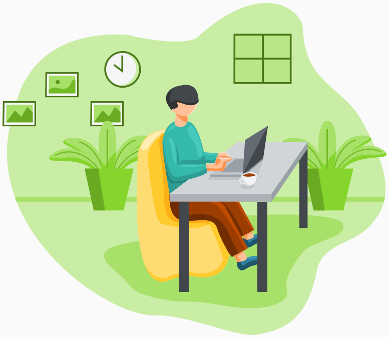
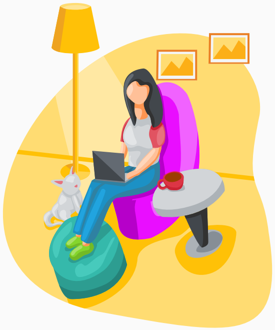
# Synql: A CRDT-based Approach for Replicated Relational Databases with Integrity Constraints

Victorien Elvinger, Claudia-Lavinia Ignat, Habibatou Ba
Inria Nancy, France

June 2024

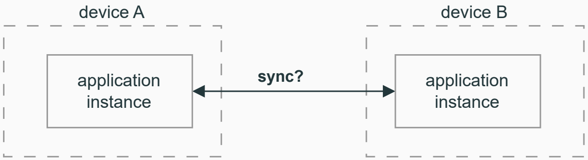
## Adding collaborative features to applications





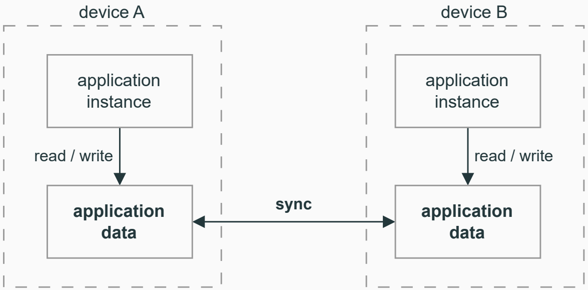
- several persons **modify together** a shared content
  - located at **different places**
  - **simultaneous** modifications or at **distinct time**
- adding collaborative features to applications is hard
  - **sequential** → **concurrent** modifications
  - **offline support**

## Adding collaborative features to applications



- replicate the application $\implies$ dedicated development

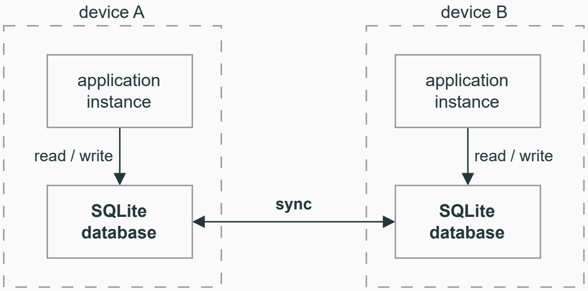
## Adding collaborative features to applications



- replicate the application $\Longrightarrow$ dedicated development
- **replicate the application data**[a]

[a] Kleppmann et al., “Local-first software: you own your data, in spite of the cloud”, *2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward!, Athens, Greece*, 2019.

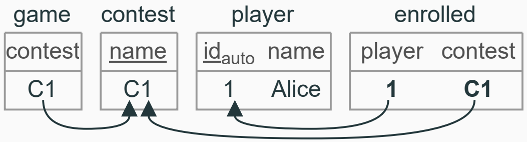
## Adding collaborative features to applications



- replicate the application $\implies$ dedicated development
- **replicate the application data**[a]
- SQLite is embedded in many applications

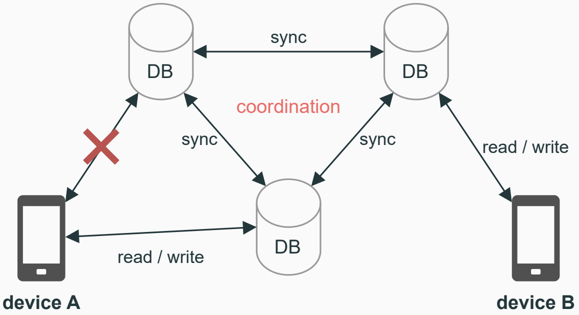
[a] Kleppmann et al., “Local-first software: you own your data, in spite of the cloud”, *2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward!, Athens, Greece*, 2019.

