SWIFTCLOUD LIMITATIONS

Valter Balegas (FCT/UNL) intern @ LIP6/INRIA

Objective

- Identify limitations in SwiftCloud
  - API support
  - System Design
- Case Study: TPC-W benchmark
  - Simulates an online book store
  - Transactional Operations
  - Traditionally implemented using relational databases

Database querying limitations (1)

- **Design:**
  - Simple database access with put/get identifier
- **Problems:**
  - How to apply query filters?
    - E.g. Retrieve all users called “John”
  - Fetch range of values
    - E.g. Retrieve 1000 orders
    - E.g. Retrieve the Most-Sold items

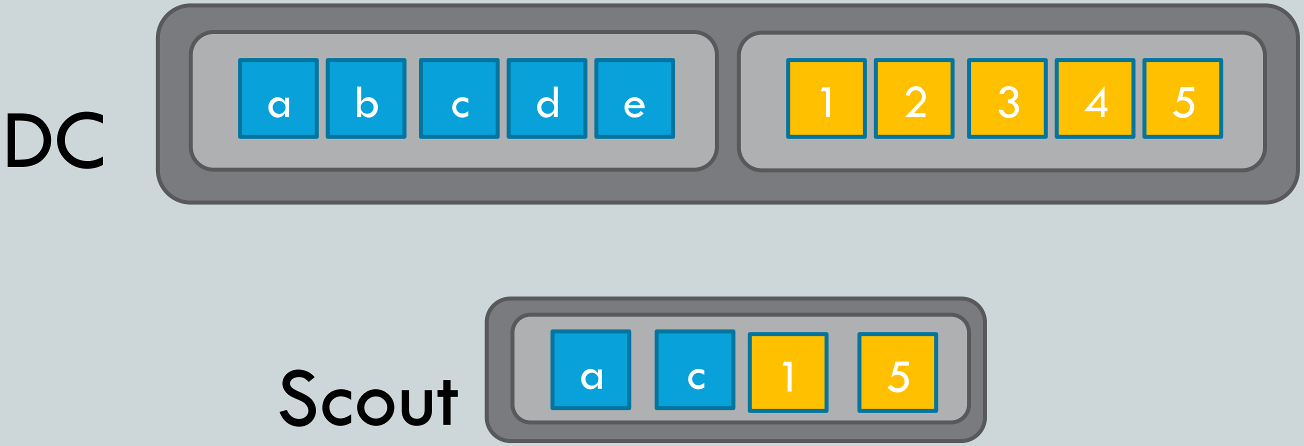
These queries require fetching all values and process them locally

Database querying limitations (2)

- **Workarounds:**
  - Maintain indexes
    - Programmer must be careful to update them
    - TOP-N CRDT
      - Abstracts the index but has to maintain all data
- **Solutions:**
  - Support server-side operations
    - Compute query results remotely

Cache control limitations (1)

- **Design:**
  - Scouts store a small portion of the database
  - Automatic caching on read operations
  - Programmer subscribe updates to maintain cache fresh



Cache control limitations (2)

- Accessing the cache
  - **Problems:**
    - No locality awareness
    - Range queries overflow the cache
  - **Solutions:**
    - Allow the programmer to decide what values are cached
    - Blind updates – execute update over objects without fetching them

Cache control limitations (3)

- Maintaining the cache
  - **Problems:**
    - Values frequently updated generate too many updates
    - High amount of update subscriptions impose great overhead
  - **Solutions:**
    - Compress updates on server side
      - ☹ more work on the data-center

