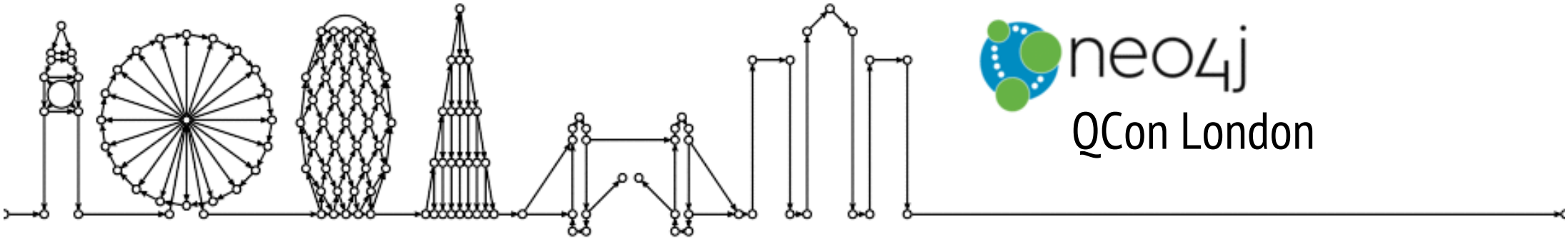
Causal Consistency For Large Neo4j Clusters

