CONSTRUCTOR UNIVERSITY

Local-first Collaboration on Relational Data

by

Veronika Sirotkina

Bachelor Thesis in Computer Science

Submission: April 2, 2024

Supervisor: ??? Anton Podkopaev

Constructor University | School of Computer Science and Engineering

Statutory Declaration

Family Name, Given/First Name	Sirotkina, Veronika
Matriculation number	30006541
Kind of thesis submitted	Bachelor Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

....

Date, Signature

Abstract

TODO

Contents

1 Introduction

With internet becoming more accessible and cloud-based software gaining momentum, more apps started to heavily depend on the presence of internet connection to work properly. This is especially true for apps that need to synchronize clients' data across multiple devices or apps that allow for real-time collaboration between clients.

The most obvious and maybe the easiest way to enable synchronization across devices is naturally to store an authoritative copy of the data in the cloud and provide a centralized server to manage changes coming from the clients. Real-time collaboration can be achieved by implementing operational transformation algorithms [10].

This approach is, of course, viable and is widely used in practice, but it has a big downside. Without internet connection access to the data is either lost completely or it is only available in a read-only mode. The access can also be lost if the provider company's servers are down, or if the company stops supporting the product, or if the company finds the contents of the document inappropriate [2].

Local-first software [6] is a new approach to designing collaborative software. In the context of this paper the most notable feature of local-first software is that it allows clients to edit shared documents even when they are offline. The changes can be integrated with the upstream version once the connectivity is reestablished.

GanttProject [5, 4] is an open-source tool for building Gantt diagrams. It's originally local-only, but at the current time it also provides a cloud storage [3]. A work-in-progress module called Colloboque will enable real-time colloboration in GanttProject.

The goal of this project is to enable local-first real-time collaboration in GanttProject by further developing Colloboque.

2 Background

TODO

2.1 Operational Transformation

TODO

2.2 Local-first Software

The original paper [6] suggests 7 principles of local-first software. Here I will list 4 which I find the most relevant.

- **Synchronization across devices**
  Imagine a client who uses a document editor on multiple devices. Synchronization involves tracking changes made by the client on one device and ensuring these changes are reflected across all other devices used by the client. In the end data on all devices must reach the same state. This is a very convenient feature that lets clients acces their work from any device.
- **The network is optional**
  Nowadays it is normal for many apps to lose most of their functionality if internet

connection is unstable. TODO: EXAMPLES. For local-first software it is important that it should retain its core functionality even when the device is offline. The client is still able to view and edit documents as he pleases, and the changes made while being offline are integrated with other replicas when the device connects to the network again.

- **Seamless collaboration**
  Real-time collaboration is a very attractive feature that lets multiplea people work on a single document simultaneously. Some notable examples of web-apps that allow for real-time collaboration are Google Docs, Google Sheets, Figma, etc. Usually such apps don't allow to edit documents offline (TODO: GOOGLE DOCS ALLOWS THIS, WHAT ABOUT GOOGLE SHEETS?). For local-first software the aim is to provide collaboration functionality on par with cloud-based apps like Figma while retaining optionality of the network.

- **Ultimate ownership and control**
  TODO

2.3 CRDT

CRDT stands for Conflict-Free Replicated Data Type [9]. CRDT object has an identifier, content, an initial state, and a set of operations. Any two objects with the same identifiers are called replicas of each other. An abstract state of a CRDT object is represented by the collective result of all read-only operations. CRDT guarantees that all well-formed update operations are commutative and idempotent with respect to the abstract state of an object. Because of these properties, replicas can make updates independently, without communicating wiht other replicas. As long as all updates are eventually communicated, all replicas are guaranteed to end up with the same abstract state.

CRDT types can be further categorized into two groups: State-based CRDTs and Operation-based CRDTs

2.3.1 State-based CRDT

First, let's define Causal History for state-based CRDT objects.

