1/3 of the storage needed by Couchbase
- RavenDB cloud budget configurations are 80% cheaper at a comparable, 99.99 percentile latency

These key finding demonstrate the advantages of using RavenDB.

THE PROTOTYPE

1 TOPIC

Since e-commerce was one example of a suitable use case, the prototype developed was an e-commerce web platform. In this web app, the user can see a list of products from multiple stores, filter, and even search for products. On the product page, more information is displayed, like users' reviews.

2 DATA PREPARATION

To fulfill the prototype's purpose, a lot of data was needed. However, with a quick search, it became clear that the specific attributes wanted weren't easy to find on any dataset. Therefore, after a search on Kaggle, five datasets were chosen, being the criteria, having lots of products with their name and description [18] [16] [22] [20] [21]. Afterward, with Python and its libraries, some scripts were made to prepare the data. The fields for stock, price, and rating were given random values to replace the null ones. All duplicates and products without names were removed. Some uniformization was made regarding the categories and reviews to an array of strings and an array of objects respectively. The clients were generated with a script that provided random names, and birthdates and joined on the website dates. The purchases were also generated randomly from all the products.

3 CONCEPTUAL DATA MODEL

The prototype developed uses many datasets of different entities and how they relate to each other are modeled in Figure 1 The data is divided into six classes. Each store has many products and each product can have zero or more reviews. The clients can make purchases of many products from different stores. Each purchase is stored as a Line which is the record of said purchase.

4 PHYSICAL DATA MODEL

The high-level physical structure decisions made can be found in Figure 2. For the backend, the database is RavenDB with NodeJS. The communication between the server and client machine is made with TCP/IP.

5 ARCHITECTURE

Our prototype relies on a simple Node.JS backend that provides a few useful endpoints and that uses the Node.JS RavenDB library to interact with the database. For the frontend application, we are using React and send our requests to the backend API for processing.

6 INITIALIZATION AND DATA STRUCTURES

We setup our RavenDB instance as a Docker container built using RavenDB's official image for version 5.3. After the container is up, the database is initialized by creating it in the Studio GUI. Next we populate the database using a series of Python scripts that send bulk requests through the REST API. To initialize the prototype, you must run the Node.JS app in both the client and server packages.

For operating in cluster mode, instead of creating a single container, another two are created and configured to talk to each other. Next, a script that configures the cluster nodes must be executed.

The data structures used for storing documents are a series of JSON documents. An illustrative example of these can be found in Annex C

7 IMPLEMENTED FEATURES

To explore the technology, it was developed a very simplified website to demonstrate the main features, such as creating a user, editing and deleting products, buying a product, listing the purchases made by a user, and searching and filtering products.

To perform these actions, the following RavenDB features were explored: query, indexes, more like this, suggestions and faceted search.

7.1 Query

Of the three possibilities given, it was chosen to use DocumentQuery and RawQuery, both supporting result paging. The former was used for its simplicity and practicality as it provides different methods for querying the database without having to understand RQL. On the other hand, the RawQuery allows queries to be made directly in RQL. It was used when the DocumentQuery was not sufficient, for example when it was necessary to filter by several fields and deal simultaneously with ranges of numbers. Although this was the best way to work around the problems encountered with some DocumentQuery methods, it should be noted that it is not the safest option since there is no parameterization of values, which are inserted directly into the query without much verification.

7.2 Indexes

Indexes are the only way to satisfy queries in RavenDB. Since they execute in the background, indexes allow the server to return search results almost immediately even if a large number of documents have just been changed. Whenever there is a query that involves searching, RavenDB automatically creates an index, if none has been queried.

Indexes can be classified into two types, both of which were used in the implementation:

- **Map**: contains one or more map functions that identify the fields to index and which documents and fields to search for. It was used, for example, in searching and filtering products;
- **Map-Reduce**: selects the required records, using a map function, followed by a reduction function that produces a smaller number of results. This was used for getting the purchases with products sold by a specific store while showing only the products that are sold by the same store.

Map-Reduce indexes are quite powerful in RavenDB and surprised us for their versatility. The example we showed involves basically creating an entire new class of document, akin to a View in SQL.

