Verteilte Systeme mit Conflict-Free Replicated Data Types

Java-Stammtisch Göttingen
9. April 2025
Tobias Baum

Office 365 ist super!

Office 365 ohne Microsoft ist noch besser!

Die Lösung: CRDTs!

Agenda

1. **Was sind CRDTs und weshalb sind sie nützlich?**
2. Konkrete Beispiele für CRDTs
3. Ein CRDT in „echter Software“

Kurz über mich …

- Tobias Baum
  - Geschäftsführer SET GmbH (http://set.de) und MDS GmbH (http://mds-it.de)
  - Forscher zu Software Engineering, insb. Code Reviews (http://tobias-baum.de)

- Was machen SET und MDS?
  - Unsere Output-Management-Software ist z. B. daran beteiligt, dass …
    - die Kontoauszüge der Sparkassen im Online-Banking landen, …
    - Versicherungsdokumente der AXA in AXA-Portal …
    - und die Post der Bundesagentur für Arbeit bei den Bürgern

- Wie komme ich zum Thema „CRDTs“?
  - Schulung eines Kollegen vor einigen Jahren ➔ interessante Idee, aber lange Jahre keine konkrete Anwendung
  - Vor 1-2 Jahren: Konkrete Anwendung für unser Produkt POSY-Postbox

Was sind CRDTs?

Conflict-Free Replicated Data Types

Was sind CRDTs?

Conflict-Free Replicated Data Types

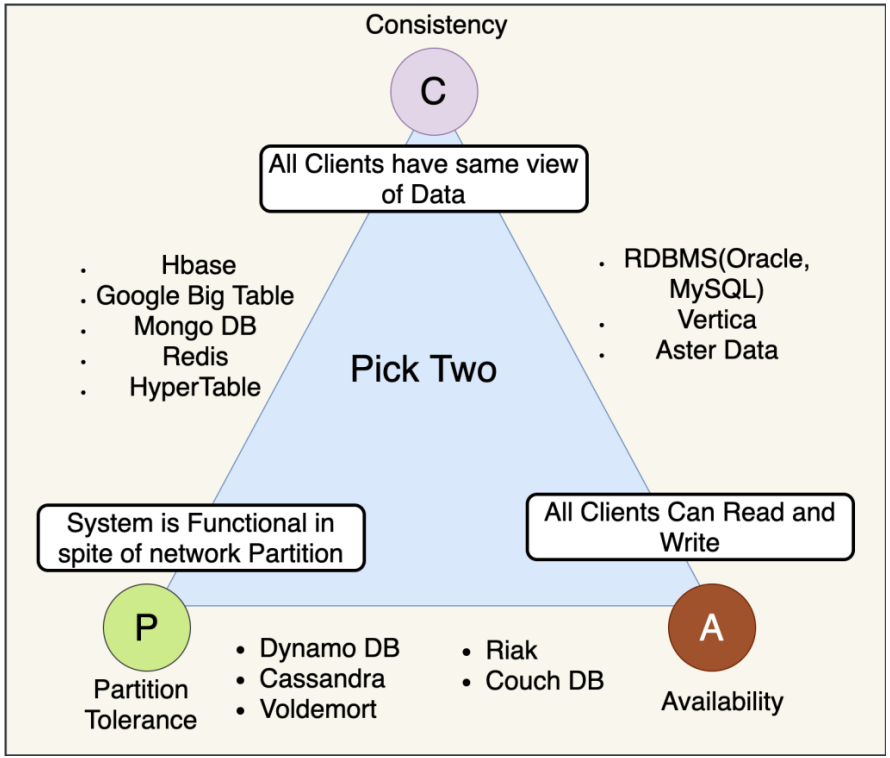
Was sind CRDTs?

Conflict-Free Replicated Data Types

Was sind CRDTs?

Conflict-Free Replicated Data Types

CAP-Theorem



Bildquelle: https://www.nitendratech.com/

Was sind CRDTs?
// Auf eine Folie gebracht

- Ein CRDT ist ein Datentyp (wiederverwendbare Klasse) mit folgenden Eigenschaften
  - Er kann auf mehrere Replikas verteilt werden
  - Jede Replika kann unabhängig voneinander nebenläufig und ohne Abstimmung mit anderen Replikas aktualisiert werden, auch offline
  - Teil der Definition ist ein Verfahren, um Inkonsistenzen/Konflikte automatisch aufzulösen
  - Replikas können zwischenzeitlich unterschiedliche Zustände haben, konvergieren aber irgendwann („eventual consistency“)