## Referential integrity



- ensure that the **target of a reference exists**
- the deletion of a reference target can result in
    - the **abortion of the deletion**
    - the **propagation of the deletion** to the reference source

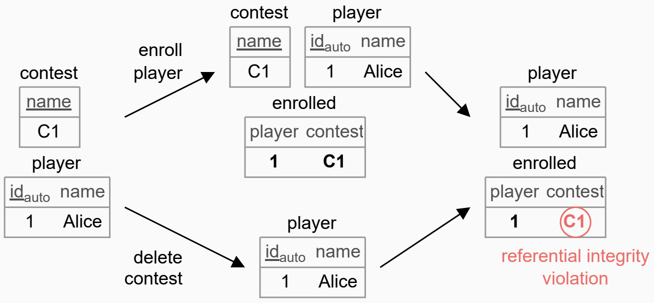
## Replicating relational databases: already done?



- **client-server** architecture
- **coordination** to maintain **data integrity**[a]

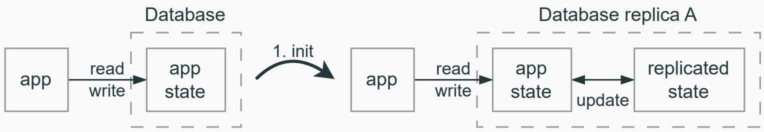
[a] Bailis et al., “Highly Available Transactions: Virtues and Limitations”, 2013.

## Referential integrity in face of coordination-less concurrencies



- concurrent deletion and referencing of a row

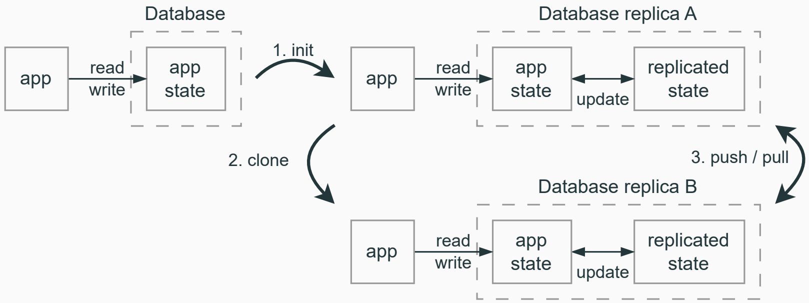
## Coordination-less replication of relational databases



- Git-like **coordination-less** replication of relational databases[a]
- can **break data integrity and user intent**
- **not Strongly Convergent**

[a] Yu et al., "Conflict-Free Replicated Relations for Multi-Synchronous Database Management at Edge", *IEEE International Conference on Smart Data Services SMDS, Beijing, China*, 2020.

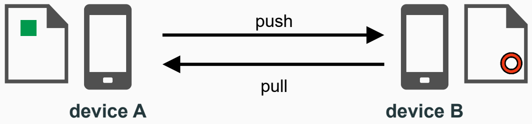
## Coordination-less replication of relational databases



- Git-like **coordination-less** replication of relational databases[a]
- can **break data integrity and user intent**
- **not Strongly Convergent**

[a] Yu et al., "Conflict-Free Replicated Relations for Multi-Synchronous Database Management at Edge", *IEEE International Conference on Smart Data Services SMDS, Beijing, China*, 2020.

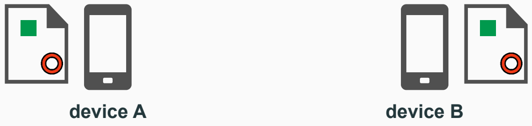
## Strong convergence



- property enforced by CRDTs[a]
- advantages:
    - low latency
    - no flickering

[a] Shapiro et al., “Conflict-Free Replicated Data Types”, *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium SSS, Grenoble, France*, 2011.

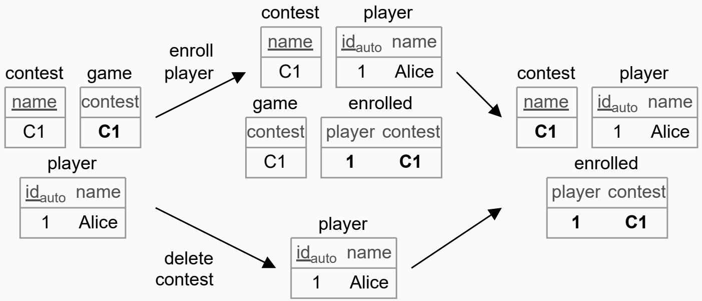
## Strong convergence



- property enforced by CRDTs[a]
- advantages:
  - low latency
  - no flickering

[a] Shapiro et al., “Conflict-Free Replicated Data Types”, *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium SSS, Grenoble, France*, 2011.