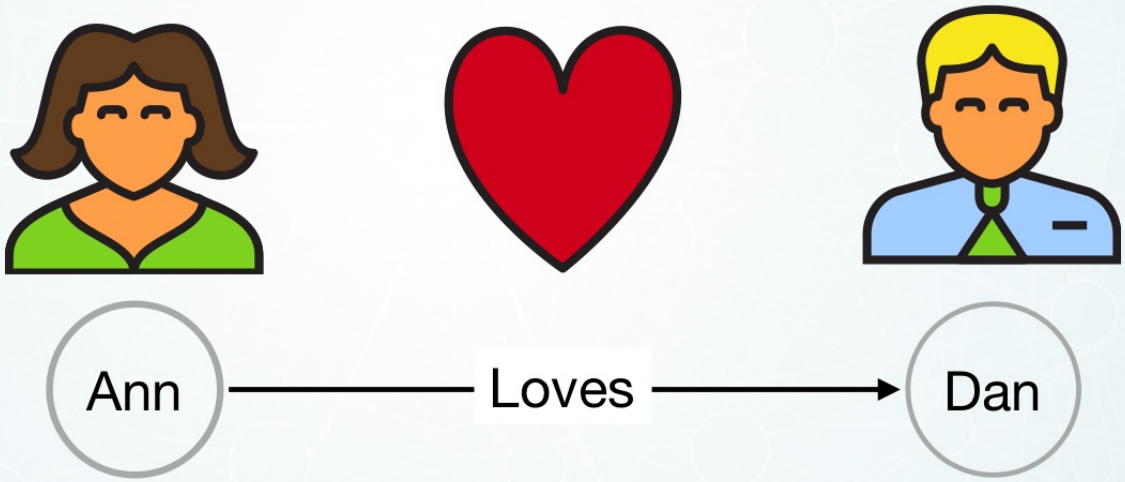
Jim Webber

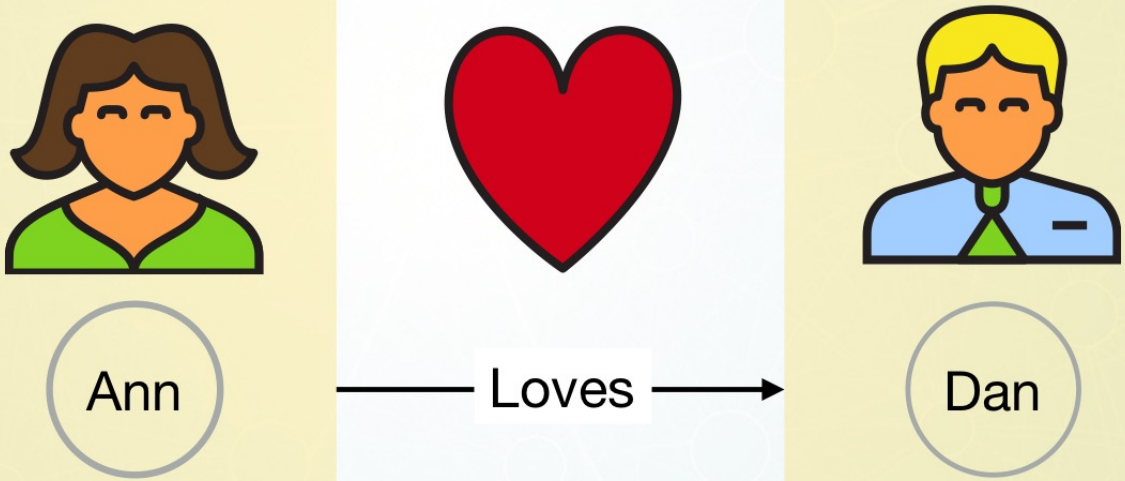
Chief Scientist, Neo4j



neo4j

QCon London


Ann
Loves
Dan

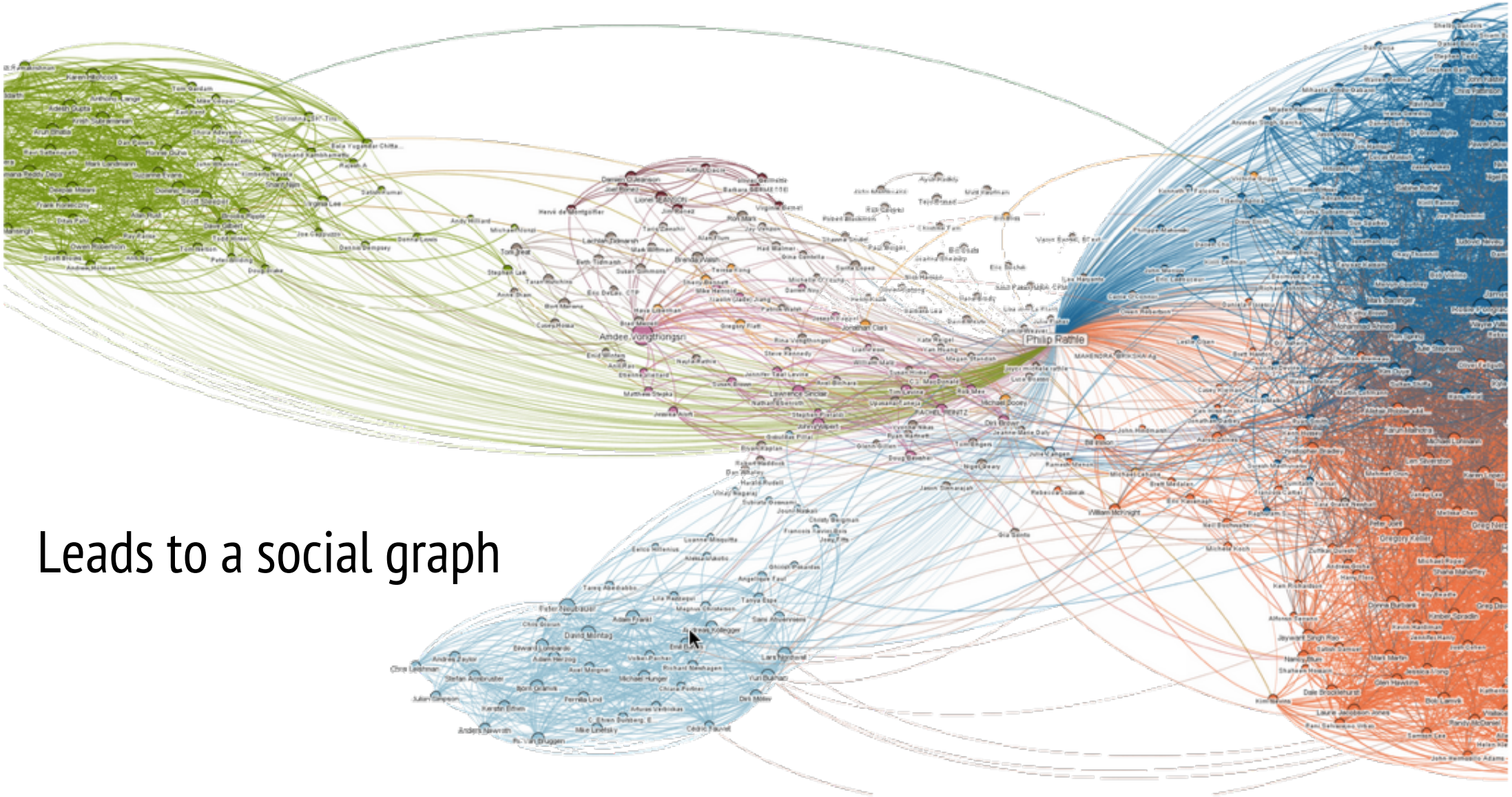


(:Person {name:"Ann"}) –[:LOVES]-> (:Person {name:"Dan"})

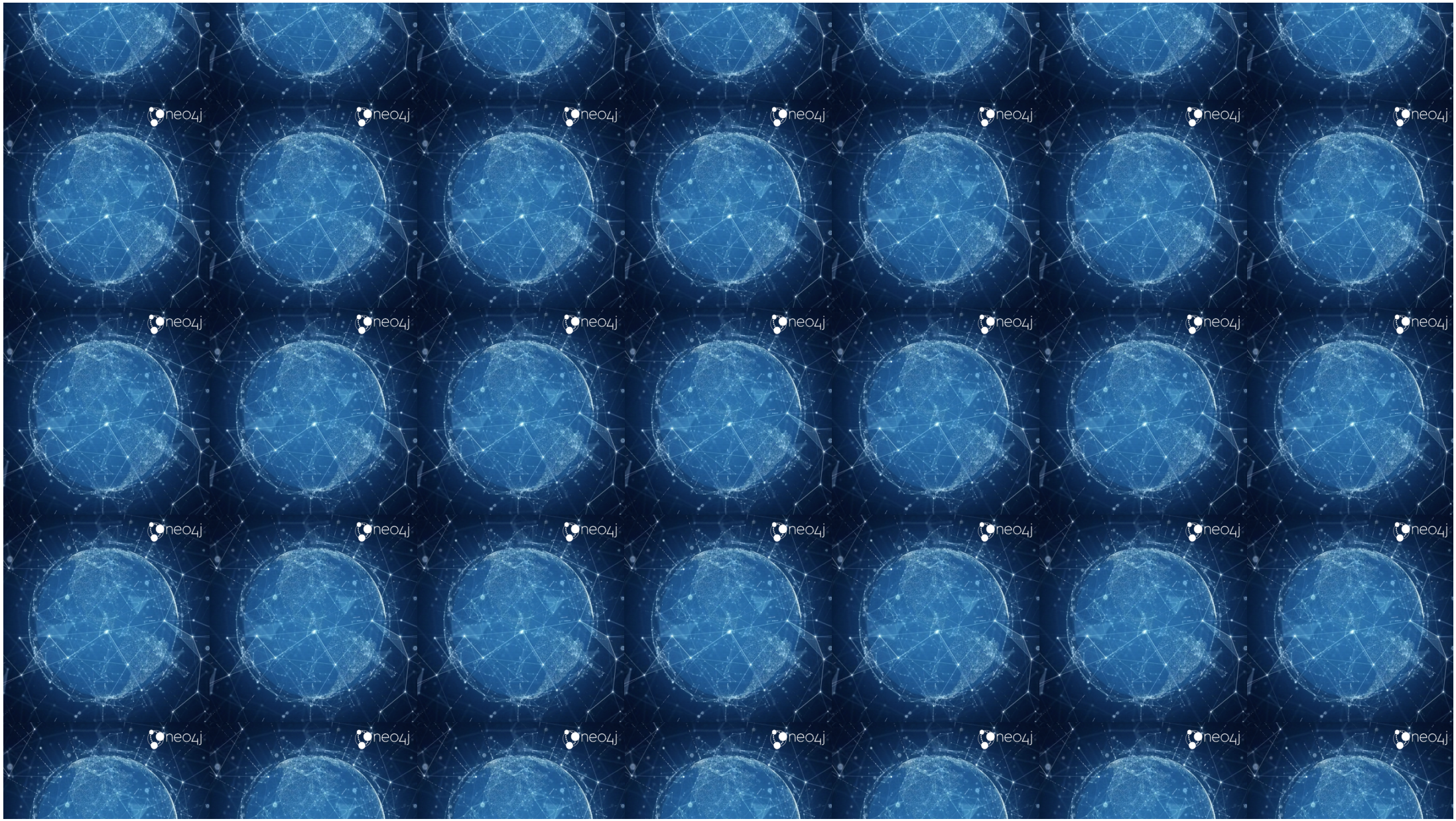
Node **Relationship** **Node**

Query: Whom does Ann love?

```
MATCH (:Person {name:"Ann"})–[:LOVES]->(whom)

RETURN whom
```