➔ Vorteil: Wiederverwendung von Konzepten und Implementierungen
➔ Vorteil: Funktioniert in Peer-to-Peer-Systemen ohne zentralen Server
➔ Nachteil: Automatische Konfliktauflösung passt nicht für jeden Anwendungsfall
➔ Nachteil: Teilweise ressourcenhungriger als „zentrale“ Ansätze

Exkurs: State-based vs Operation-based

- Formal gibt es zwei Varianten von CRDTs
  - „State-based“
  - „Operation-based“
- State-based stellt häufig geringere Anforderungen an das Netzwerkprotokoll
- Fokus dieses Vortrags sind „state-based“-Implementierungen

Agenda

1. Was sind CRDTs und weshalb sind sie nützlich?
2. **Konkrete Beispiele für CRDTs**
    1. One-Way Flag
    2. Grow-Only Counter
    3. Positive-Negative-Counter
    4. Multi-Value Register
    5. Grow-Only Set
    6. Observed-Remove-Set
    7. YATA
3. Ein CRDT in „echter Software“

One-Way-Flag
// Anwendungsbeispiel

- Ein sehr einfacher CRDT für den Anfang: „One-Way-Flag“

- Anwendungsbeispiel:
  - In einem verteilten Peer-to-Peer-System wird nach einem Signal gesucht, z. B. nach dem Beweis für außerirdisches Leben
  - Wenn einer es gefunden hat, meldet er „ich hab es“
  - Wenn diese Information alle Knoten erreicht hat, sollen alle im Zustand „jemand hat es gefunden“ sein

One-Way-Flag
// Schnittstelle

```
interface OneWayFlag {
    /**
     * Setzt das Flag auf "true".
     */
    void activate();

    /**
     * Liefert den Wert des Flags.
     */
    boolean get();

    /**
     * Fügt diese Instanz des CRDTs mit der übergebenen zusammen.
     */
    void merge(OneWayFlag other);
}
```

One-Way-Flag
// Implementierung

```
class OneWayFlagImpl implements OneWayFlag {
    private boolean flag;

    public void activate() {
        this.flag = true;
    }

    public boolean get() {
        return this.flag;
    }

    public void merge(OneWayFlagImpl other) {
        this.flag |= other.flag;
    }
}
```

Idee: Probleme durch Einschränken der Schnittstelle umschiffen

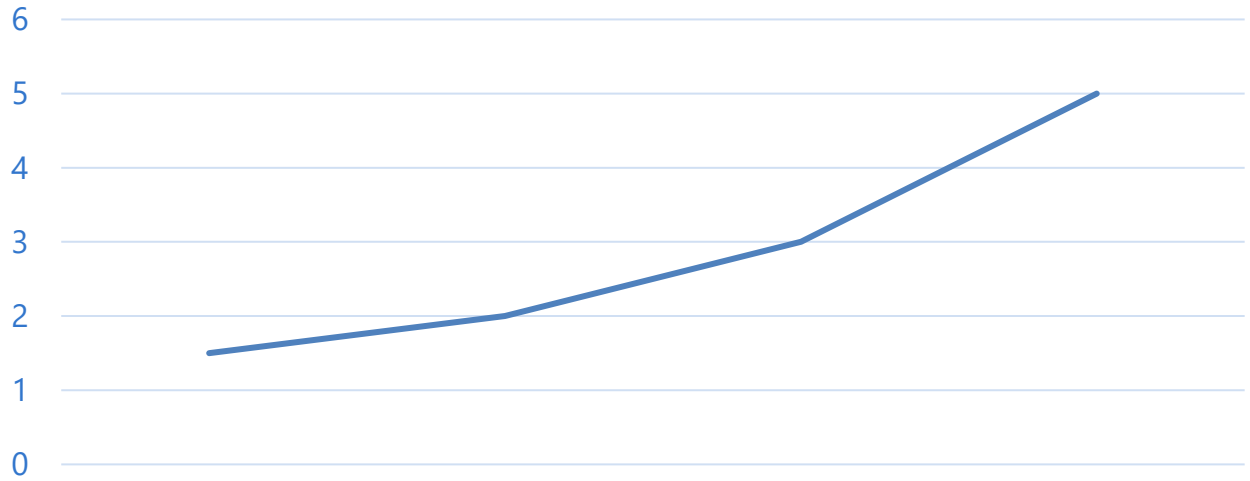
!

Wie funktioniert eigentlich der Austausch der Informationen?