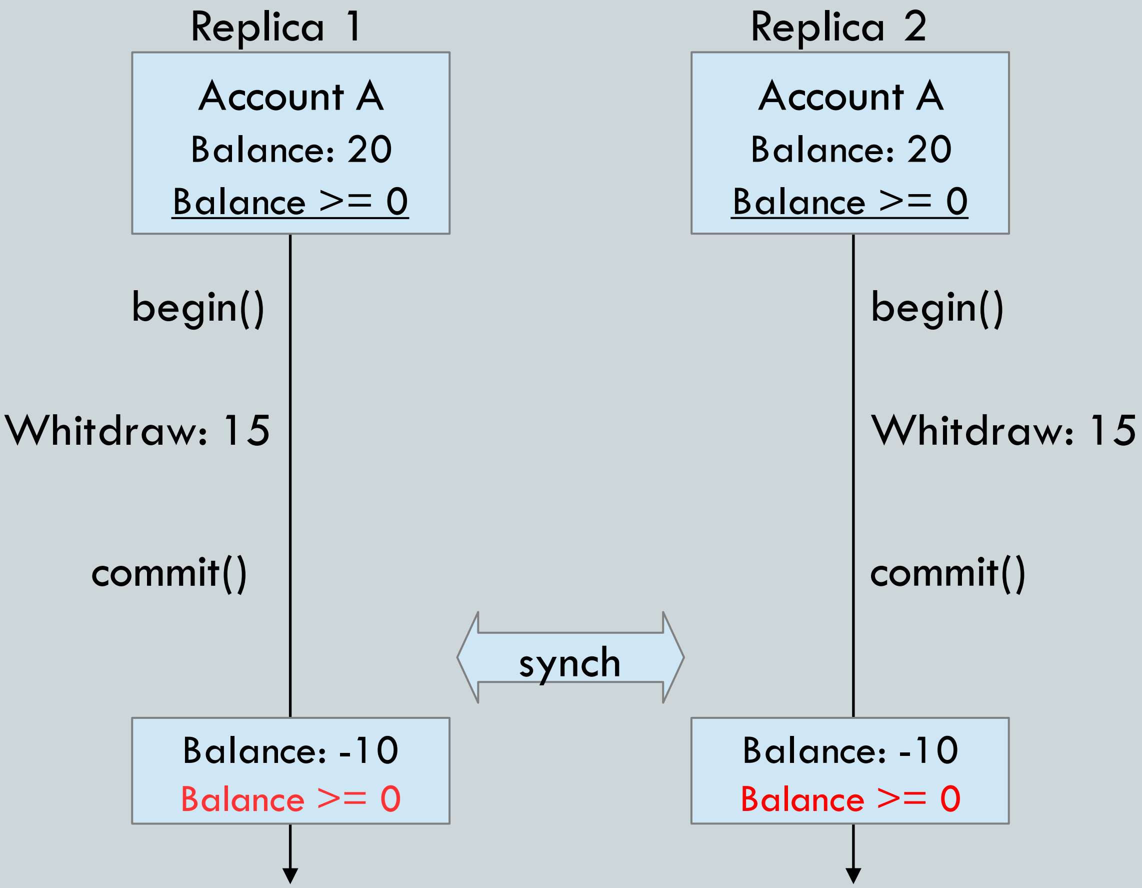
Data consistency limitations (1)

- Going beyond state convergence

- **Design:**
  - Asynchronous system
- **Problems:**
  - Maintaining data invariants
  - Referential integrity

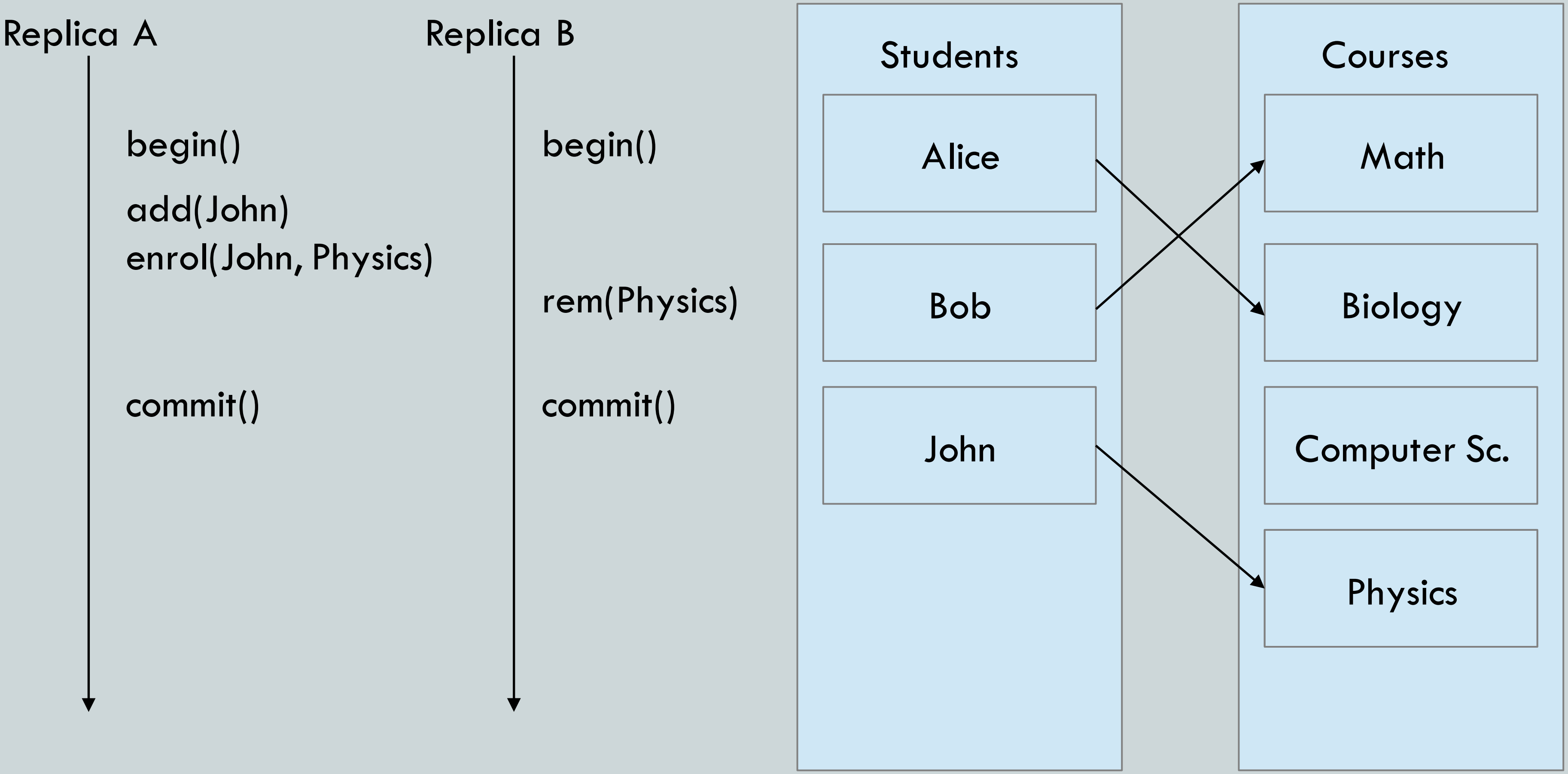
Data consistency limitations (2)