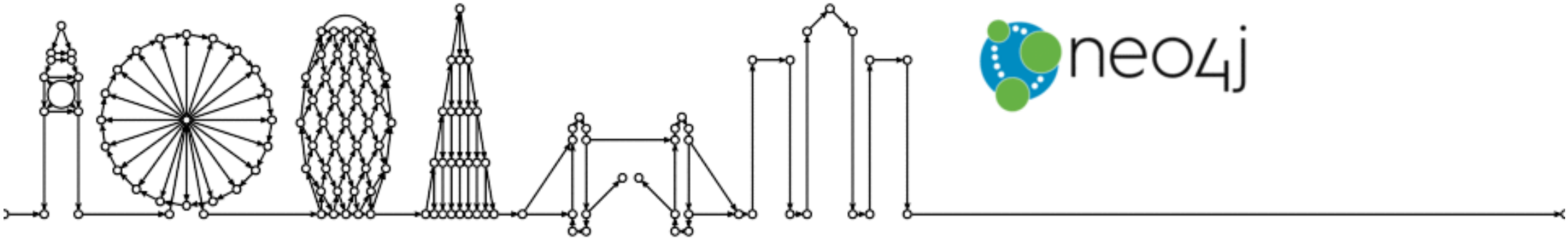


Leads to a social graph

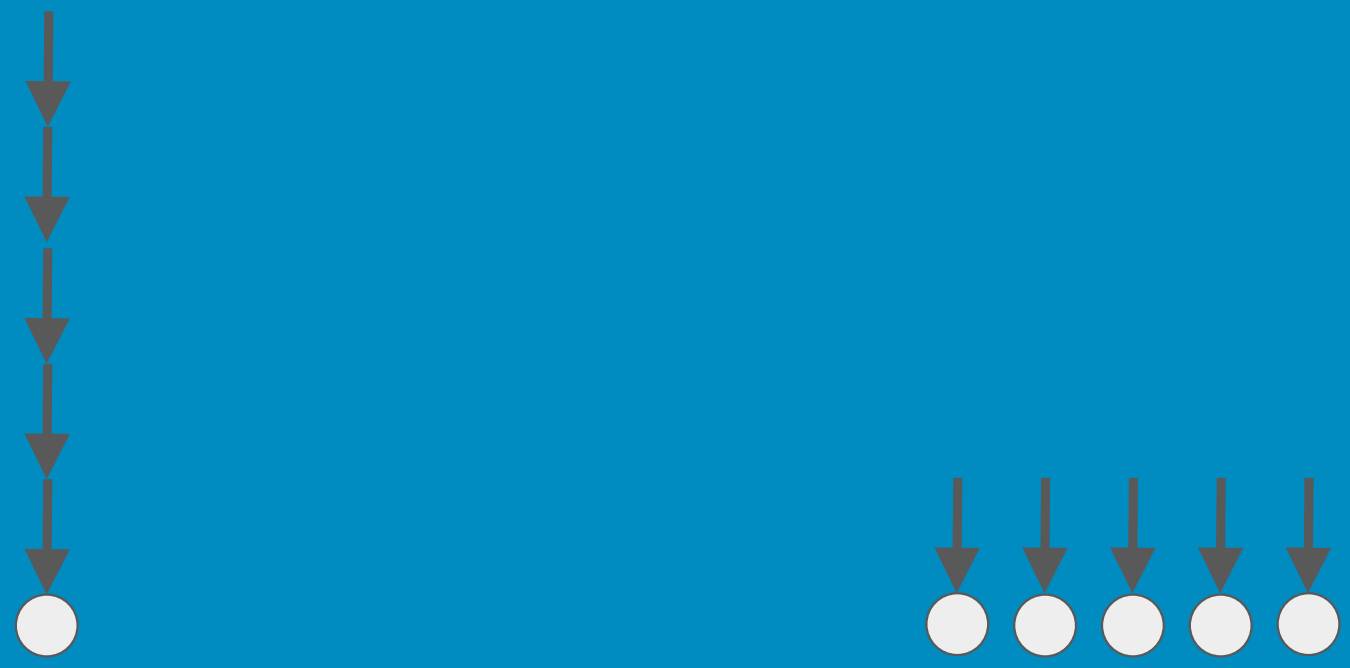

neo4j

Motivation

Why do we need clusters of Neo4j?