- Der Austausch der Informationen übers Netzwerk kann in allen Fällen ähnlich funktionieren
  - Übermitteln des Zustands der CRDTs mit einem beliebigen Serialisierungsmechanismus (z.B. Java „Serializable“)
  - Verschiedene Mechanismen zur Auswahl der Empfänger für die Information möglich, z. B. …
    - „alles an alle verschicken“,
    - oder „an eine zufällige Auswahl der Knoten verschicken“ (Gossip-Protokoll)
  - Beim Empfang wird jeweils „merge“ aufgerufen

Grow-Only Counter
// Anwendungsbeispiel

- Anwendungsbeispiel:
  - In einem verteilten System wird etwas gezählt, z.B. Besucher an den Zugängen zu einer Veranstaltung
  - Am Ende soll eine Gesamtsumme bekannt sein



Grow-Only Counter
// Schnittstelle

```
interface GrowOnlyCounter {
    /**
     * Erhöht den Zählerstand.
     */
    void increment();

    /**
     * Liefert den Wert des Zählers.
     */
    int get();

    /**
     * Fügt diese Instanz des CRDTs mit der übergebenen zusammen.
     */
    void merge(GrowOnlyCounter other);
}
```

Grow-Only Counter
// Implementierung (1/3)

```
class GrowOnlyCounterImpl implements GrowOnlyCounter {
    private String ownId;
    private Map<String, Integer> countersPerNode = new HashMap<>();

    public GrowOnlyCounterImpl() {
        this.ownId = generateUUID();
        this.countersPerNode.put(this.ownId, 0);
    }

    public void increment() {
        this.countersPerNode.put(this.ownId,
            this.countersPerNode.get(this.ownId) + 1);
    }

    ...
}
```

Idee: Nutzung von UUIDs

Grow-Only Counter
// Implementierung (2/3)

```
class GrowOnlyCounterImpl implements GrowOnlyCounter {

    ...

    public int get() {
        AtomicInteger sum = new AtomicInteger();
        this.countersPerNode.values().stream()
            .forEach(i -> sum.getAndAdd(i));
        return sum.get();
    }

    ...
}
```

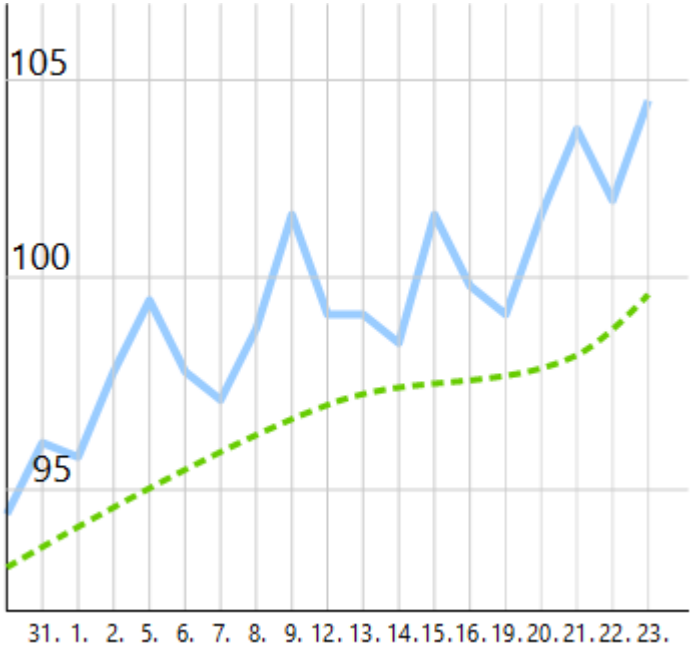
Idee: Ergebnis von „get“ bei Bedarf aus node-weisen Daten ermitteln

Grow-Only Counter
// Implementierung (3/3)

```
class GrowOnlyCounterImpl implements GrowOnlyCounter {

    ...

    public void merge(GrowOnlyCounterImpl other) {
        for (Entry<String, Integer> e : other.countersPerNode.entrySet()) {
            Integer cur = this.countersPerNode.get(e.getKey());
            if (cur == null || cur < e.getValue()) {
                this.countersPerNode.put(e.getKey(), e.getValue());
            }
        }
    }
}
```

Positive-Negative-Counter
// Anwendungsbeispiel

- Manchmal will man auch runterzählen …



Positive-Negative-Counter
// Schnittstelle

```
interface PositiveNegativeCounter {
    /**
     * Erhöht den Zählerstand.
     */
    void increment();

    /**
     * Verringert den Zählerstand.
     */
    void decrement();

    int get();

    void merge(PositiveNegativeCounter other);
}
```

Positive-Negative-Counter
// Implementierung (1/2)

```
class PositiveNegativeCounterImpl implements PositiveNegativeCounter {
    private GrowOnlyCounter positive = new GrowOnlyCounter();
    private GrowOnlyCounter negative = new GrowOnlyCounter();

    public void increment() {
        this.positive.increment();
    }

    public void decrement() {
        this.negative.increment();
    }

    public int get() {
        return this.positive.get() - this.negative.get();
    }
    ...
}
```