For each index, one can configure the document fields where it can be specified whether a field is used for full-text search or suggestions, whether it needs to be highlighted, and which type of indexing (search, exact, etc) and analyser (StandardAnalyzer, StopAnalyzer, SimpleAnalyzer, etc) to be used.

RavenDB provides a term boosting mechanism since the indexing is built on top of the Lucene engine. It enables each field's relevance level to be entered to improve the search engine "*and provide users with much more accurate results*". It was used to ensure that, when searching by product, the product name field has more relevance than the description, and that as a result the products with the names referring to the search appear first.

7.3 More Like This

MoreLikeThis provides a list of similar results for a given item, using the whole item to compare or specific fields.

This was used to show similar products to a given product, a feature so often found on e-commerce websites.

7.4 Suggestions

RavenDB provides a powerful suggestions tool to assist the user in searching for an item. Given one or more words, the server, using the appropriate function, returns the suggestions for the input, providing alternatives to the user when for example there are no results for the initial search.

It was used to suggest to the user names of clients.

7.5 Faceted Search

In addition to pagination, it is useful to give feedback to users on how many results per category were returned by the server in a given search. For this, there is Faceted Search which consists in grouping the values of a given field in facet values, with an associated facet count. Therefore, the user will be able to more easily interpret the results and understand if they need to narrow their search.

This was used in the filtering of the products.

CONCLUSIONS

In this report, the NoSQL database selected is described as well as the prototype developed to highlight the database's main features. RavenDB's website is extremely complete and intuitive, so finding the documentation needed was easy. At first glance, it appears that the documentation was helpful in every way. However, when the implementation started, that did not remain true. Raven's API is much more robust for C# so by choosing node, one's job can be very complicated. The documentation is not as good and explanatory for node. Despite that, a prototype was developed that showcases RavenDB's main features. One thing that is worth noting is Raven's incredibly versatile indexing, which allows for executing complex queries with low cost.

REFERENCES

[1] Eliot Horowitz Dwight Merriman and Kevin Ryan. 2022. *Advantages of NoSQL Databases.* MongoDB. Retrieved June 03, 2022 from https://www.mongodb.com/nosql-explained/advantage

[2] Oren Ein. 2004. *Ayende @ Rahien Blog.* RavenDB. Retrieved June 03, 2022 from https://ayende.com/blog/

[3] Oren Ein. 2008. *ACID database transactions.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/why-ravendb/acid-transactions

[4] Oren Ein. 2008. *Buy RavenDB NoSQL Database.* RavenDB. Retrieved June 02, 2022 from https://ravendb.net/buy

[5] Oren Ein. 2008. *Couchbase vs RavenDB Performance at Rakuten Kobo.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/whitepapers/couchbase-vs-ravendb-performance-at-rakuten-kob

[6] Oren Ein. 2008. *Databases comparison.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/comparison?ravendb-vs-mongodb-vs-cosmosdb-vs-postgresql

[7] Oren Ein. 2008. *NOSQL database guide.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/learn/docs-guide

[8] Oren Ein. 2008. *NOSQL Ddatabase use cases.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/news/use-cases

[9] Oren Ein. 2008. *NoSQL Videos.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/learn/videos

[10] Oren Ein. 2008. *RavenDB.* RavenDB. Retrieved June 03, 2022 from https://ravendb.net/

[11] Oren Ein. 2008. *RavenDB Step-By-Step Coding Walkthrough.* RavenDB. Retrieved June 03, 2022 from https://demo.ravendb.net/

[12] Oren Ein. 2008. *RavenDB support options.* RavenDB. Retrieved June 02, 2022 from https://ravendb.net/support

[13] Oren Ein. 2008. *Your free RavenDB NOSQL database license.* RavenDB. Retrieved June 02, 2022 from https://ravendb.net/why-ravendb/tools-with-your-free-ravendb-nosql-database-license

[14] Oren Ein. 2011. *When should you NOT use RavenDB?* RavenDB. Retrieved June 03, 2022 from https://ayende.com/blog/136197/when-should-you-not-use-ravendb

[15] Oren Ein. 2011. *When should you use RavenDB.* RavenDB. Retrieved June 03, 2022 from https://ayende.com/blog/136196/when-should-you-use-ravendb

[16] JCPenney. 2017. *JCPenney products.* Kaggle. Retrieved June 03, 2022 from https://www.kaggle.com/datasets/PromptCloudHQ/all-jc-penny-products