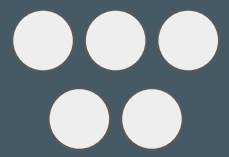
Massive Throughput



Data Redundancy

Data Redundancy

Data Redundancy



Data Redundancy

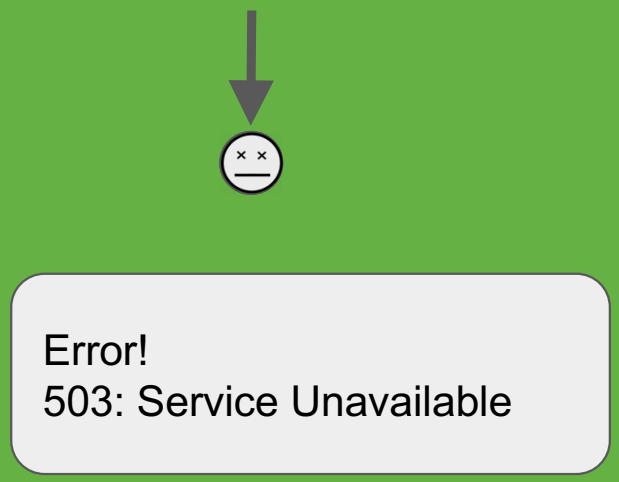
High Availability

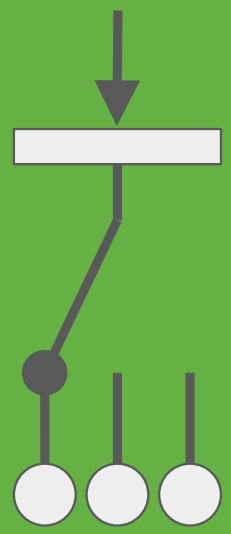
High Availability

High Availability

Error!
503: Service Unavailable

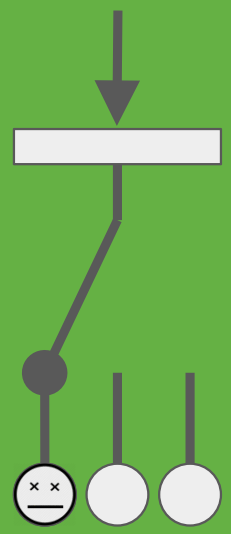
High Availability





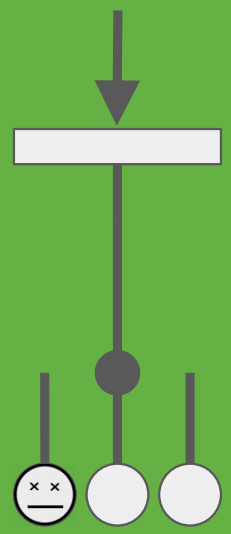