Idee: Mehrere CRDTs zu einem komplexeren kombinieren

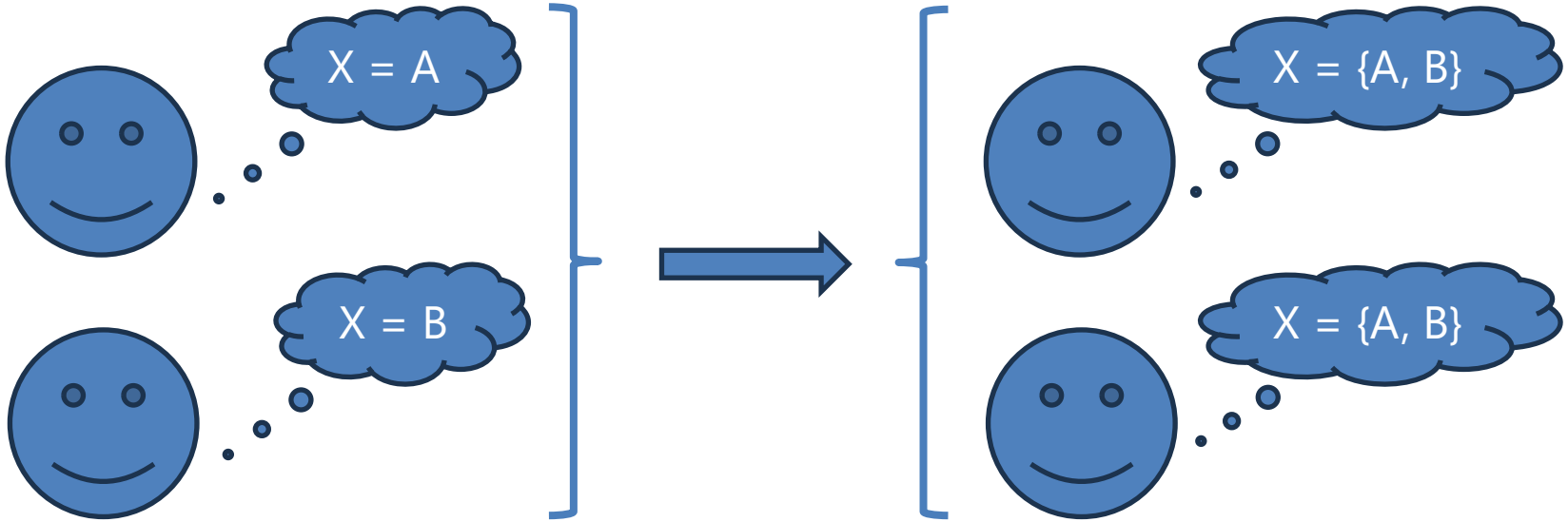
Idee: „Entfernen“ durch „Hinzufügen“ abbilden

Positive-Negative-Counter
// Implementierung (2/2)

```
class PositiveNegativeCounterImpl implements PositiveNegativeCounter {
    ...

    public int merge(PositiveNegativeCounterImpl other) {
        this.positive.merge(other.positive);
        this.negative.merge(other.negative);
    }
}
```

Multi-Value Register
// Motivation

- Anwendungsbeispiel: Ein Variablenwert soll von mehreren Nutzern geändert werden können. Falls Konflikte auftreten, werden diese dem Nutzer angezeigt.



Multi-Value Register
// Schnittstelle

```
interface MultiValueRegister<T> {
    /**
     * Setzt den Wert.
     */
    void set(T value);

    /**
     * Liefert den aktuellen Wert. Im Konfliktfall liefert es alle
     * in Konflikt stehenden Werte.
     */
    Set<T> get();

    /**
     * Fügt diese Instanz des CRDTs mit der übergebenen zusammen.
     */
    void merge(MultiValueRegister<T> other);
}
```

Multi-Value Register
// Implementierung – set

```
class MultiValueRegisterImpl<T> implements MultiValueRegister<T> {
    private static final class Entry<T> {
        private T value;
        private LogicalTimestamp time;
    }

    private List<Entry<T>> entries = new ArrayList<>();
    private VectorClock clock = new VectorClock();

    public void set(T value) {
        this.entries.clear();
        this.entries.add(new Entry(value, this.clock.tick()));
    }

    ...
}
```