- Maintaining data invariants



Data consistency limitations (3)

- Referential Integrity



Replica A

begin()

add(John)

enrol(John, Physics)

commit()

Replica B

begin()

rem(Physics)

commit()

Students

Alice

Bob

John

Courses

Math

Biology

Computer Sc.

Physics

Data consistency limitations (4)

- **Problems:**
  - Maintaining data invariants
  - Referential integrity
- **Solutions:**
  - Reservation techniques

Conclusions

- Current design promotes simplicity
- System allows to implement TPC-W
  - Some operations are processed very inefficiently
  - Key-Value data-model not very suitable to this application
- We can always add more features to the data-model
  - ☹ More complexity at the data-centre
  - 😐 Key-Value store loses simplicity

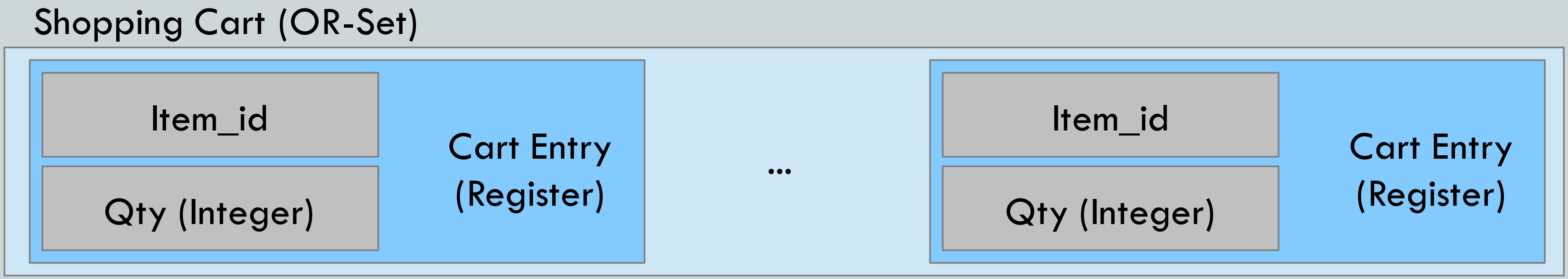
Questions?

Other limitations

- Data-model cut across layers
  - Cripples modularity and encapsulation
  - increase the points of vulnerability

Data-model adaptation

- Simple data structures easily implemented with current CRDT Library
  - Registers to store entities (authors, addresses,...)
  - OR-Sets to avoid loosing updates on the shopping cart
  - Counters to store items stock and amount sold
- However... Complex CRDTs not implemented efficiently without CRDT composition


Shopping Cart (OR-Set)
Item_id
Qty (Integer)
Cart Entry (Register)
...
Item_id
Qty (Integer)
Cart Entry (Register)