Definition 2.1 (Causal History — state-based)**.** For any replica x_i of x:

- Initially, $C(x_i) = \emptyset$.
- After executing update operation f, $C(f(x_i)) = C(x_i) \cup \{f\}$.
- After executing merge against states x_i, x_j, $C(merge(x_i, x_j)) = C(x_i) \cup C(x_j)$.

TODO

2.3.2 Operation-based CRDT

TODO

2.3.3 Automerge

Automerge [1] is a library that provides a JSON-like CRDT for JavaScript. It was designed specifically for implementing local-first software. Automerge documents support all JSON primitive data types, as well as Map, List, Text, and Counter.

TODO: AUTOMERGE USAGE EXAMPLE

Automerge documents support both merging full documents and applying individual changes. To define rules for merging documents, it's enough to define rules for merging Maps, Lists, Text and Counters. Full explanation of the merging rules in Automerge can be found in Automerge documentation [7].

TODO: CONFLICT OBJECT

2.3.4 Crecto

2.4 Leader-Based Log Replication

TODO

2.5 Time Warp

TODO

3 Approaches

TODO

3.1 CRDT approach

TODO

3.2 Time Warp approach

TODO

3.3 Transaction replay approach

TODO: POLISH CODE LISTINGS

The Colloboque server stores transaction history and the state of the project after each transaction. When a client loses connection to the server, it keeps commiting transactions locally. When connection is re-established, the set of changes applied by the client is sent to the server. The server then takes the state of the project that the out-of-sync client based its changes on and uses it to start two transactions. The first transaction contains all the changes that the client applied locally, while the second transaction contains all the changes processed by the server while the client was disconnected. The transactions make use of isolation level REPEATABLE READ.

The default mode for transactions in Postgres is READ COMMITTED [8]. It means that a transaction is allowed to read changes that where committed by other transactions after the current transaction has started.

```sql
BEGIN;
SELECT * FROM data; -- 0 rows
-- another transaction commits inserting a row
SELECT * FROM data; -- 1 rows
COMMIT;
```

When using REPEATABLE READ transaction mode, the changes from newly committed transactions are not visible, but if the current transaction makes a conflicting update, it will lead to a serialization error.

```sql
BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT * FROM data; -- 1 rows
-- another transaction commits deleting a row
SELECT * FROM data; -- 1 rows
UPDATE data SET col = 1;
-- ERROR: could not serialize access due to concurrent delete
```

With this approach the Postgres database can reliably identify whether the changes that the client made offline produce confilcts. If there are conflicts, they are discarded on the server. The client will then copy the local version of the document, sync with the server and resolve the conflicts by hand.

4 Implementation

References

[1] *Automerge*. https://github.com/automerge/automerge.

[2] Brian Fung. "A mysterious message is locking Google Docs users out of their files". In: *The Washington Post* (2017). URL: https://www.washingtonpost.com/news/the-switch/wp/2017/10/31/a-mysterious-message-is-locking-google-docs-users-out-of-their-files/.

[3] *GanttProject Cloud official web-page*. https://ganttproject.cloud/.

[4] *GanttProject GitHub repository*. https://github.com/bardsoftware/ganttproject.

[5] *GanttProject official web-page*. https://www.ganttproject.biz/.

[6] Martin Kleppmann et al. "Local-first software: you own your data, in spite of the cloud". In: *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2019. Athens, Greece: Association for Computing Machinery, 2019, pp. 154–178. ISBN: 9781450369954. DOI: 10.1145/3359591.3359737. URL: https://doi.org/10.1145/3359591.3359737.

[7] *Merging rules for objects in Automerge documents*. https://automerge.org/docs/under-the-hood/merge_rules/.

[8] *PostgreSQL Transaction Isolation Levels documentation*. https://www.postgresql.org/docs/current/transaction-iso.html.

[9] Marc Shapiro et al. "A comprehensive study of Convergent and Commutative Replicated Data Types". In: (Jan. 2011).

[10] Chengzheng Sun and Clarence Ellis. "Operational transformation in real-time group editors: issues, algorithms, and achievements". In: *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 1998, pp. 59–68.