Idee: Logische Zeit

Multi-Value Register
// Implementierung – get

```
class MultiValueRegisterImpl<T> implements MultiValueRegister<T> {

    ...

    public Set<T> get() {
        return this.entries.stream()
            .map(e -> e.value)
            .collect(Collectors.toList());
    }

    ...
}
```

Idee: Konflikauflösung auf den Leser/Nutzer verlagern

Multi-Value Register
// Implementierung – merge

```
class MultiValueRegisterImpl<T> implements MultiValueRegister<T> {

    ...

    public void merge(MultiValueRegisterImpl<T> other) {
        this.clock.merge(other.entries);
        for (Entry<T> oe : other.entries) {
            this.entries.removeIf(te -> te.time.isBeforeOrEqual(oe.time));
        }
        List<Entry<T>> copy = new ArrayList<>(other.entries);
        for (Entry<T> te : this.entries) {
            copy.removeIf(oe -> oe.time.isBefore(te.time));
        }
        this.entries.addAll(copy);
    }

    ...
}
```

Grow-Only Set
// Motivation

- Manchmal will man Dinge sammeln …

Grow-Only Set
// Schnittstelle

```
interface GrowOnlySet<T> {
    /**
     * Fügt der Menge einen Eintrag hinzu.
     */
    void add(T item);

    /**
     * Liefert die Gesamtmenge.
     */
    Set<T> get();

    /**
     * Fügt diese Instanz des CRDTs mit der übergebenen zusammen.
     */
    void merge(GrowOnlySet<T> other);
}
```

Grow-Only Set
// Implementierung

```
class GrowOnlySetImpl<T> implements GrowOnlySet<T> {
    private Set<T> content = new HashSet<>();

    public void add(T item) {
        this.content.add(item);
    }

    public Set<T> get() {
        return this.content;
    }

    public void merge(GrowOnlySetImpl<T> other) {
        this.content.addAll(other.content);
    }
}
```

Observed-Remove-Set
// Motivation

- Manchmal will man einer Menge nicht nur Dinge hinzufügen …
- … sondern auch löschen …
- … und ggf. später wieder hinzufügen.

Observed-Remove-Set
// Schnittstelle

```
interface ORSet<T> {

    void add(T item);

    void remove(T item);

    boolean contains(T item);

    Set<T> get();

    void merge(ORSet<T> other);
}
```

Observed-Remove-Set
// Implementierung – add

```
class ORSetImpl<T> implements ORSet<T> {
    private static final class OREntry {
        private Set<String> addIds = new HashSet<>();
        private Set<String> tombstones = new HashSet<>();
    }

    private Map<T, OREntry> entries = new HashMap<>();

    public void add(T item) {
        OREntry e = this.entries.get(item);
        if (e == null) {
            e = new OREntry();
            this.entries.put(item, e);
        }
        e.addIds.add(generateUUID());
    }

    ...
}
```

Idee: UUIDs für Einfügungen

Observed-Remove-Set
// Implementierung – remove

```
class ORSetImpl<T> implements ORSet<T> {
    private static final class OREntry {
        private Set<String> addIds = new HashSet<>();
        private Set<String> tombstones = new HashSet<>();
    }

    private Map<T, OREntry> entries = new HashMap<>();
    ...

    public void remove(T item) {
        OREntry e = this.entries.get(item);
        if (e != null) {
            e.tombstones.addAll(e.addIds);
        }
    }

    ...
}
```

Observed-Remove-Set
// Implementierung – contains

```
class ORSetImpl<T> implements ORSet<T> {
    private static final class OREntry {
        private Set<String> addIds = new HashSet<>();
        private Set<String> tombstones = new HashSet<>();
    }

    private Map<T, OREntry> entries = new HashMap<>();
    ...

    public boolean contains(T item) {
        OREntry e = this.entries.get(item);
        if (e == null) {
            return false;
        }
        return !e.addIds.equals(e.tombstones);
    }

    ...
}
```