High Availability

Error!
503: Service Unavailable

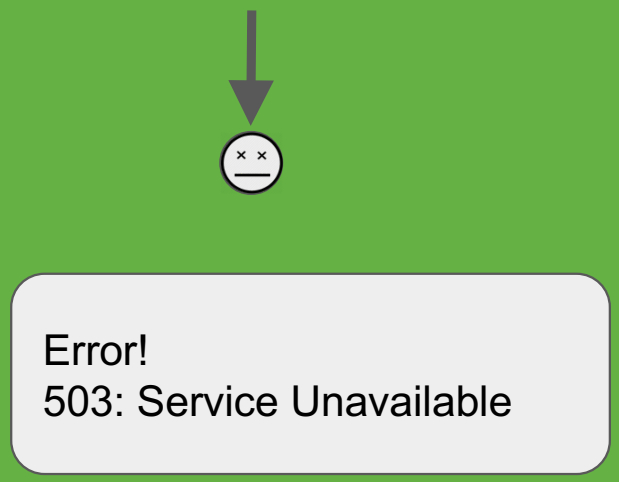


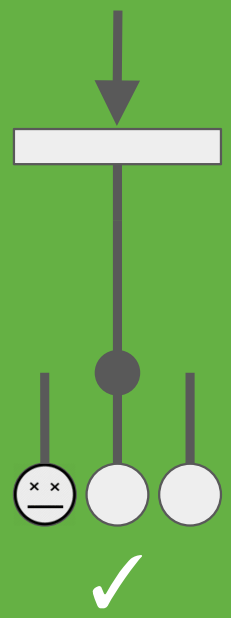
High Availability

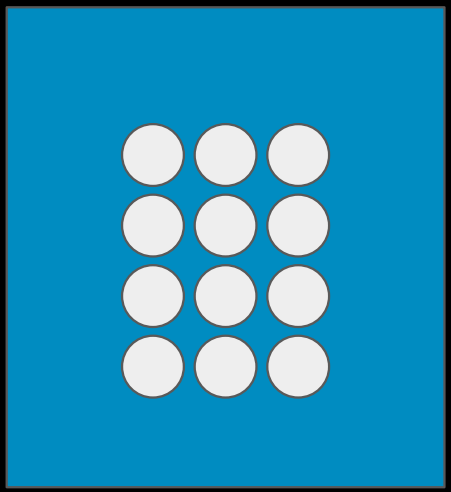
Error!
503: Service Unavailable



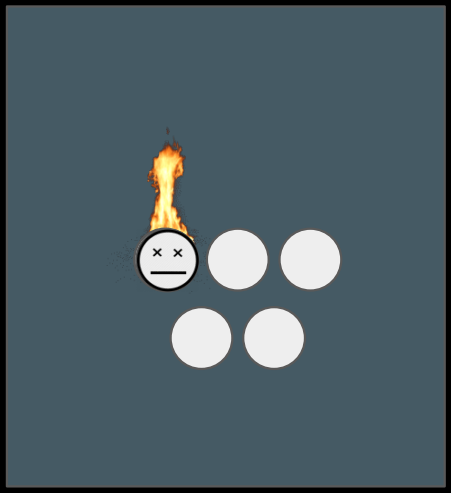
High Availability



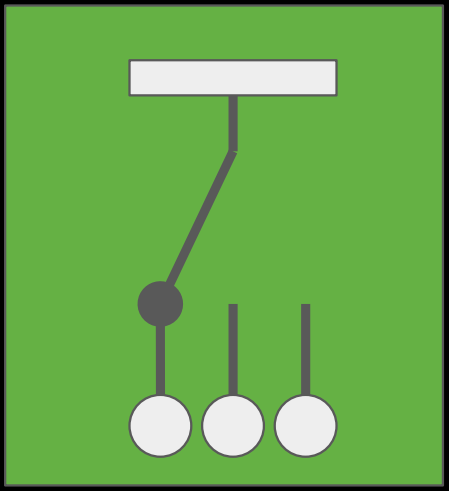




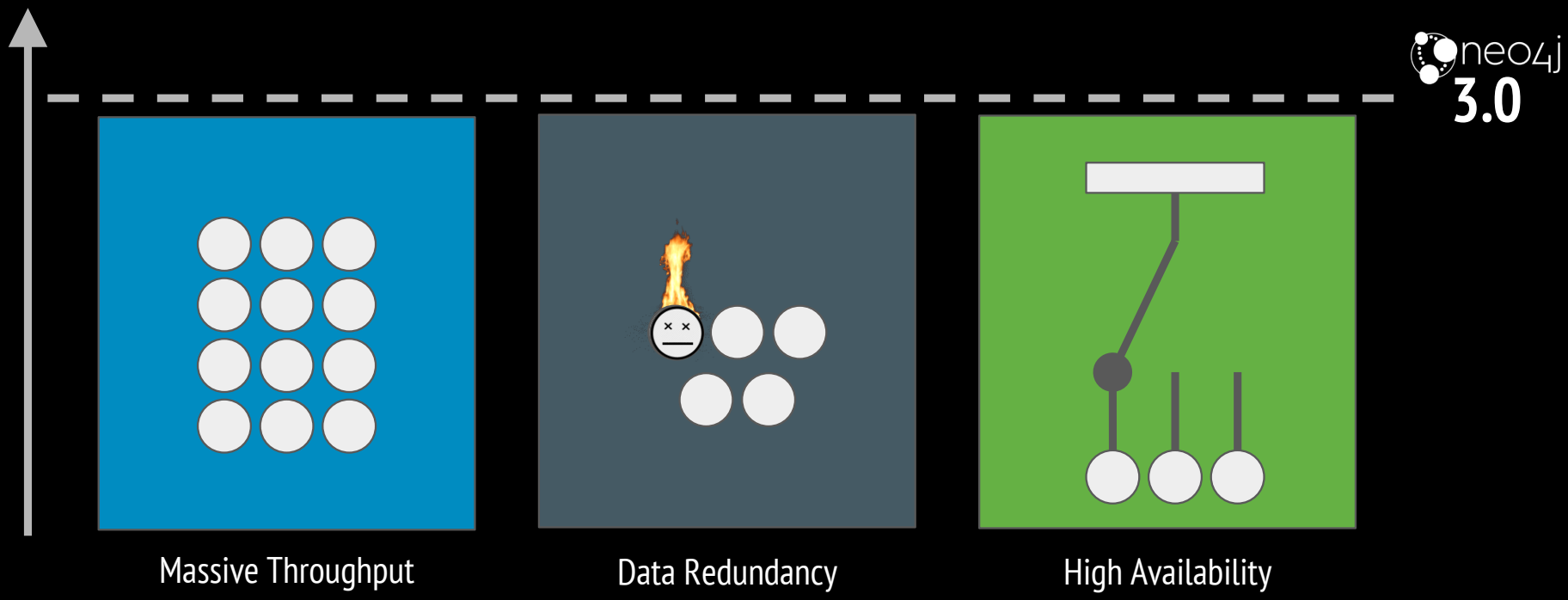
