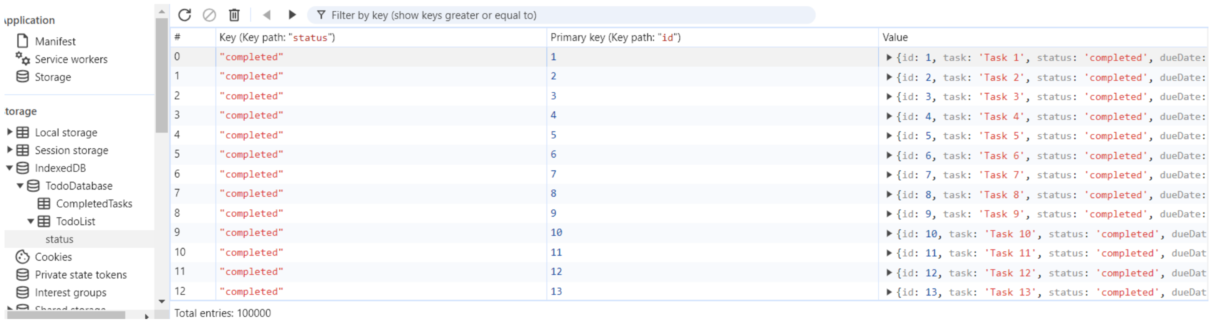
Project Part 1: Data and Scalability

1.Team member names and student IDs:

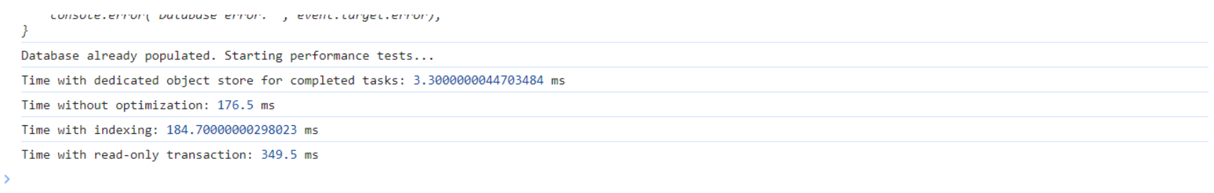
Mythry Rudra - 989473114
Rani Supriya - 989471012
Naveena Voora - 989472599

2. Screenshots of the implementation tasks 1 through 6



#	Key (Key path: "status")	Primary key (Key path: "id")	Value
0	"completed"	1	▶{id: 1, task: 'Task 1', status: 'completed', dueDate:
1	"completed"	2	▶{id: 2, task: 'Task 2', status: 'completed', dueDate:
2	"completed"	3	▶{id: 3, task: 'Task 3', status: 'completed', dueDate:
3	"completed"	4	▶{id: 4, task: 'Task 4', status: 'completed', dueDate:
4	"completed"	5	▶{id: 5, task: 'Task 5', status: 'completed', dueDate:
5	"completed"	6	▶{id: 6, task: 'Task 6', status: 'completed', dueDate:
6	"completed"	7	▶{id: 7, task: 'Task 7', status: 'completed', dueDate:
7	"completed"	8	▶{id: 8, task: 'Task 8', status: 'completed', dueDate:
8	"completed"	9	▶{id: 9, task: 'Task 9', status: 'completed', dueDate:
9	"completed"	10	▶{id: 10, task: 'Task 10', status: 'completed', dueDat
10	"completed"	11	▶{id: 11, task: 'Task 11', status: 'completed', dueDat
11	"completed"	12	▶{id: 12, task: 'Task 12', status: 'completed', dueDat
12	"completed"	13	▶{id: 13, task: 'Task 13', status: 'completed', dueDat

Total entries: 100000



```
    console.error('Database error:', event.target.error);
}
Database already populated. Starting performance tests...
Time with dedicated object store for completed tasks: 3.3000000044703484 ms
Time without optimization: 176.5 ms
Time with indexing: 184.70000000298023 ms
Time with read-only transaction: 349.5 ms
```

Project: Performance Optimization Techniques for NoSQL Databases Using Indexed DB

Objective:

To explore performance optimization techniques for large datasets within Indexed DB by simulating a 100,000-record 'Todo List' database. The objective is to evaluate the effectiveness of different optimization techniques and understand the impact of database design and configuration on retrieval speed.

Tasks and Results:

Baseline Test (Without Optimization)

Execution Time: 176.5 ms

Description: In this test, all records were retrieved without any specific optimization applied, providing a baseline performance measure for comparisons.

Lesson Learned: Running the test without optimization provided a useful benchmark against which other techniques were evaluated.