Observed-Remove-Set
// Implementierung – get

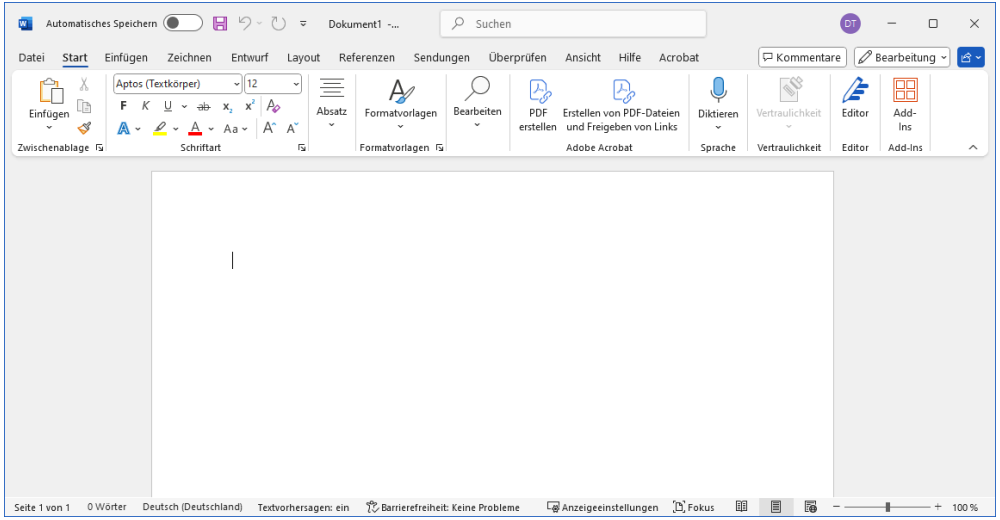
```
class ORSetImpl<T> implements ORSet<T> {
    private static final class OREntry {
        private Set<String> addIds = new HashSet<>();
        private Set<String> tombstones = new HashSet<>();
    }

    private Map<T, OREntry> entries = new HashMap<>();
    ...

    public Set<T> get() {
        return this.entries.keySet().stream()
            .filter(k -> this.contains(k))
            .collect(Collectors.toSet());
    }

    ...
}
```

Observed-Remove-Set
// Implementierung – merge

```
class ORSetImpl<T> implements ORSet<T> {
    private static final class OREntry {
        private Set<String> addIds = new HashSet<>();
        private Set<String> tombstones = new HashSet<>();
    }

    private Map<T, OREntry> entries = new HashMap<>();
    ...

    public void merge(ORSetImpl<T> other) {
        for (Entry<T, OREntry> e : other.entries.entrySet()) {
            OREntry myEntry = this.entries.get(e.getKey());
            if (myEntry == null) {
                this.entries.put(copyOf(e.getValue()));
            } else {
                myEntry.addIds.addAll(e.getValue().addIds);
                myEntry.tombstones.addAll(e.getValue().tombstones);
            }
        }
    }
}
```

YATA
// Motivation

- Anwendungsfall: Ein Text soll gemeinsam bearbeitet werden

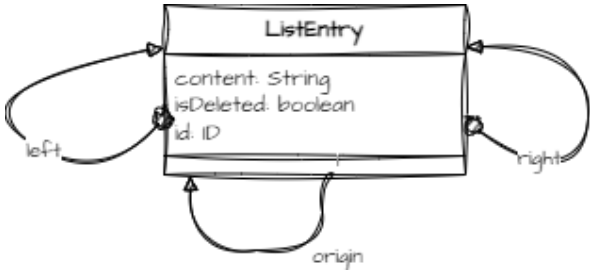


YATA
// Schnittstelle

```
interface YATA {

    void insert(int index, String text);

    void remove(int index, int count);

    String get();

    void merge(YATA other);
}
```

YATA
// Grundideen

- Grundideen von YATA (und vielen anderen Text-CRDTs)
  - Jeder Buchstabe (!) im Dokument wird über eine eindeutige ID (Replika-ID + logische Uhr) identifiziert
  - Intern wird nicht mit Indizes, sondern mit diesen IDs gearbeitet
  - Speicherung erfolgt als doppelt verkettete Liste
  - Konfliktauflösung passiert zunächst über Vergleich der Einfügepositionen und falls das nicht reicht über Vergleich der IDs
  - Löschungen löschen nicht wirklich, sondern markieren nur als gelöscht (Tombstones)
  - Einige Tricks (z.B. Kompression/Zusammenfassung der IDs), damit es effizient bleibt



Agenda

1. Was sind CRDTs und weshalb sind sie nützlich?
2. Konkrete Beispiele für CRDTs
3. **Ein CRDT in „echter Software“**
    1. Was ist Föderation für Dokumentenportale und was hat das mit CRDTs zu tun?
    2. Encrypted OR-Set

Hintergrund: Browserbasierte Föderation für Dokumentenportale // Wofür braucht man das?