[17] Jeff Atwood Joel Spolsky. 2008. *Questions tagged [ravendb].* StackOverflow. Retrieved June 02, 2022 from https://stackoverflow.com/questions/tagged/ravendb

[18] Griko Nibras. 2020. *Amazon Cell Phones Reviews.* Kaggle. Retrieved June 03, 2022 from https://www.kaggle.com/datasets/grikomsn/amazon-cell-phones-reviews

[19] Paweł Pekról. 2007. *Welcome to RavenDB Community Discussions.* Github. Retrieved June 02, 2022 from https://github.com/ravendb/ravendb/discussions

[20] PROMPTCLOUD. 2017. *Flipkart Products.* Kaggle. Retrieved June 03, 2022 from https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products

[21] PROMPTCLOUD. 2017. *Toy Products on Amazon.* Kaggle. Retrieved June 03, 2022 from https://www.kaggle.com/datasets/PromptCloudHQ/toy-products-on-amazon

[22] Emily Russell. 2020. *eCommerce Products.* Kaggle. Retrieved June 03, 2022 from https://www.kaggle.com/datasets/mewbius/ecommerce-products

A CONCEPTUAL DATA MODEL

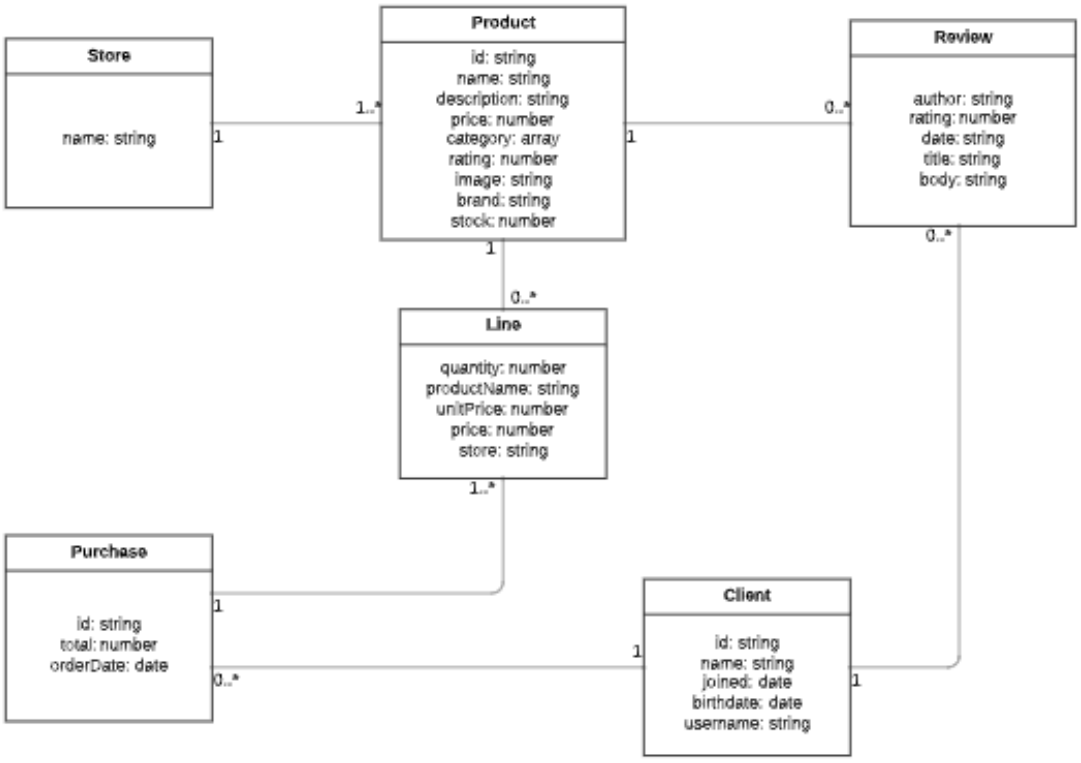

Fig. 1. Conceptual Data Model

B PHYSICAL DATA MODEL

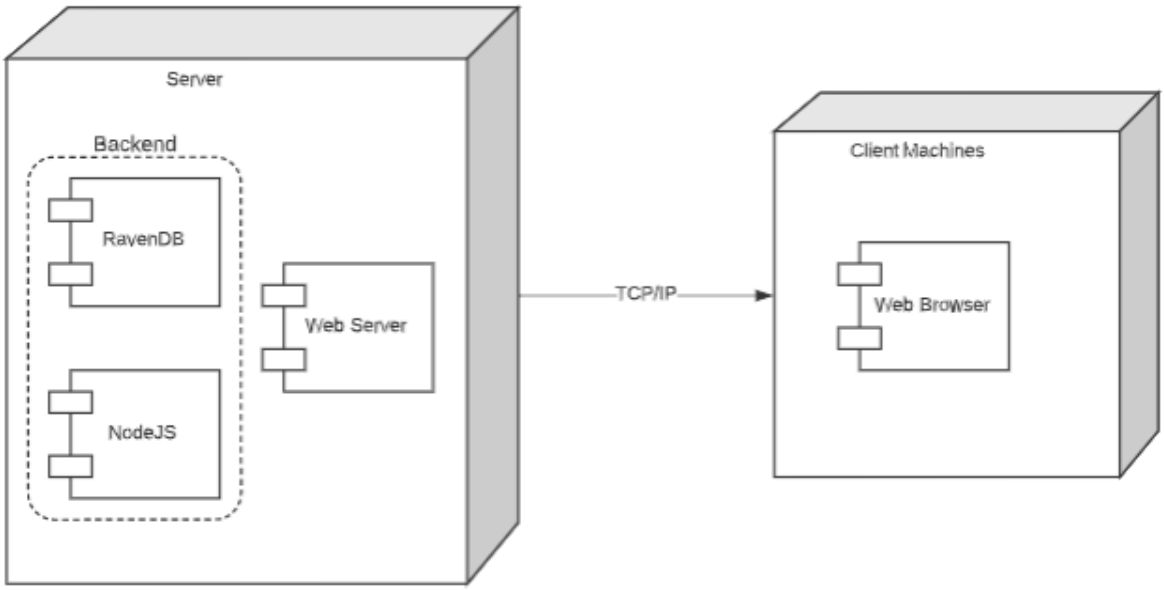

Fig. 2. Physical Data Model

C DATA STRUCTURE EXAMPLES

Client:

```
{
    "name": "Julius Edwards",
    "birthdate": "1951-06-22T20:28:01",
    "joined": "2015-07-17T19:44:45",
    "Id": "clients/1",
    "username": "Julius-Edwards-1951"
}
```

Product:

```
{
    "name": "Hornby 2014 Catalogue",
    "brand": "Hornby",
    "price": 3.42,
    "stock": 121,
    "rating": 4.9,
    "category": [
        "Hobbies",
        "Model Trains & Railway Sets",