Massive Throughput

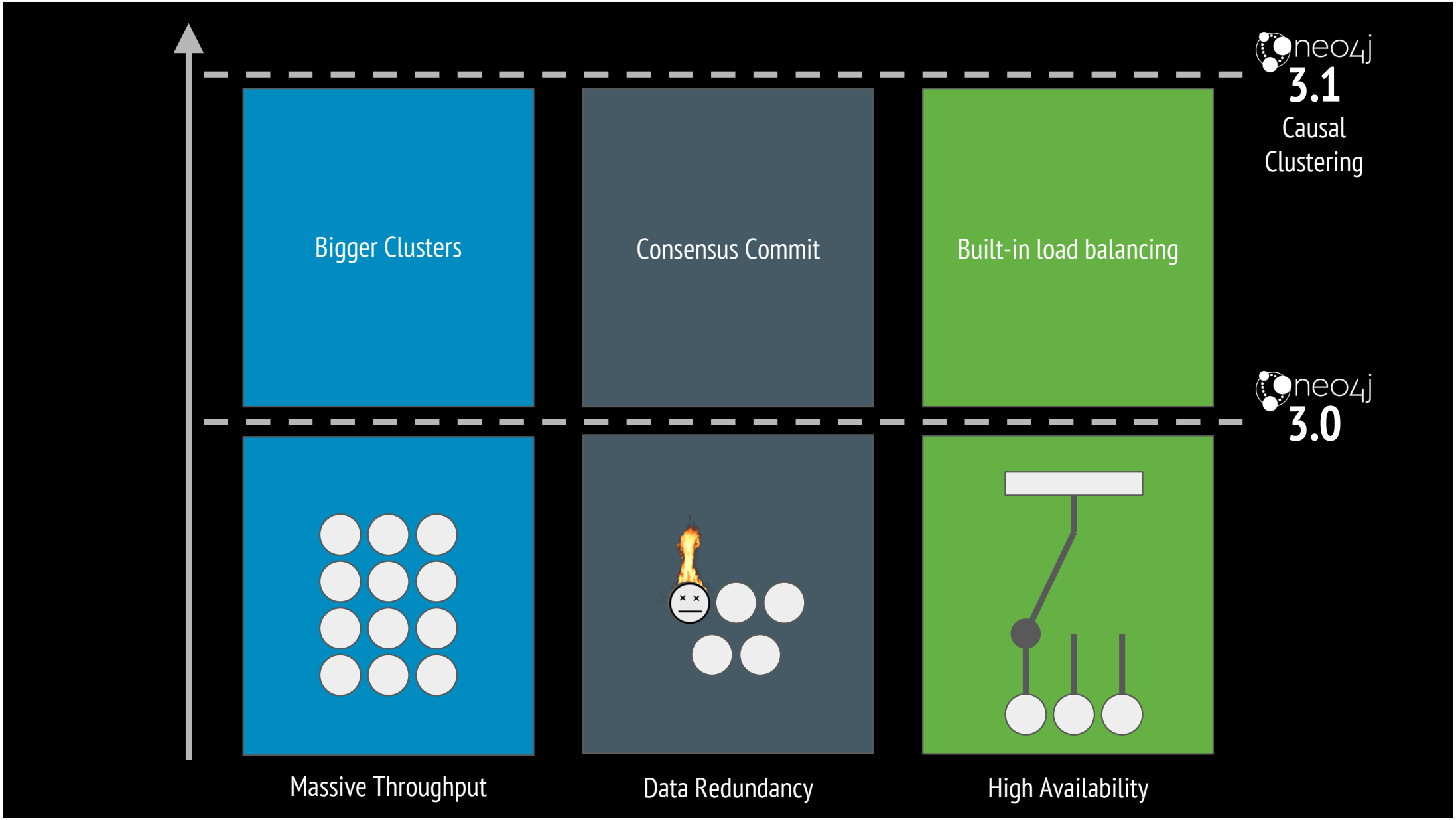


Data Redundancy



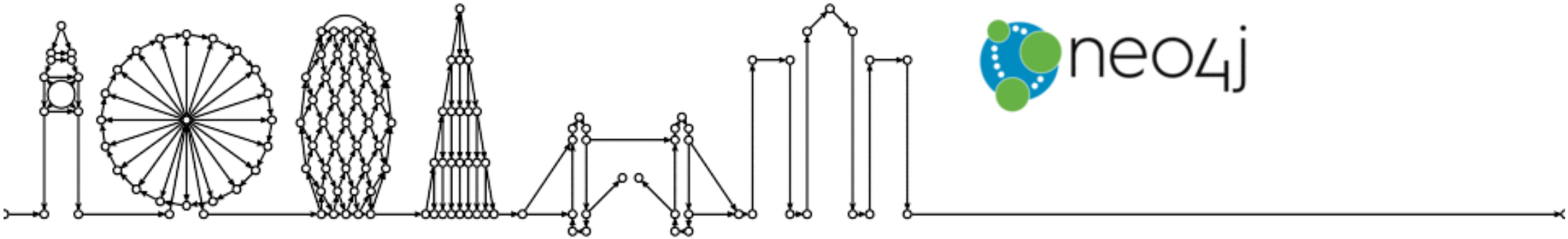
High Availability

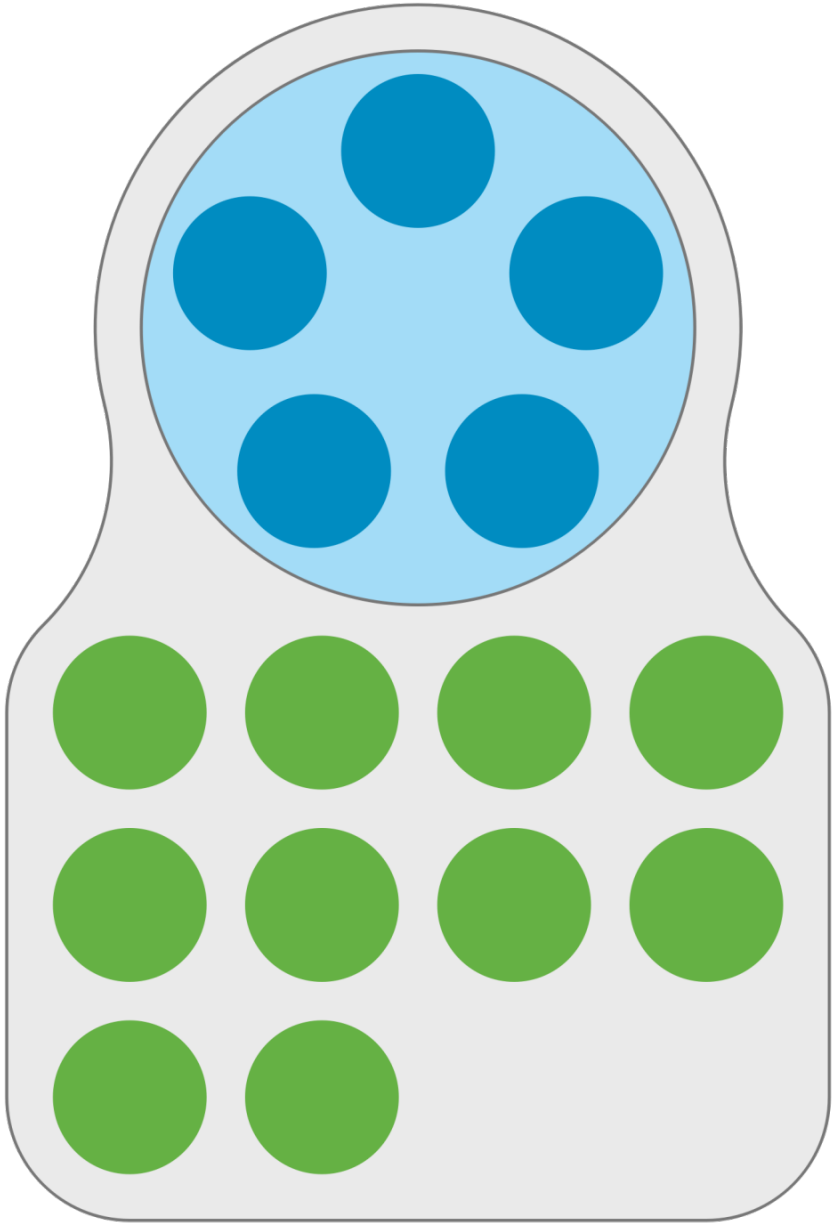

neo4j
3.0
Massive Throughput
Data Redundancy
High Availability


neo4j
3.1
Causal
Clustering
Bigger Clusters
Consensus Commit
Built-in load balancing
neo4j
3.0
Massive Throughput
Data Redundancy
High Availability