Elektronische Zustellung von Dokumenten geschieht aktuell hauptsächlich auf 3 Wegen:

- E-Mail
- Anbindung eines zentralen, externen Dokumentenportal-Anbieters
- Entwicklung/ Erweiterung eines eigenen Portals

E-Mail: Häufig keine Option

Unverschlüsselt
➔ 77 - 96 % der Bevölkerung wollen nicht, dass ihr E-Mail-Provider Zugriff auf Ihre Bank- und Versicherungsdokumente hat

Keine Zustellgarantie
➔ 47 - 74 % der Bevölkerung hatten schon mindestens einmal Probleme mit verschollenen E-Mails

Spam / Information Overload
➔ 62 - 87 % der Bevölkerung haben schon mindestens einmal eine wichtige E-Mail übersehen

* Quelle der Statistiken: SDP-2-Studie

Einheitlichkeit

Zentraler Zugang

Kein Betriebsaufwand

Anbindung eines zentralen, externen Dokumentenportal-Anbieters

Eine zentrale Instanz verwaltet/ kontrolliert alle Dokumente

„Imperium“

Große Macht für

Single Point of

Single Point of Failure

Betreiber behält volle Kontrolle über Daten und

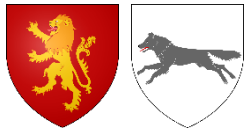
Flexibilität für Betreiber

Kein Vertrauen in Dritte nötig

Entwicklung/ Erweiterung eines eigenen Portals

Nebeneinander unabhängiger Systeme

„Kleinstaaterei“



Unbequemer Zugriff (viele

Wildwuchs

Fehlende Synergien

Föderation to the Rescue!

E-Mail

„Imperium“

„Kleinstaaterei“

Föderation!

Föderation, was ist das?

Science-Fiction

„**Die Vereinte Föderation der Planeten** […] ist eine große interstellare Allianz […]. Die einzelnen Mitglieder der Föderation haben sich unter […] gemeinsamen Zielen und Vorstellungen […] zusammengeschlossen."
(Quelle: fandom.com)

Politik

„Der Begriff **Konföderation** bezieht sich auf einen vertraglichen Zusammenschluss selbständiger Einheiten, die nach außen hin gemeinsam auftreten, ihre Souveränität aber beibehalten."
(Quelle: Wikipedia)

IT

„Ein Datenbankmanagementsystem ist ein **Föderiertes Informationssystem**, wenn es Zugriff auf mehrere andere autonome Informationsquellen ermöglicht, ohne dass deren Daten kopiert werden."
(Quelle: Wikipedia)

Föderation in Bezug auf Dokumentenportale

Selbständige Einheiten …

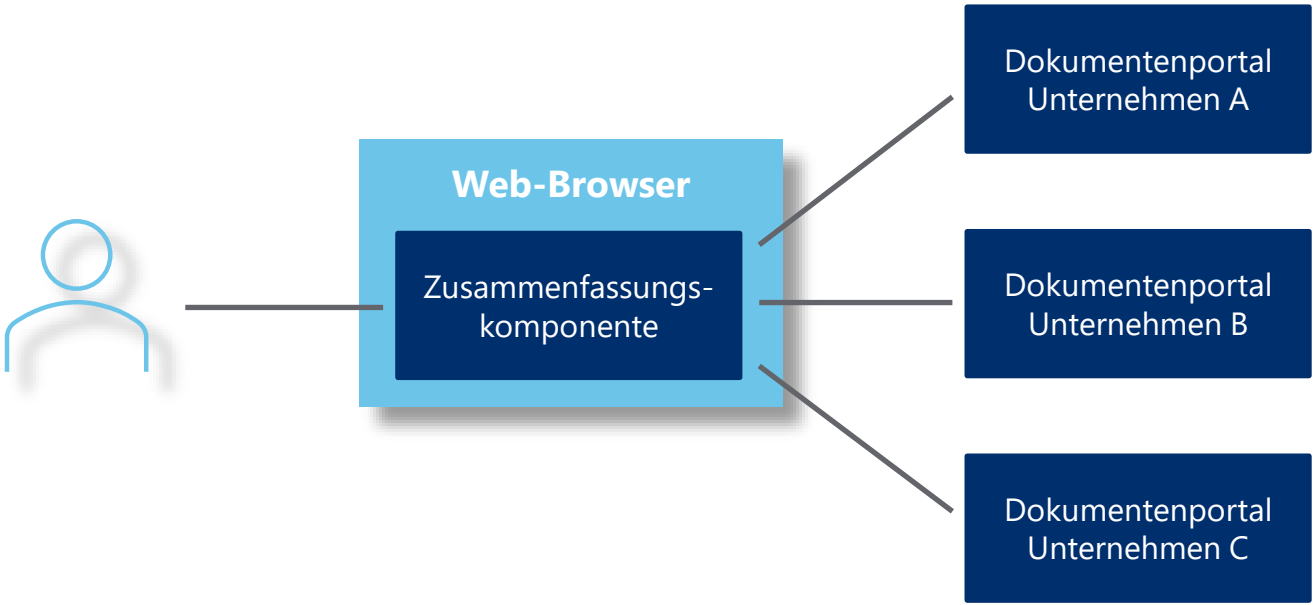
… die nach außen hin gemeinsam auftreten.