        "Rail Vehicles",
```

```
        "Trains"
    ],
   "description": "Product Description Hornby 2014 Catalogue Box Contains 1 x one catalogue",
    "product_information": "Technical Details Item Weight640 g Product Dim...",
    "reviews": [
        {
            "author": "Copnovelist",
            "rating": "4.0",
            "date": "2014-4-6",
            "title": "Worth Buying For The Pictures Alone (As Ever)",
            "body": "Part of the magic for me growing up as a boy..."
        },
        ...
        }
    ],
    "store": "amazon",
    "Id": "products/amazon/1"
}
```


Purchase:


```
{
    "Id": "purchases/2",
    "client": "clients/60847",
    "lines": [
        {
            "quantity": 1,
            "productId": "products/flipkart/8555",
            "productName": "E-plant Plant Container Set",
            "unitPrice": 999.0,
            "price": 999.0,
            "store": "flipkart"
        },
        {
            "quantity": 9,
            "productId": "products/flipkart/14275",
            "productName": "Oviyon Embroidered Men's Polo Neck T-Shirt",
            "unitPrice": 999.0,
            "price": 8991.0,
            "store": "flipkart"
        },
        {
            "quantity": 2,
            "productId": "products/flipkart/85",
           "productName": "Redbag Eight Armed Goddess Sherawali Maa Showpiece  -  10.8 cm",
            "unitPrice": 1600.0,
            "price": 3200.0,
```

```
            "store": "flipkart"
        }
    ],
    "total": 13190.0,
    "orderDate": "2021-06-18T00:56:02"
}
```