# Redis Project: Tommies Idea Board

By *Luca Comba*

## Introduction

This project explores Redis, an in-memory data structure store, often used as a cache, message broker, and database. It's known for its speed and flexibility in handling various data structures beyond simple key-value pairs.

The project involved learning about Redis, its history, core commands, and common use cases. To apply this knowledge, a web application called "Tommies Idea Board" was developed using Go. This application allows users (Tommies) to share ideas, which are then stored and retrieved from a Redis database.

**Why Redis?** Redis was chosen for its high performance due to its in-memory nature, making it suitable for applications requiring rapid data access, like real-time idea sharing. Its support for various data structures like Hashes was also a good fit for storing structured idea data.

**Minimal Goal** Understand the fundamentals of Redis (commands, data types, persistence) and build a basic CRUD (Create, Read, Update, Delete) application using Redis as the database.

**Optimal Goal** Deploy the "Tommies Idea Board" application with a functional front-end, allowing multiple users to post and view ideas stored in Redis, and explore more advanced Redis features if time permitted.

## Exploring

My previous database experience was primarily with relational databases like PostgreSQL or MySQL. This project was an opportunity to explore the NoSQL landscape, specifically key-value and in-memory databases.

**Initial Ideas** Initial ideas involved exploring different NoSQL databases or perhaps focusing on a specific database concept like indexing or transaction management in a relational context.

**Other Options** Other NoSQL options like MongoDB (document database) or Cassandra (wide-column store) were considered. However, the speed and specific use cases of Redis (caching, session management, real-time data) seemed particularly interesting and distinct from relational models.

**Final Decision** I have decided to focus on Redis because of the large usage in the industry, its unique features, and the opportunity to learn about in-memory databases and key-value stores, which I believe to be extremely intereting.

## Building

The project involved setting up a Redis instance (likely via Docker as shown in the `docker-compose.yml`), learning the Go Redis client library (`go-redis`), designing the data storage approach, and building a simple Go web server to handle requests. A basic front-end was also developed to interact with the API.

The core design utilizes Redis Hashes to store individual ideas. Each idea is stored under a unique key (e.g., `idea:<id>`). A simple key-value pair (`idea_uid`) might have been used initially or for tracking the last used ID. The application uses a Go backend to interact with Redis and serve API endpoints for creating and retrieving ideas.

Manually generated data was used for testing, specifically when the server is initialized (`SET foo bar`, `HSET` for the first idea). The idea hash structure included fields like timestamp, title, description, writer, and tags.

The application supports: - Storing key-value pairs (e.g., `SET foo bar`, accessed via `/store/{key}` endpoint). - Storing idea details in Hashes (e.g., `HSET idea:1 title "My Idea" ...`, potentially accessed via `/idea/{id}` endpoint). - Retrieving a specific idea by its ID (e.g., `HGETALL idea:1`, accessed via `/idea/{id}`). - Retrieving all ideas (likely involving scanning keys or using a Set/List to track idea IDs, accessed via `/ideas` endpoint).

## Discovering

**What Went Well** The project provided practical experience with Redis, moving beyond theoretical knowledge. Key learnings included: - The speed advantage of in-memory storage. - The directness and simplicity of Redis commands compared to SQL queries. - The utility of different Redis data structures (Strings, Hashes) for specific tasks. - How to integrate Redis with a backend application (Go). - Containerization using Docker for development and deployment.

**What I Learned** Before this project, my understanding of NoSQL databases, particularly in-memory key-value stores like Redis, was limited. This project significantly advanced my knowledge by requiring hands-on implementation. I now have a much better grasp of when and why to choose Redis, its core mechanics, and how to build applications with it.

**Challenges** Perhaps implementing user authentication or more complex querying (e.g., searching ideas by tag efficiently) wasn't completed. Managing relationships (e.g., linking users to ideas) is less straightforward in Redis than in relational databases, which could have been a challenge. Ensuring data durability and understanding the trade-offs between snapshotting and AOF persistence might have required more time.

More sophisticated querying using Redisearch, implementing user sessions using Redis, or adding real-time features using Pub/Sub could be explored. Error handling and production-ready deployment configurations would also be areas for improvement.

## Class Topics Illustrated

1. Key-Value Stores
2. NoSQL vs SQL
3. In-Memory Databases
4. Database Clients and APIs
5.

## Correctness and Completeness

TO DO