Unabhängig voneinander betriebene Dokumentenportale/Postboxen mit jeweils **eigener Datenhaltung** (wie bei „Kleinstaaterei“) …

… die für den Anwender unter einer **gemeinsamen Nutzerschnittstelle** verfügbar sind (wie bei „Imperium“).

Föderierte Identität/Login (z. B. OpenID) **+** Föderierte Postfach-Übersicht

Browser-basierte Föderation



Eigenständiges Dokumentenportal in einem Zusammenschluss kooperierender Systeme

„Föderation“

Betreiber behält volle Kontrolle über Daten

Betreiber behält Flexibilität

Kein Vertraue nötig

Synergi

Einfacher, einheitlicher Zugang

Gewisses Maß an Standardisierung und Zusammenarbeit nötig

Was hat browserbasierte Föderation mit CRDTs zu tun?

- Für die Föderation muss eine Liste der beteiligten Server auf den Endgeräten verwaltet werden
- Die Liste soll zwischen verschiedenen Geräten (z.B. Laptop und Handy) synchronisiert werden
  ➔ verteiltes System mit Partition Tolerance
- Wenn ich mein Handy verliere, soll es trotzdem möglich sein, die Liste wiederherzustellen
  ➔ Sicherung auf dem Server ist nötig
- Ein Föderationsmitglied soll nicht wissen, bei welchen anderen Föderationsmitgliedern ich einen Account habe
  ➔ Sicherung muss verschlüsselt erfolgen
- Ich will den Schlüssel nur angeben müssen, wenn das nötig ist
  ➔ bestimmte Operationen müssen ohne Schlüssel möglich sein

Encrypted OR-Set
// Schnittstelle

```
interface EORSet {
    // für Hinzufügen, Löschen, Lesen wird der Schlüssel benötigt

    void add(String item, Key key);

    void remove(String item, Key key);

    Set<String> getDecrypted(Key key);

    // für Transfer, Vergleich, Merge wird kein Schlüssel benötigt

    ORSet<byte[]> getEncrypted();

    void equals(EORSet other);

    void merge(EORSet other);
}
```

Encrypted OR-Set
// Implementierung (1/2)

```
class EORSetImpl implements EORSet {
    private ORSet<byte[]> encrypted = new ORSet<>();

    public void add(String item, Key key) {
        this.encrypted.add(encrypt(item, key));
    }

    public void remove(String item, Key key) {
        this.encrypted.remove(encrypt(item, key));
    }

    public Set<String> getDecrypted(Key key) {
        return this.encrypted.map(item -> decrypt(item, key));
    }
    ...
}
```

Encrypted OR-Set
// Implementierung (2/2)

```
class EORSetImpl implements EORSet {
    ...

    public ORSet<byte[]> getEncrypted() {
        return this.encrypted;
    }

    public boolean equals(EORSetImpl other) {
        return this.encrypted.equals(other.encrypted);
    }

    public void merge(EORSetImpl other) {
        this.encrypted.merge(other.encrypted);
    }
}
```

Abschluss-Exkurs: Für Mathematiker …

- Die positiven Eigenschaften eines CRDTs lassen sich formal beweisen

- Bedingungen
  1. Auf den Zuständen des CRDTs lässt sich eine partielle Ordnung definieren
  2. Die merge-Funktion berechnet die Vereinigung („join“) in einem Halbverband („semilattice“)
     1. Kommutativ
     2. Assoziativ
     3. Idempotent
  3. Alle update-Funktionen müssen monoton in Bezug auf die partielle Ordnung sein

Zusammenfassung

- CRDTs ermöglichen eine einfache Implementierung ausfallsicherer verteilter Systeme ohne zentralen Server mit „eventual consistency“
- Man muss mit gewissen Einschränkungen in der Semantik leben
- Aber es gibt CRDTs für diverse Anwendungsfälle
  - Zähler
  - Felder/„Register“
  - Mengen
  - Text/Listen
  - …
- Aus einfachen CRDTs lassen sich komplexere CRDTs bauen

- SET setzt einen CRDT im Produkt POSY-Postbox für ein Dokumentenportal mit Föderationsarchitektur ein

- Quellen / Links
  - https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type
  - https://crdt.tech/
  - https://mattweidner.com/
  - https://yjs.dev/
  - https://vlcn.io/
  - https://github.com/netopyr/wurmloch-crdt
  - https://set.de/postbox