Challenge Faced: The retrieval time was notably high, demonstrating the inefficiency of fetching large datasets without any specific data retrieval structure.

Insight: The baseline test underlined the need for optimizations in large datasets, as the retrieval time was too high for practical applications in performance-critical scenarios.

Test with Indexing on Status Attribute

Execution Time: 184.7 ms

Description: This test used an index on the status attribute to optimize retrieval. The goal was to allow IndexedDB to locate "completed" tasks more efficiently by leveraging this index.

Lesson Learned: The indexed retrieval did not improve performance as anticipated, yielding a longer execution time compared to the baseline.

Challenge Faced: The index on "completed" tasks did not deliver a performance benefit, potentially due to the high ratio of records being retrieved or the mixed dataset structure, which made the index less effective.

Insight: This test emphasized that indexing is not always a straightforward solution to performance problems in NoSQL databases, especially with large datasets that involve mixed status fields.

Test with Read-Only Transaction

Execution Time: 349.5 ms

Description: A read-only transaction was used in this test, intending to reduce overhead by restricting the transaction to data reading operations only.

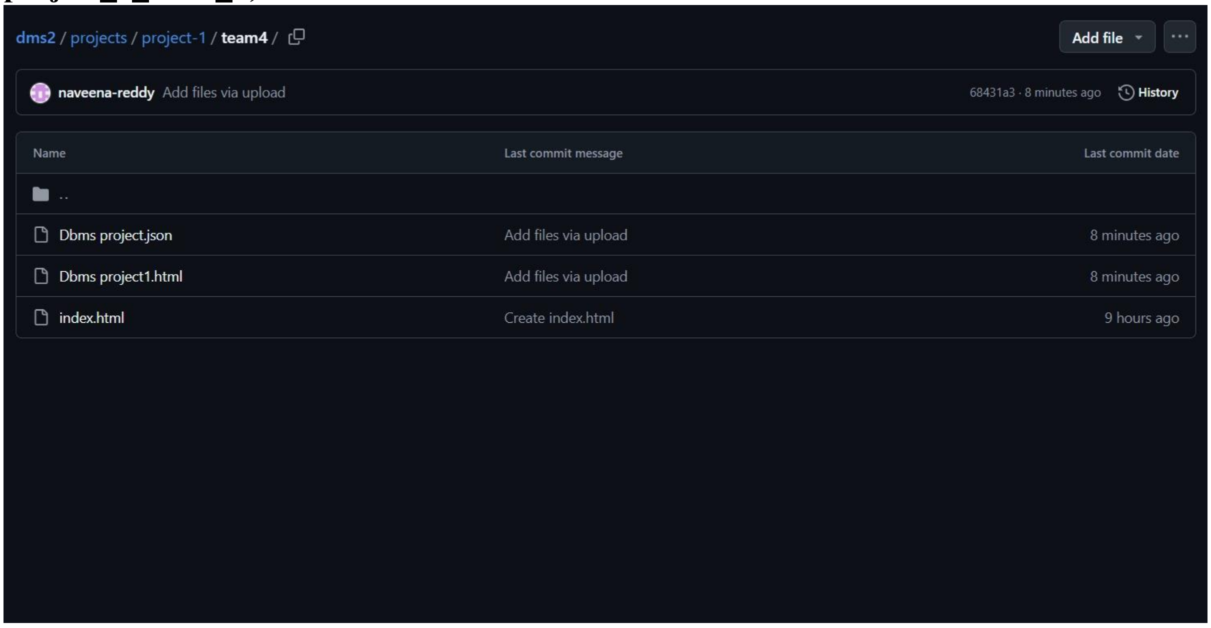
3. Lessons learned from the project

According to goals, utilizing a read-only transaction unexpectedly led to decreased performance.

It is clear that transaction settings do not always equal faster performance in IndexedDB because the read-only transaction optimization produced no performance gains.

The significance of verifying every possible optimization was brought to light by this assignment. Although read-only transactions are thought to reduce overhead, in fact they do not always improve efficiency, especially when working with large datasets in IndexedDB. One of the best ways to optimize big NoSQL databases is to separate frequently requested groups of data into separate object stores. This resulted in a demonstration of the significance of object store structure and data modeling for high-speed access in performance-critical applications.

4. Screenshot of the GitHub branch submission (e.g., project_1_team_1, project_1_team_n)


dms2 / projects / project-1 / team4 /

Add file

naveena-reddy Add files via upload

68431a3 · 8 minutes ago History

Name	Last commit message	Last commit date
..		
Dbms project.json	Add files via upload	8 minutes ago
Dbms project1.html	Add files via upload	8 minutes ago
index.html	Create index.html	9 hours ago