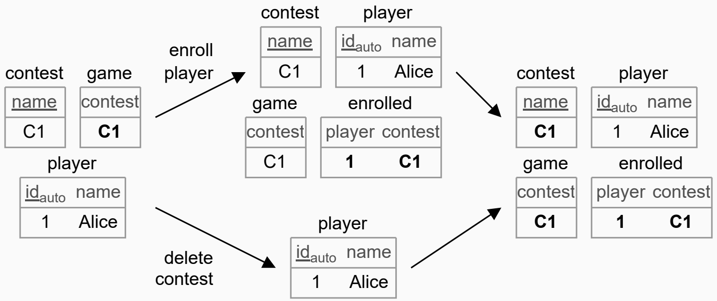
## Referential integrity maintenance - state of the art



- writes are compensated[a] in order to ensure integrity
- the *contest* is restored
- however, the *game* is not restored

[a] Balegas et al., “IPA: Invariant-preserving Applications for Weakly-consistent Replicated Databases”, 2018.

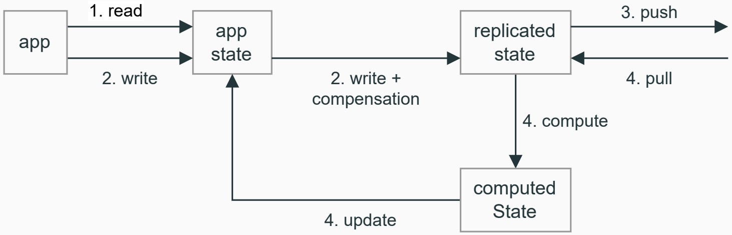
## Referential integrity maintenance - desired output



- the *game* should be restored

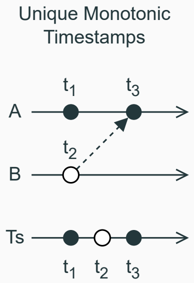
Can we replicate a relational database without any coordination that enforces Strong Convergence and maintains data integrity?

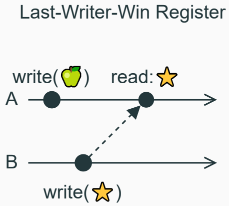
## Architecture overview

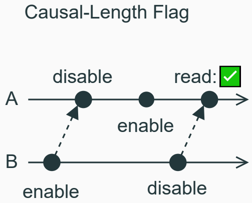


- app read without overhead
- an app write triggers replicated state update
- push / pull in background
- a pull merges the received state and computes app state

## Replicated state: composing CRDTs



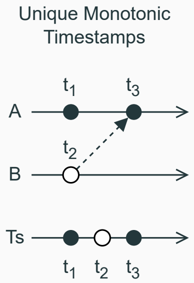


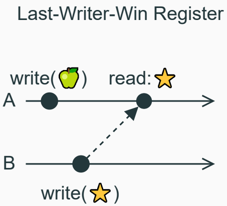


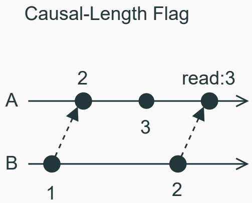
- globally unique and monotonic timestamps
  - monotonic: greater than previously observed timestamps
- Last-Writer-Win (LWW) Register[a] **keeps the newest value**
- state of CLFlag computed from the **longest chain**

[a] Johnson et al., “Maintenance of duplicate databases”, 1975.

## Replicated state: composing CRDTs



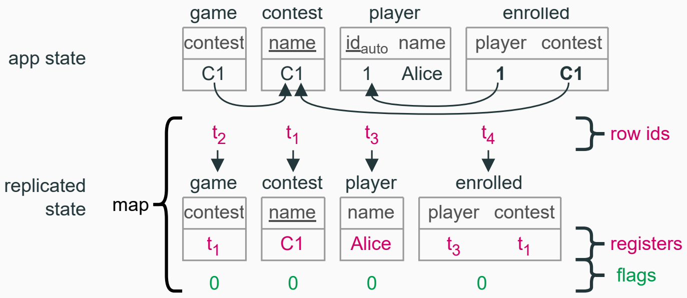




- globally unique and monotonic timestamps
  - monotonic: greater than previously observed timestamps
- Last-Writer-Win (LWW) Register[a] **keeps the newest value**
- state of CLFlag computed from the **longest chain**

