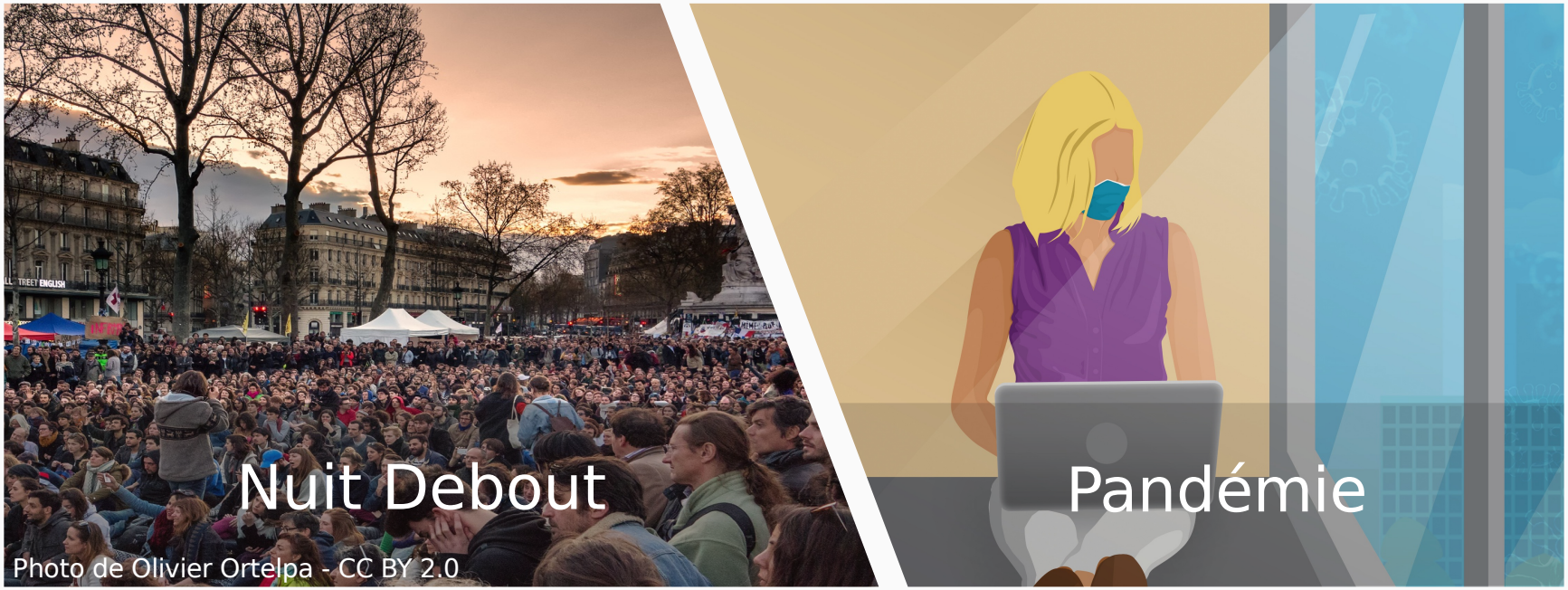
# Synql: Replicated Relational Databases and integrity maintenance

Victorien Elvinger

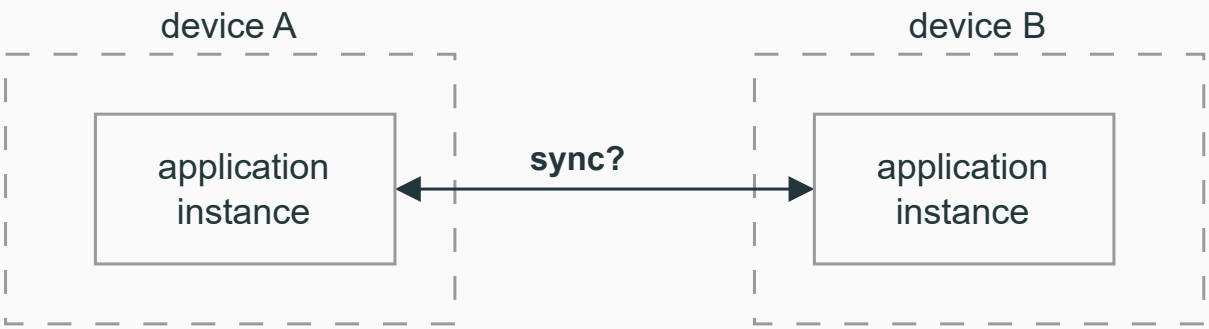

February 2023

# Collaborative applications



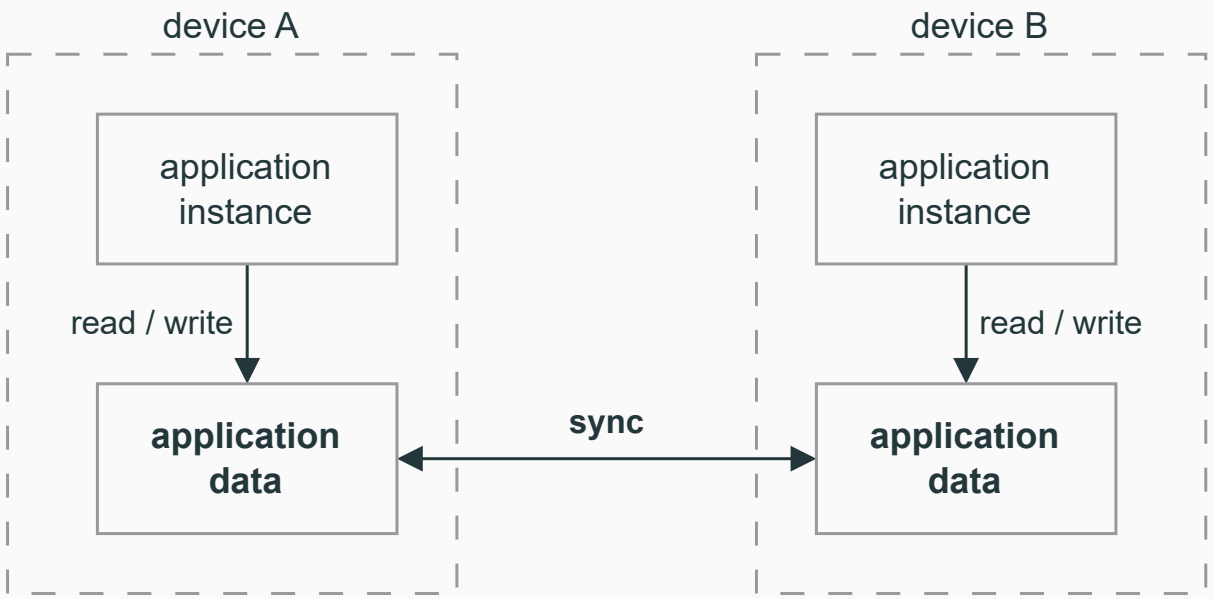

- several persons **modify together** a shared content
  - located at **different places**
  - **simultaneous** modifications or at **distinct time**
- adding collaborative features to applications is hard
  - **sequential** → **concurrent** modifications
  - **offline support**