Roles for safety and scale

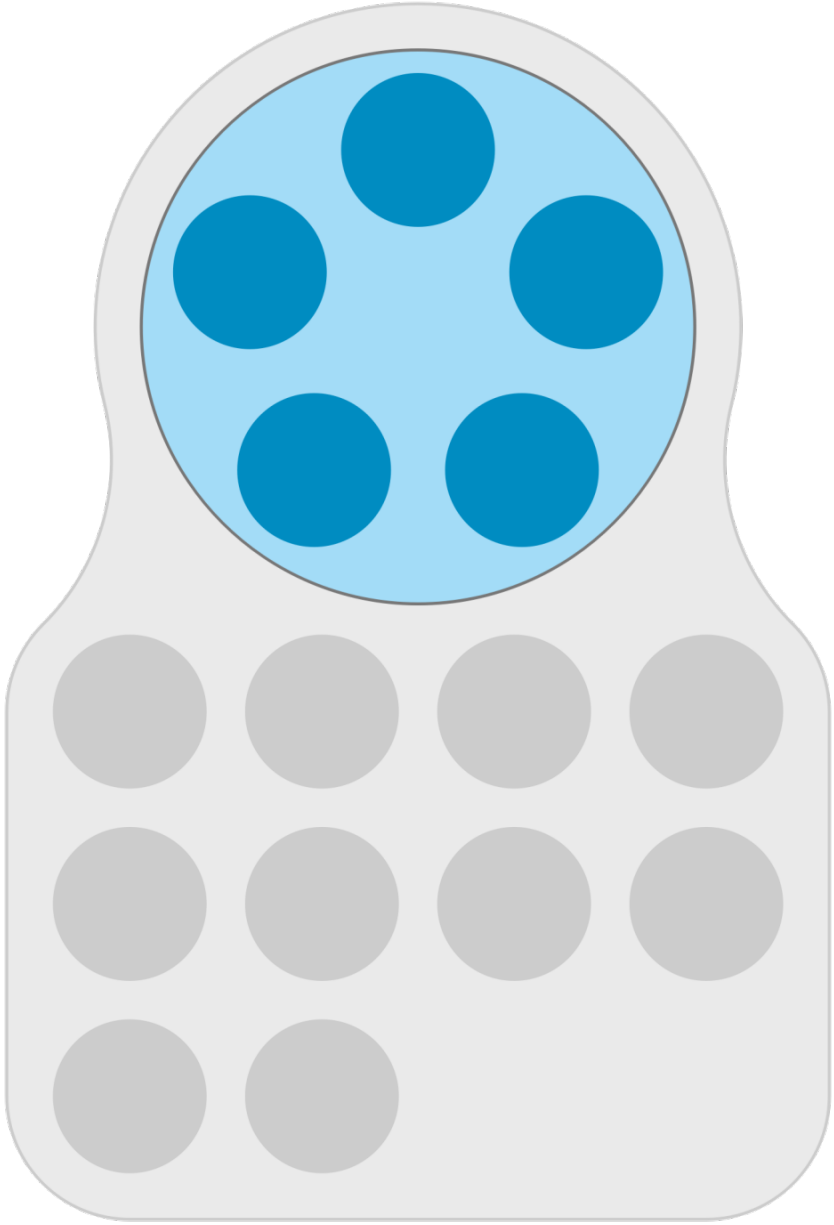
Divide and conquer complexity





Core

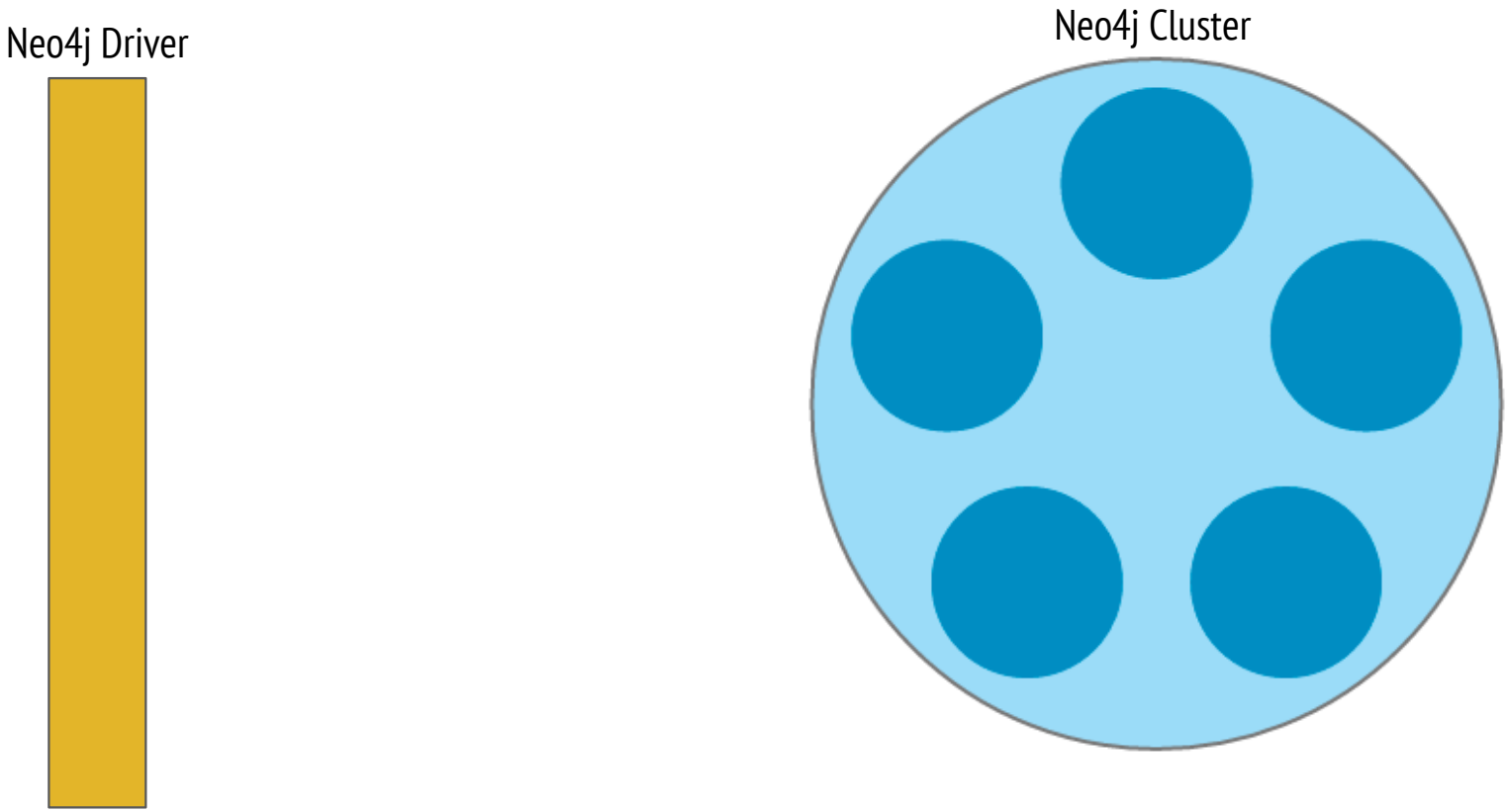
Read Replicas



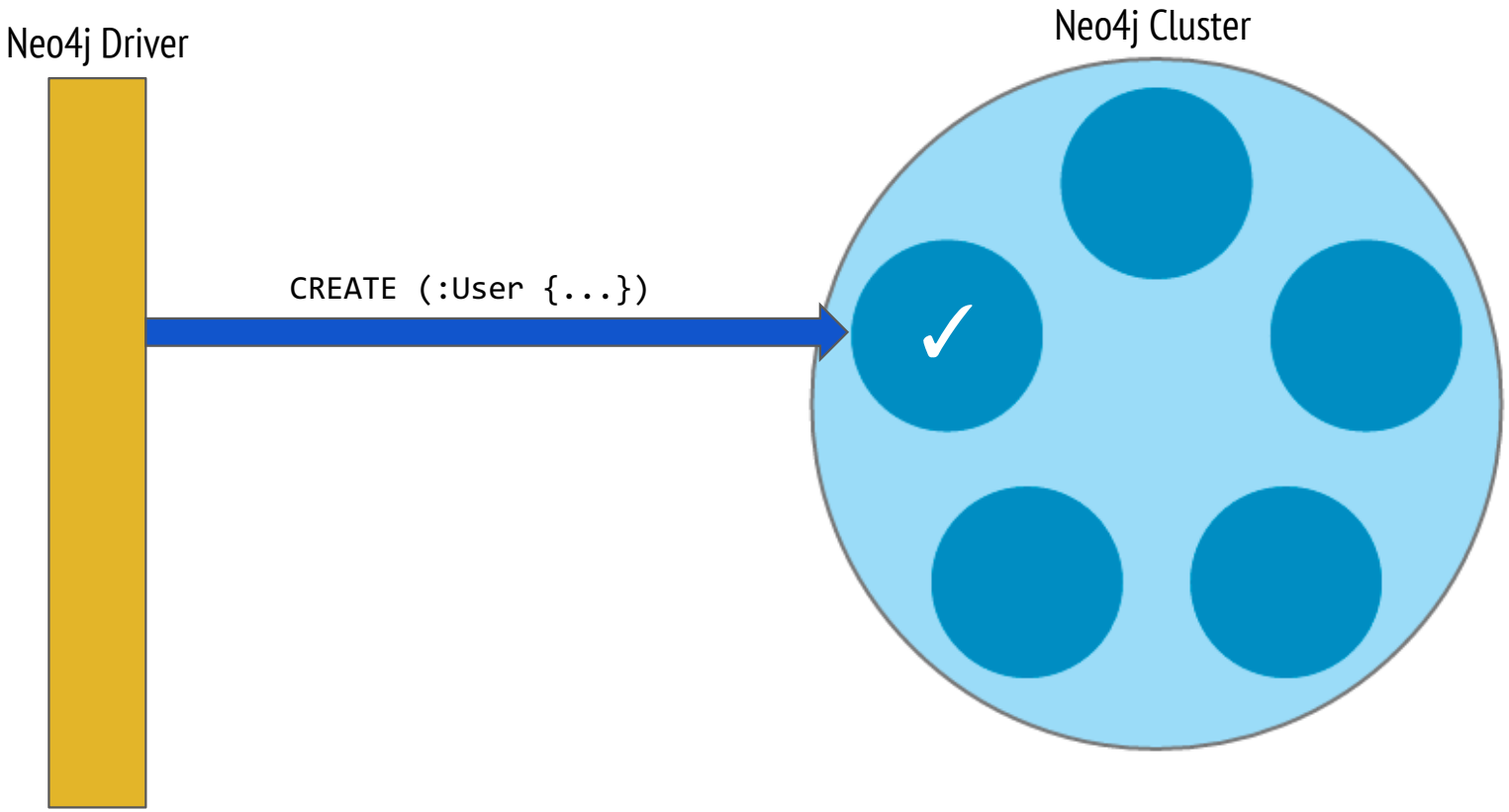