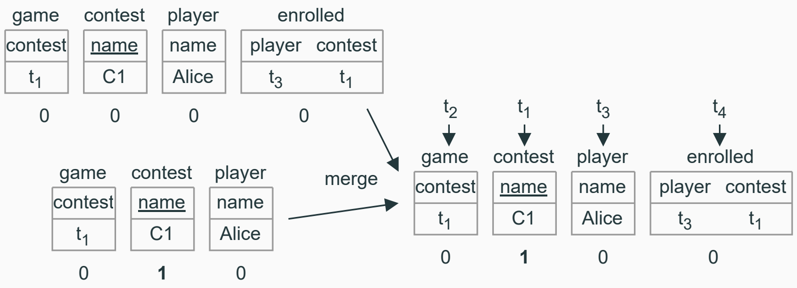
[a] Johnson et al., "Maintenance of duplicate databases", 1975.

## Replicated state: composing CRDTs



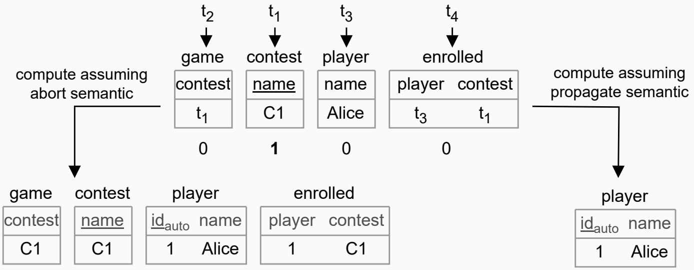
- **timestamps as row identifiers**
- a CL-Flag indicates if a row is removed
- a replicated attribute is a LWW-Register
- row identifiers as values of foreign keys

## Replicated state: composing CRDTs



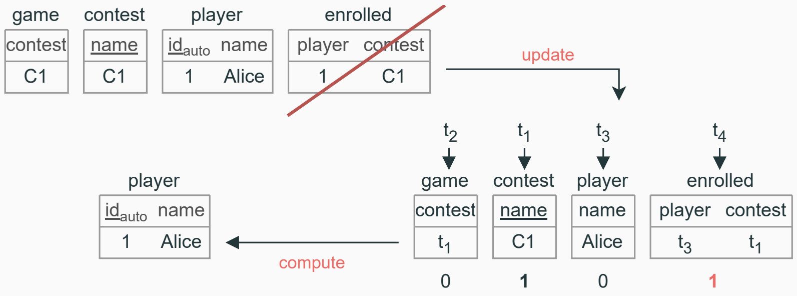
- the replicated state encodes only the app write

## Compute app state from replicated state



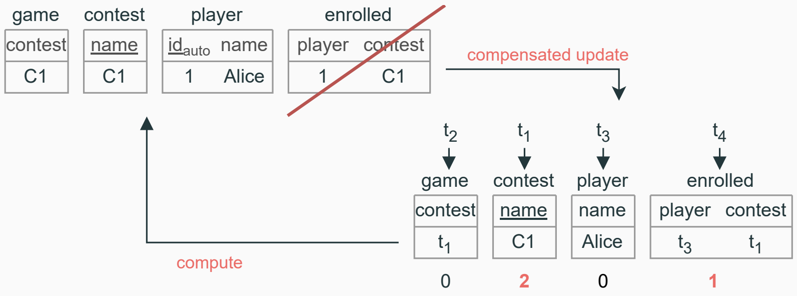
- app state is **derived fom the replicated state**
- leverage database schema for selecting **computation semantic**

## Compensation of app writes



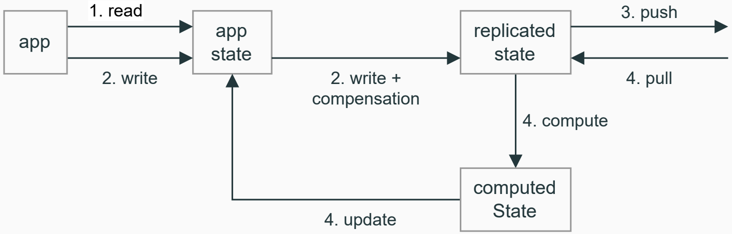
- state computation can result in surprising effect on app writes

## Compensation of app writes



- state computation can result in surprising effect on app writes
- **app writes must be compensated** for ensuring user intent

## Conclusions



- coordination-less replication of relational database
    - maintains data integrity
    - Strongly Convergent
- composition of CRDTs + state computation + compensations

FEEDBACK