# Adding collaborative features to applications




- replicate the application?
  - require dedicated development

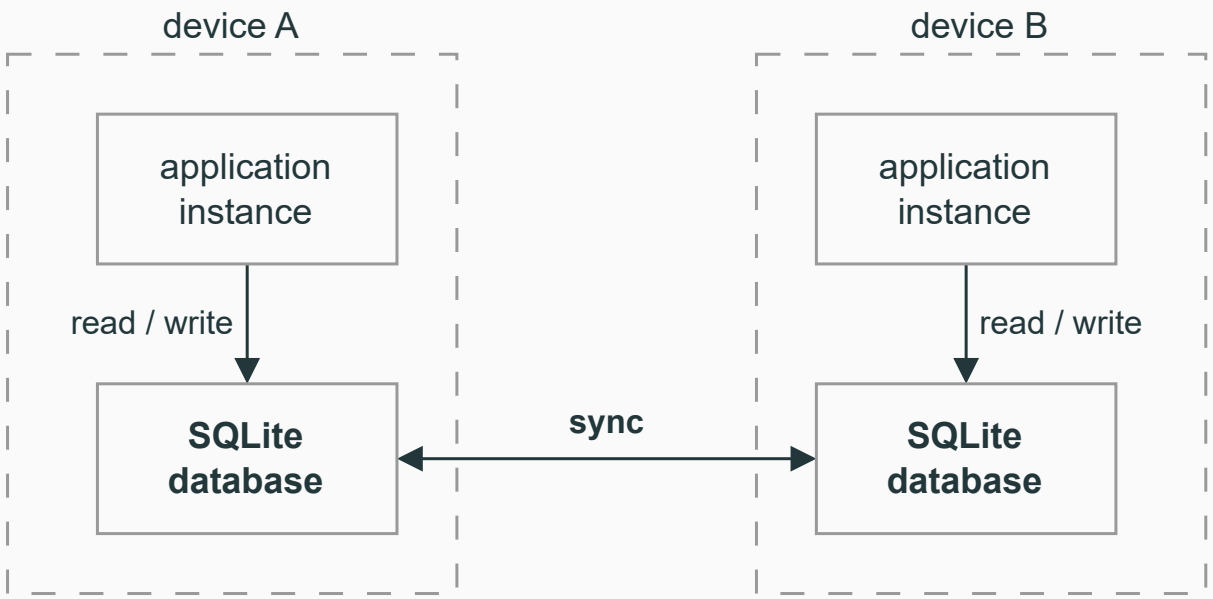
# Adding collaborative features to applications




- replicate the application?
  - require dedicated development
- **replicate the application data**[a]

[a] Kleppmann et al., “Local-first software: you own your data, in spite of the cloud”.

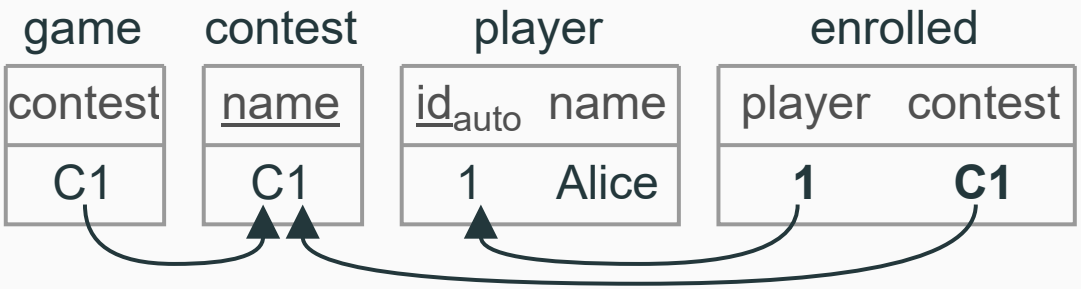
# Adding collaborative features to applications




- replicate the application?
  - require dedicated development
- **replicate the application data**[a]
- SQLite is embedded in many applications

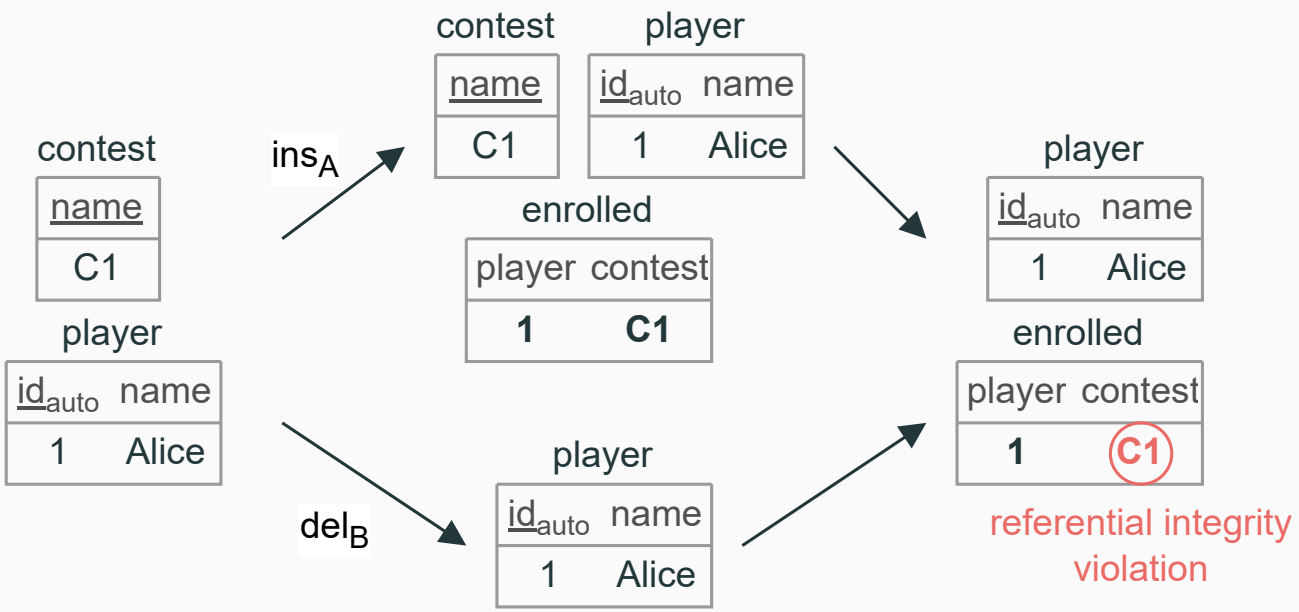
[a] Kleppmann et al., “Local-first software: you own your data, in spite of the cloud”.

# Referential integrity



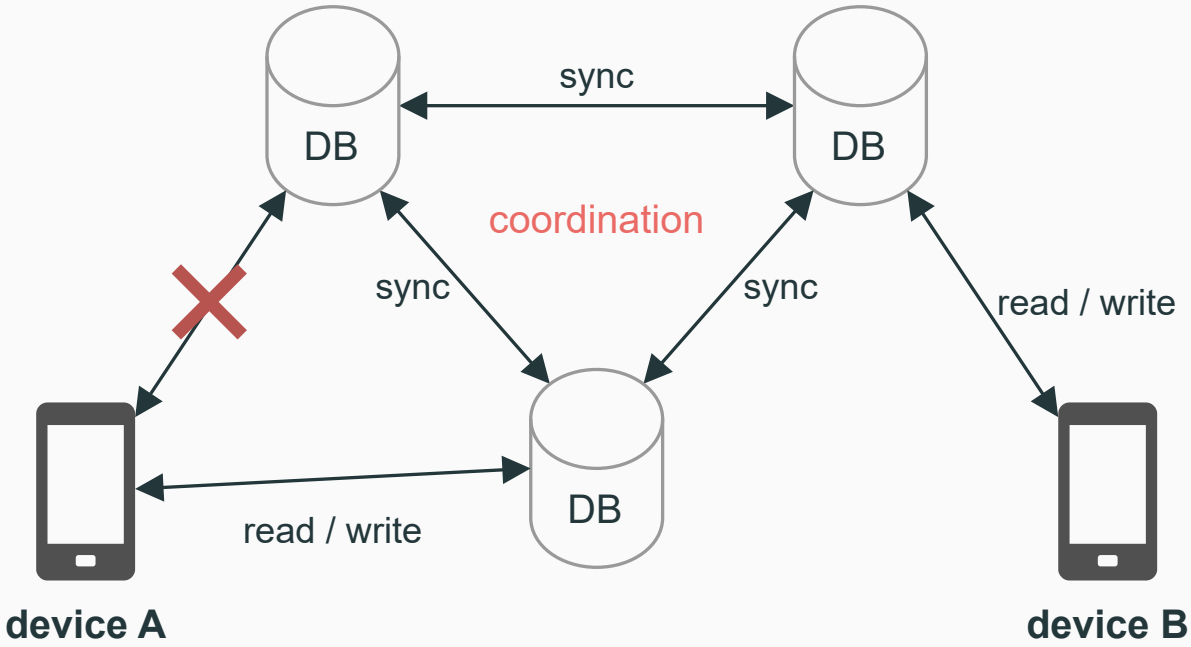

- ensure that the **target of a reference exists**
- the deletion of a target can result in
  - the **abortion of the deletion**
  - the **propagation of the deletion to its sources**

# Referential integrity in face of concurrencies




- concurrent deletion and referencing of a row

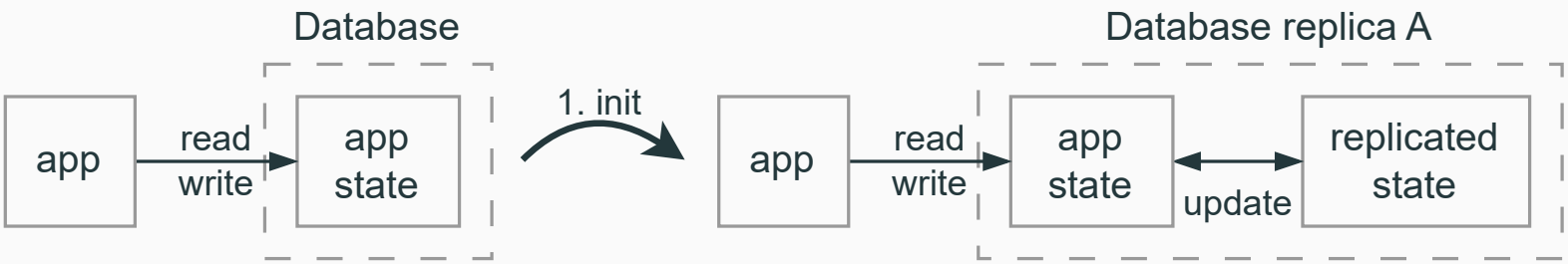
# Replicating relational databases: already done?




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

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

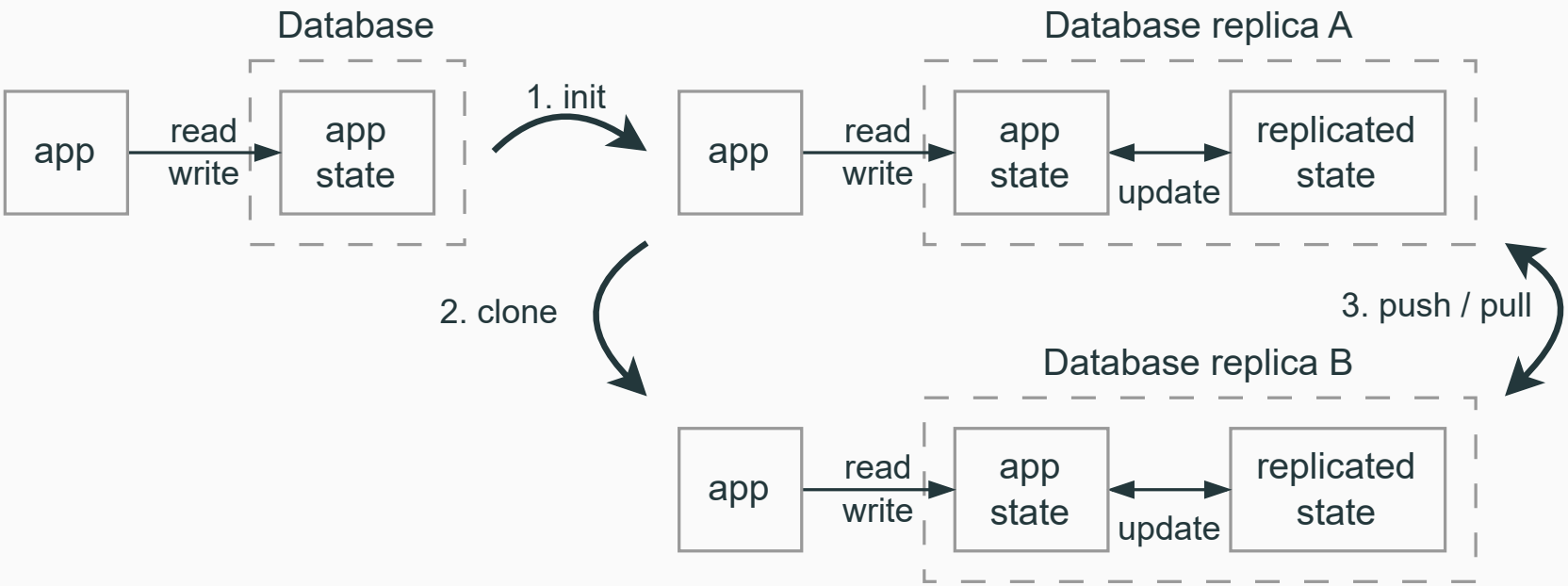
# Git-like model for replicating a database




- **coordination-less** replication of relational database[a]
  - based on Conflict-free Replicated Data Types (CRDTs)
- 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".

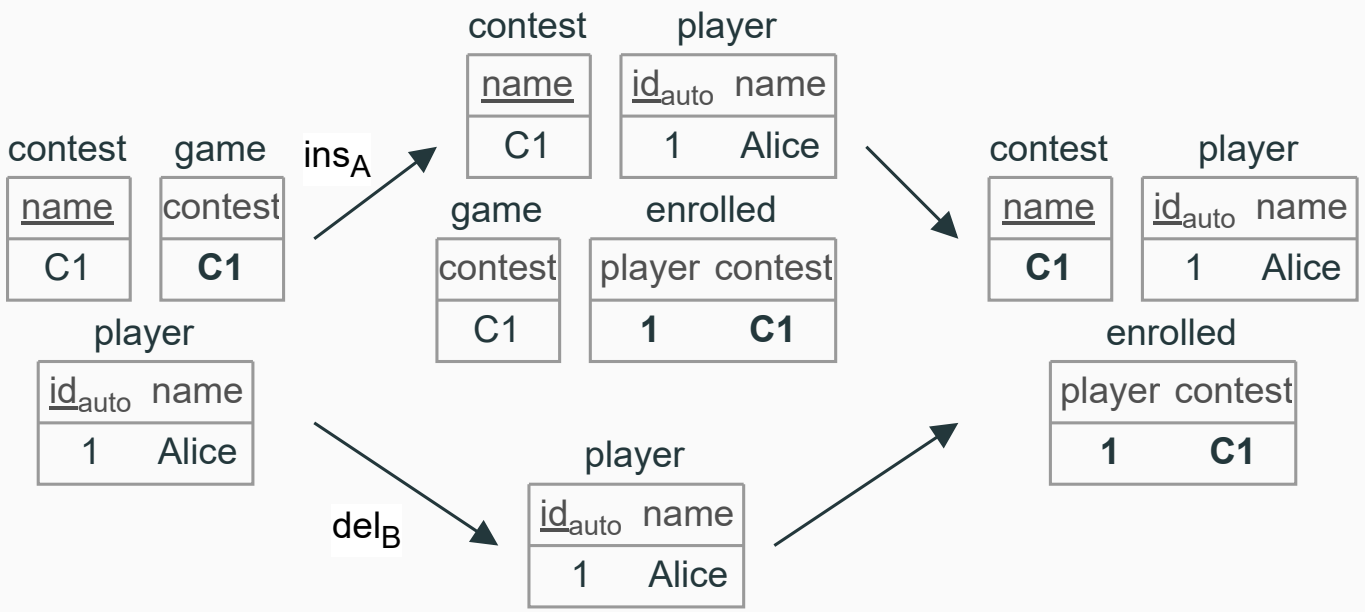
# Git-like model for replicating a database




- **coordination-less** replication of relational database[a]
  - based on Conflict-free Replicated Data Types (CRDTs)
- 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".

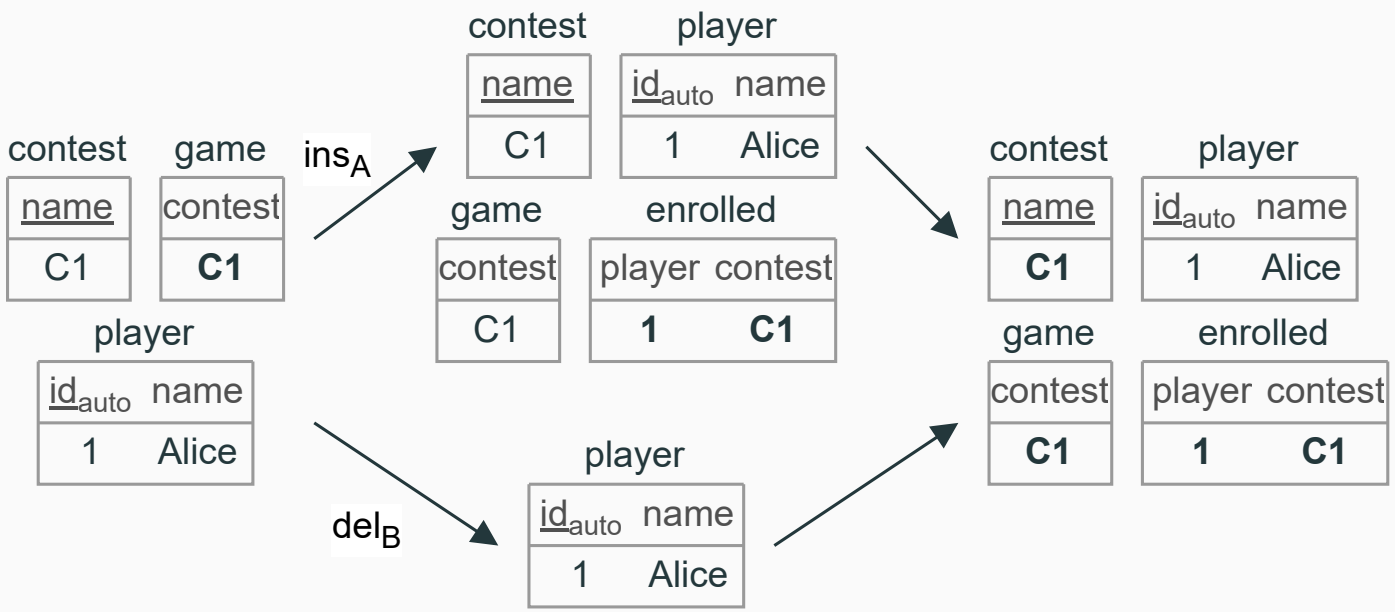
# Referential integrity maintenance - state of the art




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

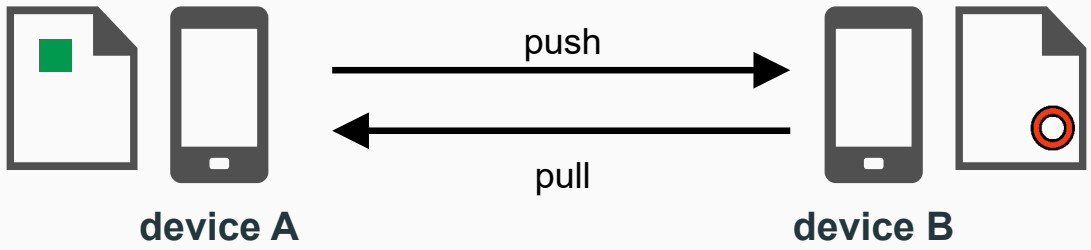
[a] Balegas et al., "IPA: Invariant-preserving Applications for Weakly-consistent Replicated Databases".

# Referential integrity maintenance - desired output




- the *game* should be restored

# Strong convergence




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

[a] Shapiro et al., “Conflict-Free Replicated Data Types”.

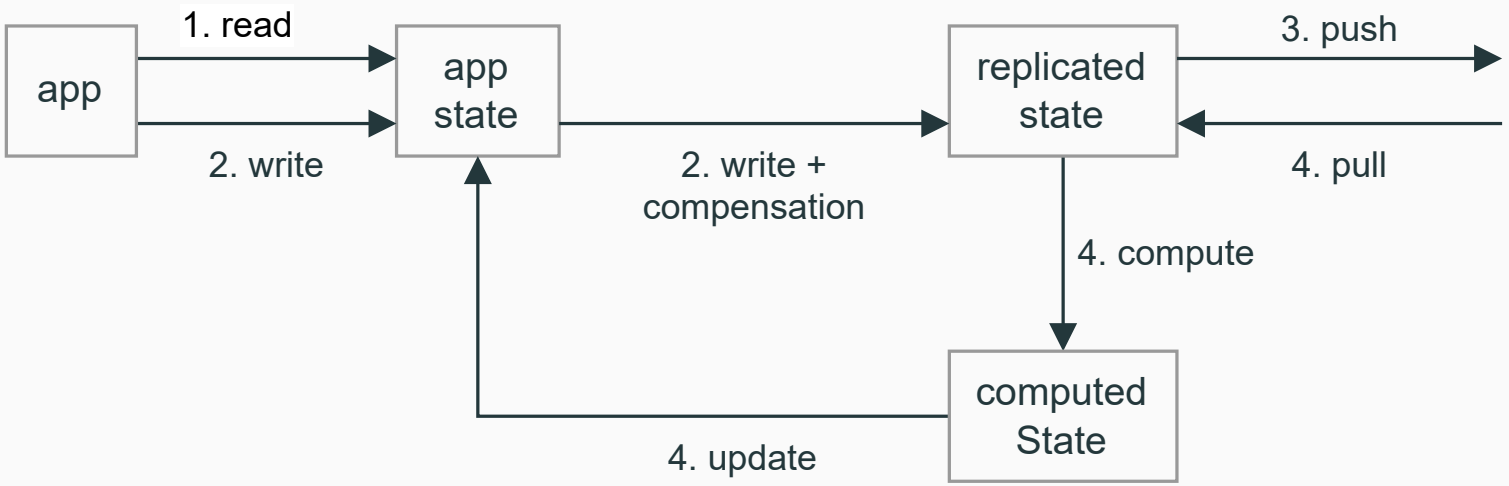
# Strong convergence


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

[a] Shapiro et al., “Conflict-Free Replicated Data Types”.

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**

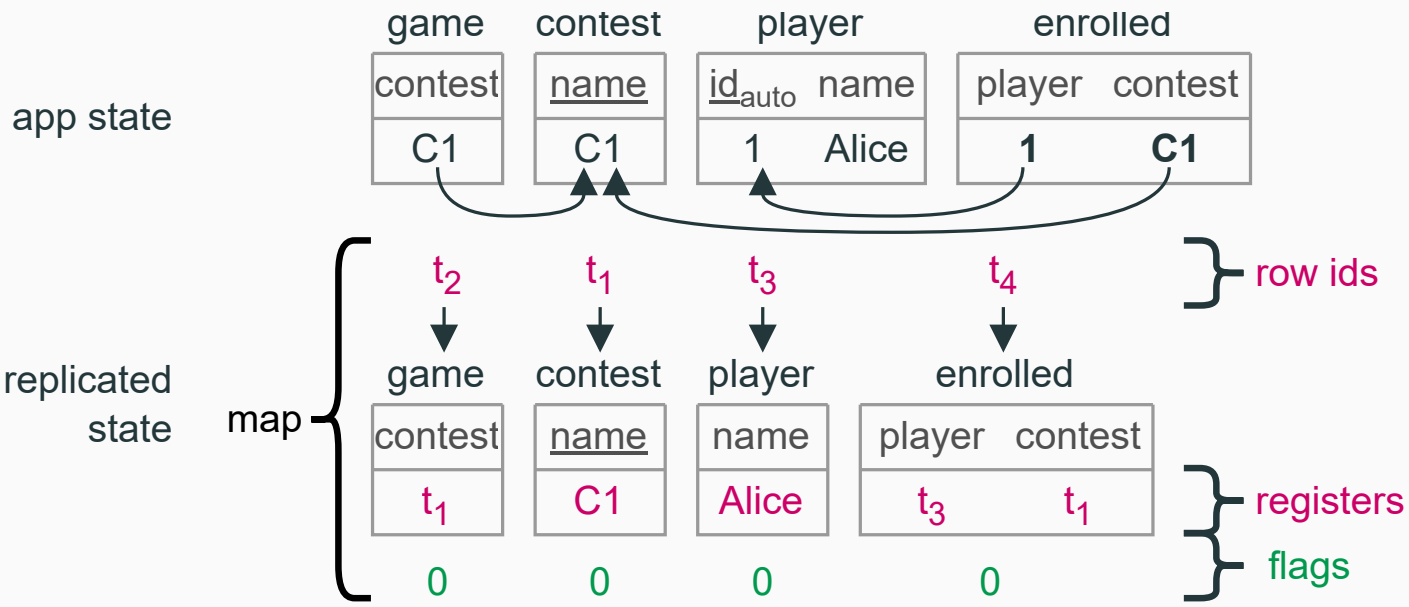
[a] Johnson et al., "Maintenance of duplicate databases".

# 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**

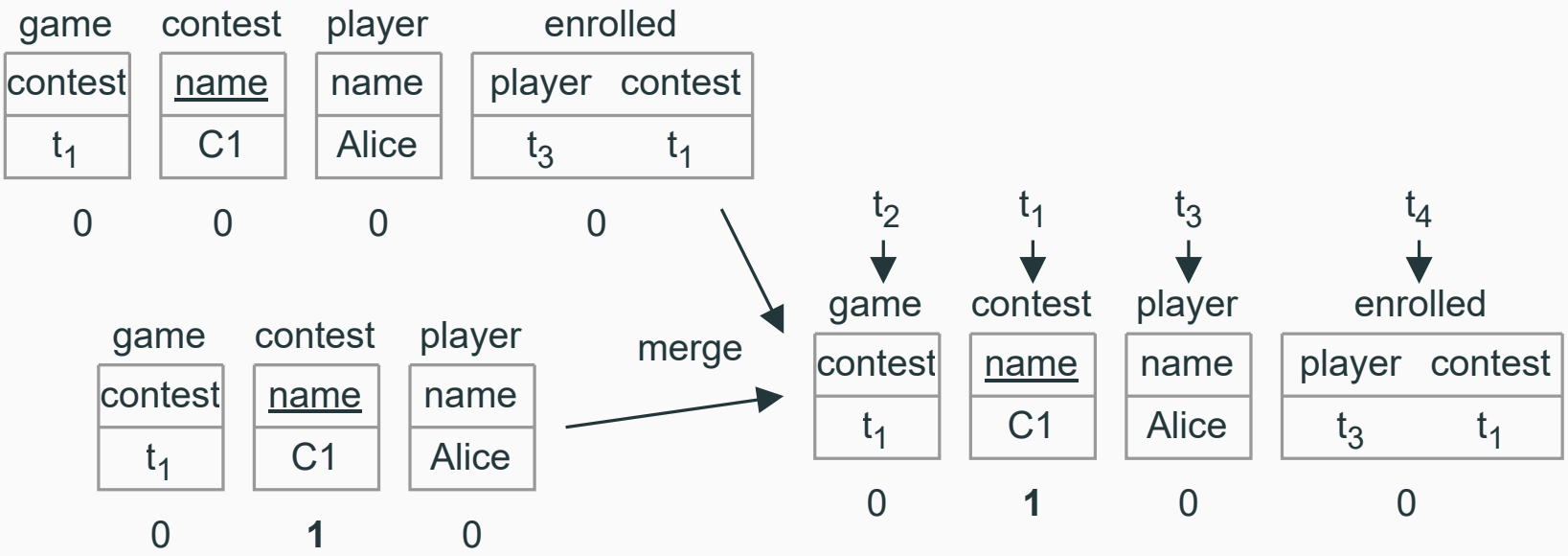
[a] Johnson et al., "Maintenance of duplicate databases".

# Replicated state: composing CRDTs



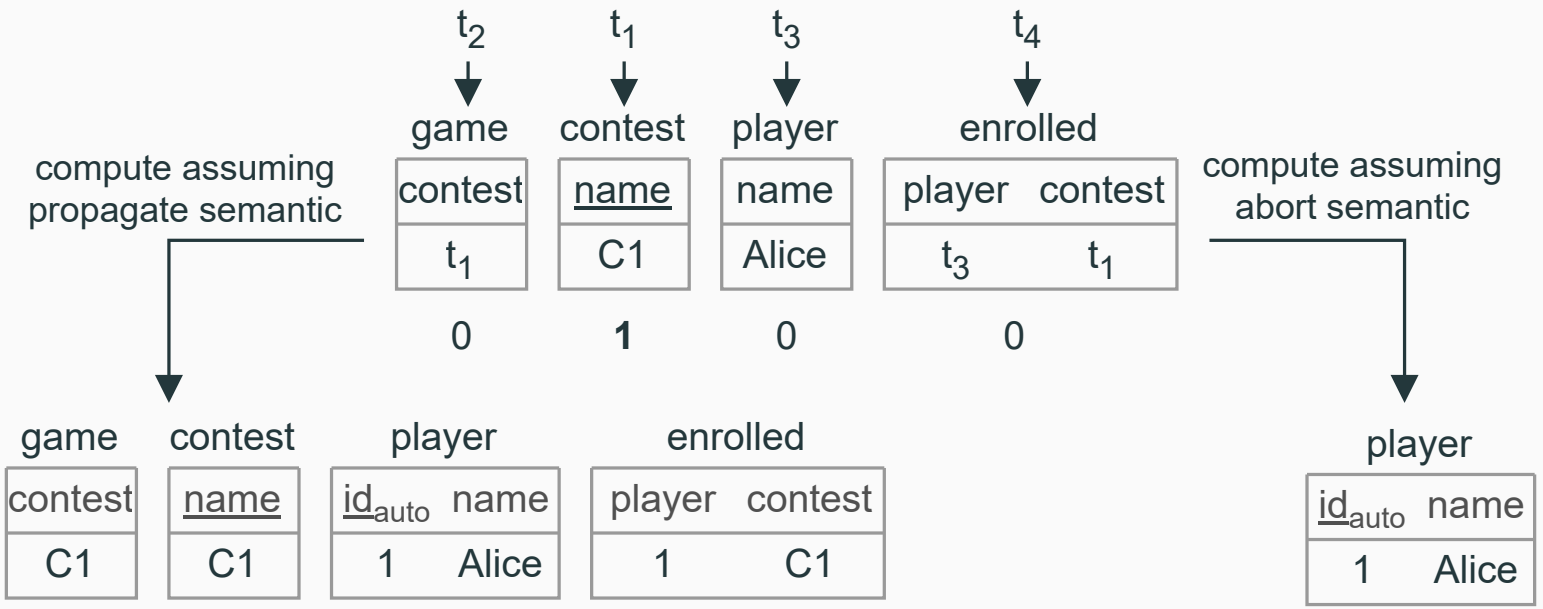

- **timestamps as row identifiers**
- a CL-Flag indicates if a row is removed
- a replicated attribute is a LWW-Register
- row identifiers as values of foreign keys

# Replicated state: composing CRDTs



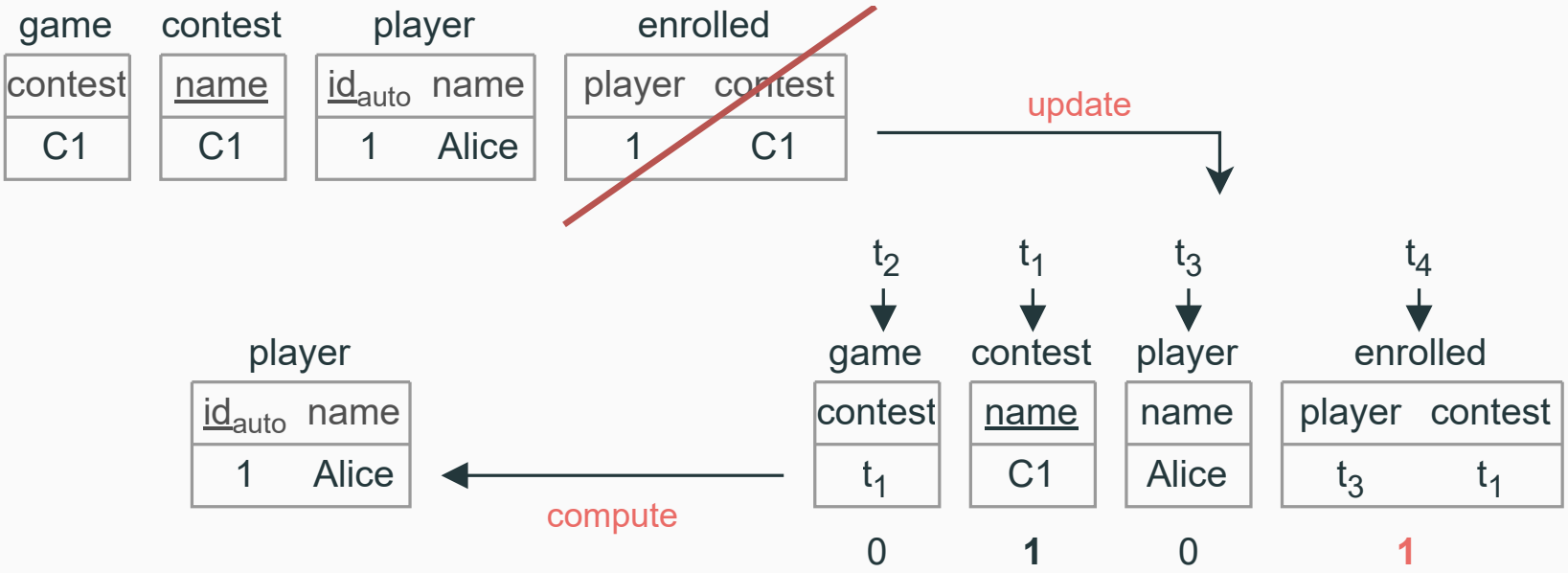

- the replicated state encodes only the app write

# Compute app state from replicated state



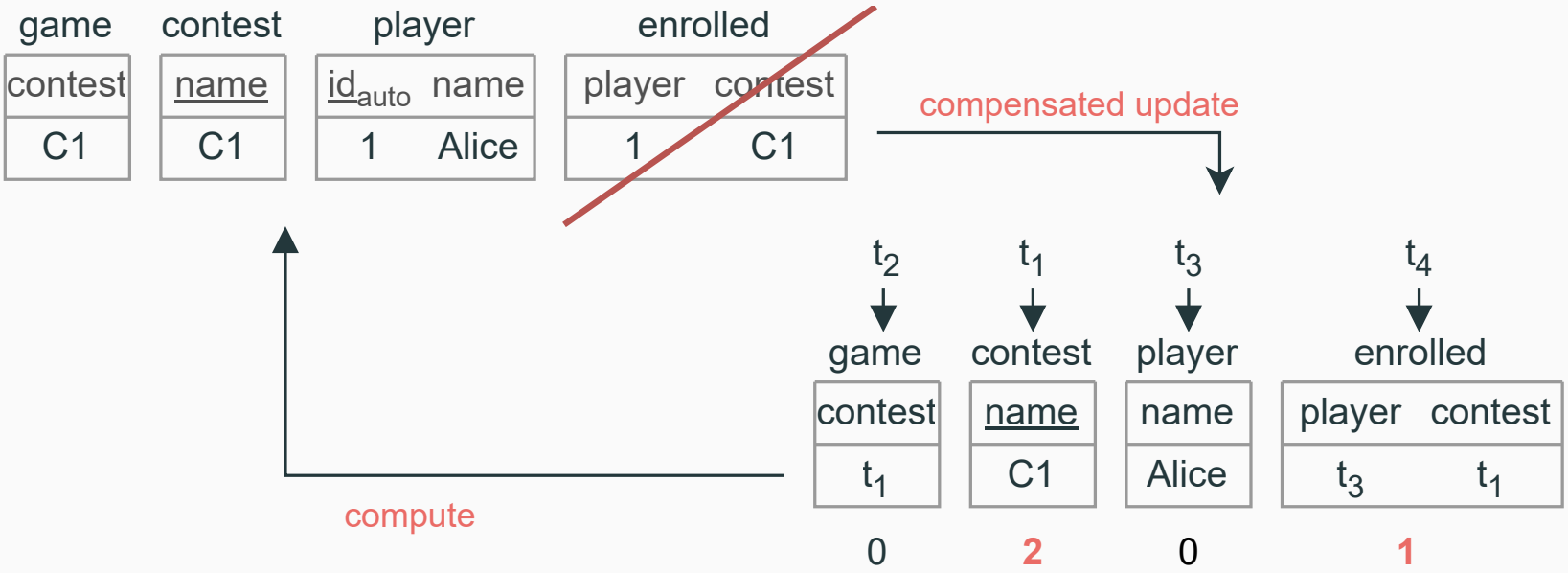

- app state is **derived fom the replicated state**
- leverage database schema for selecting **computation semantic**

# Compensation of app writes



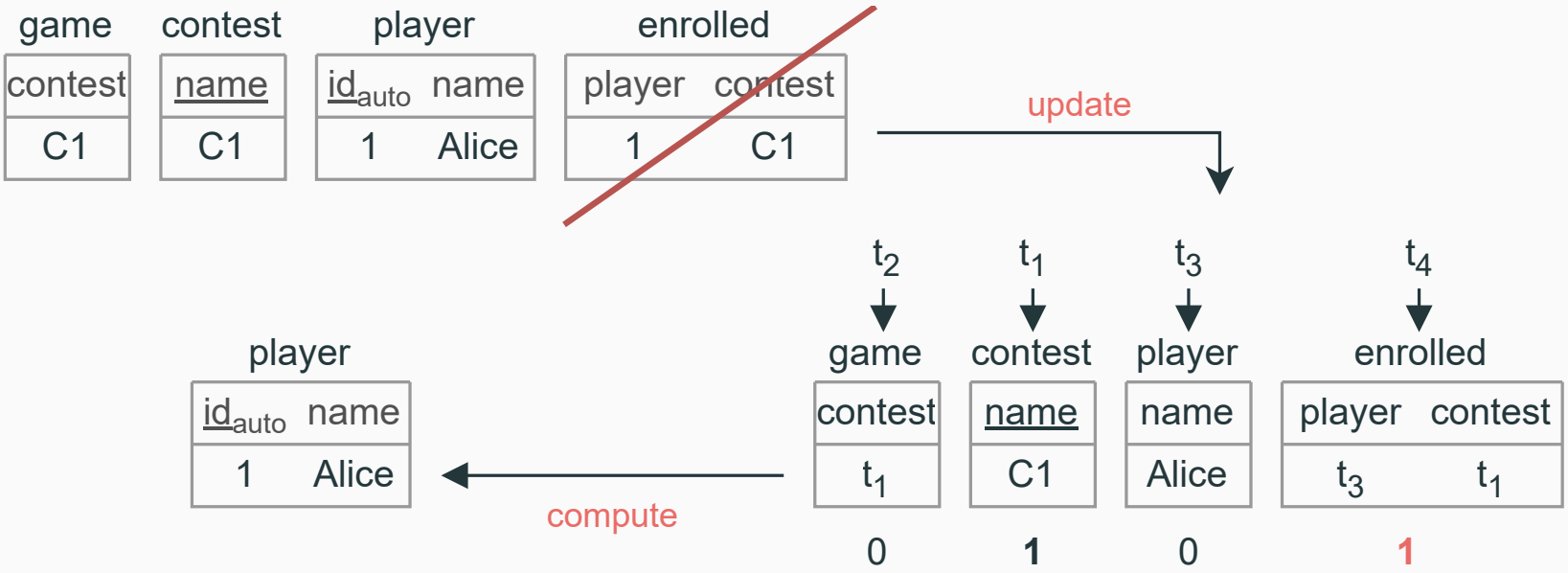

- state computation can result in surprising effect on app writes

# Compensation of app writes



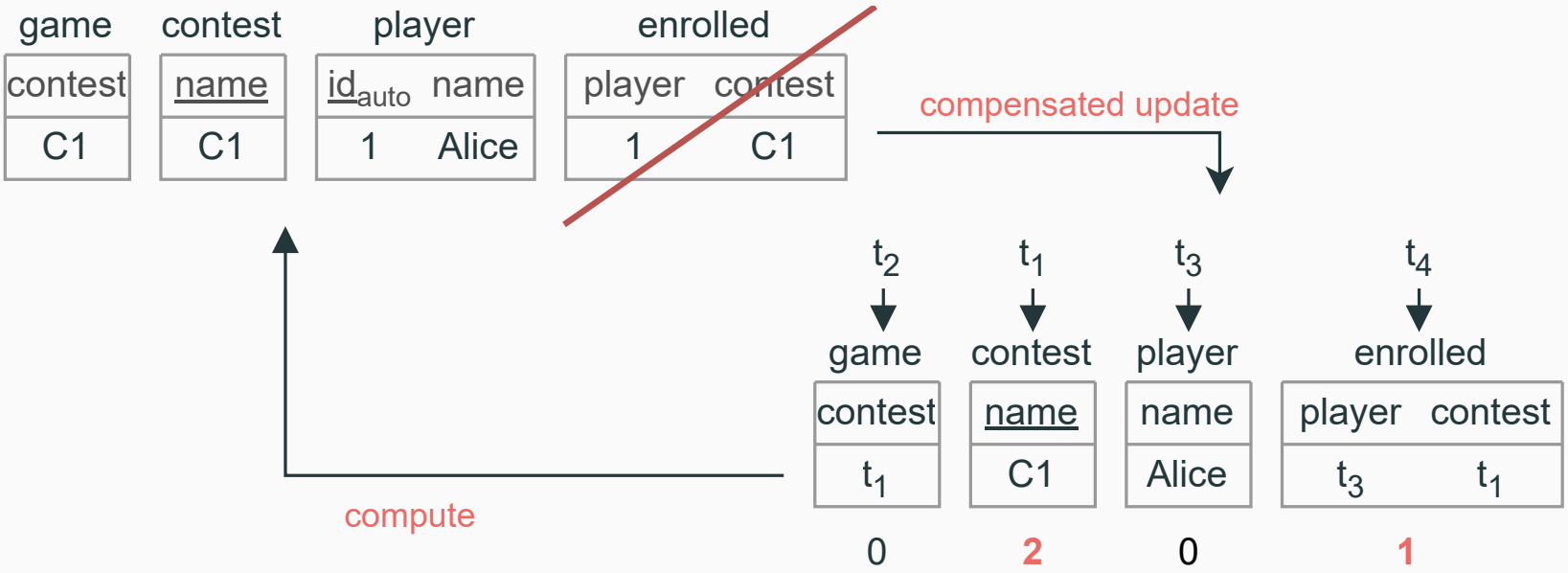

- state computation can result in surprising effect on app writes
- **app writes must be compensated** for ensuring user intent

# Compensation of app writes



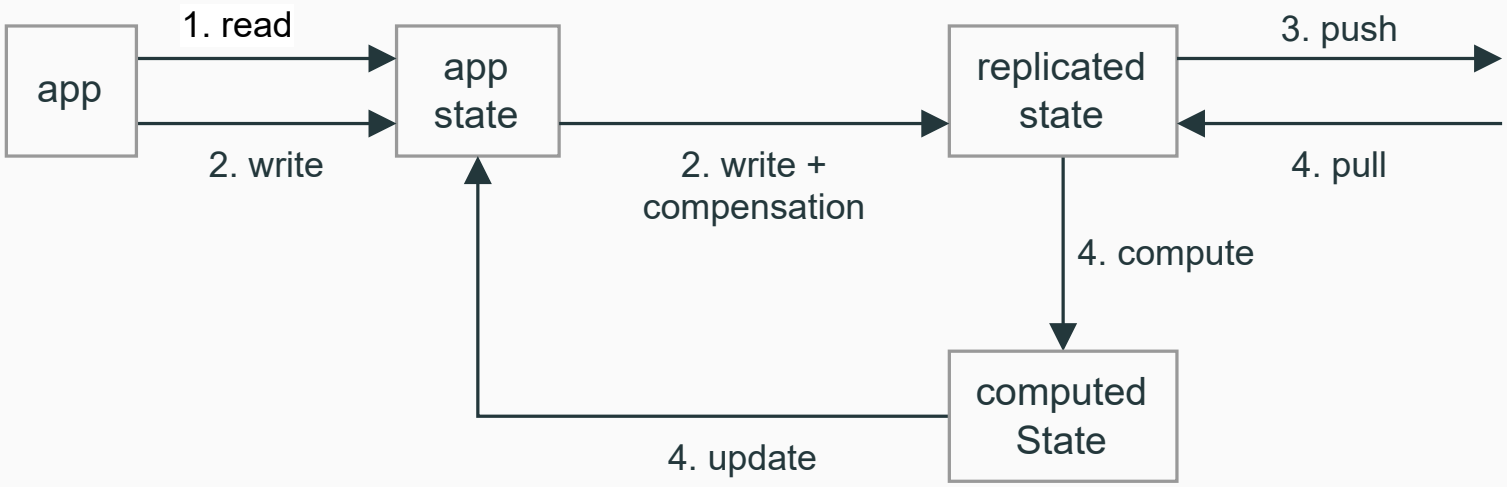

- state computation can result in surprising effect on app writes

# 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