Core

- Small group of Neo4j databases
- Fault-tolerant Consensus Commit
- Responsible for data safety

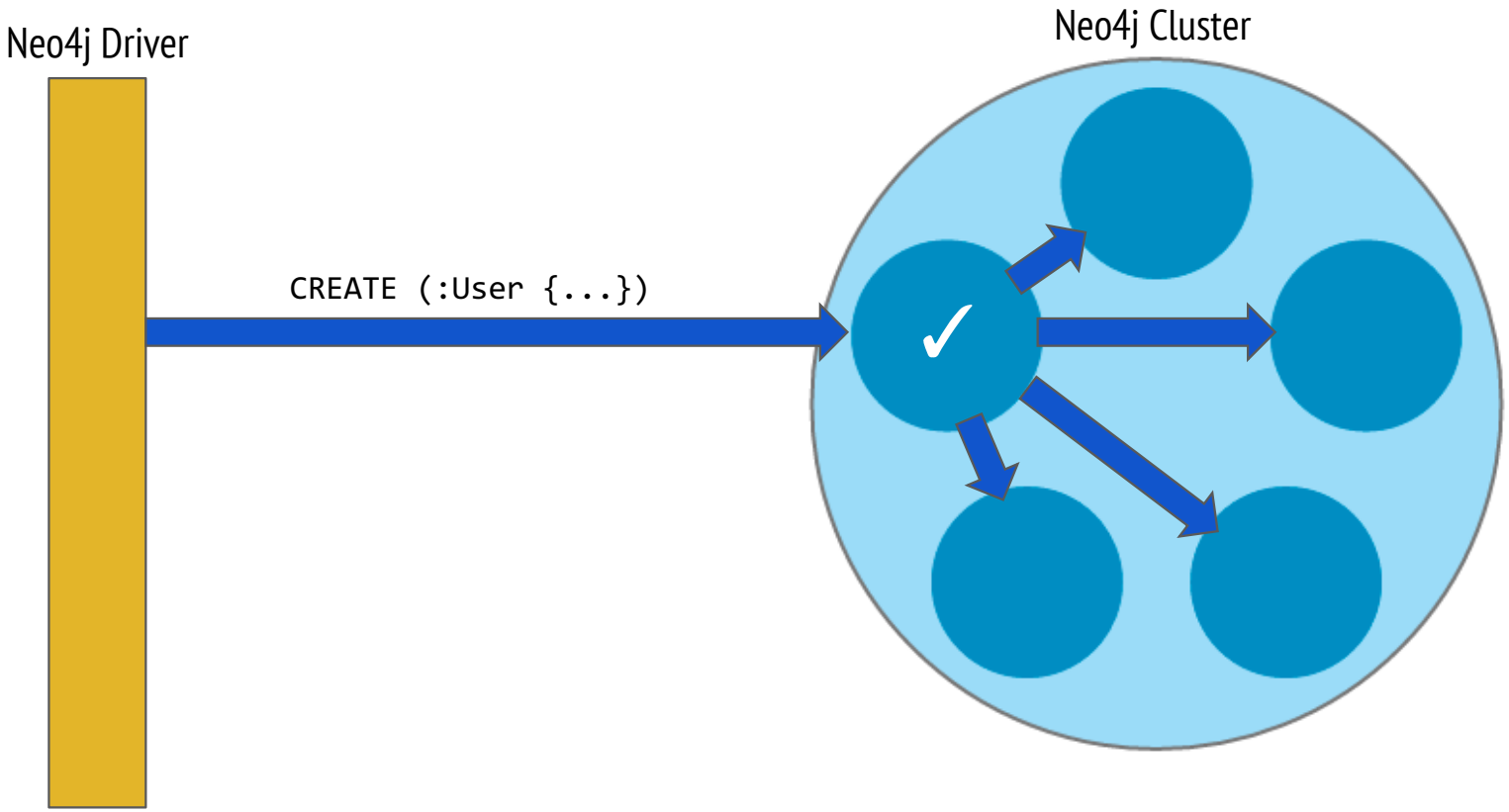
Writing to the Core Cluster



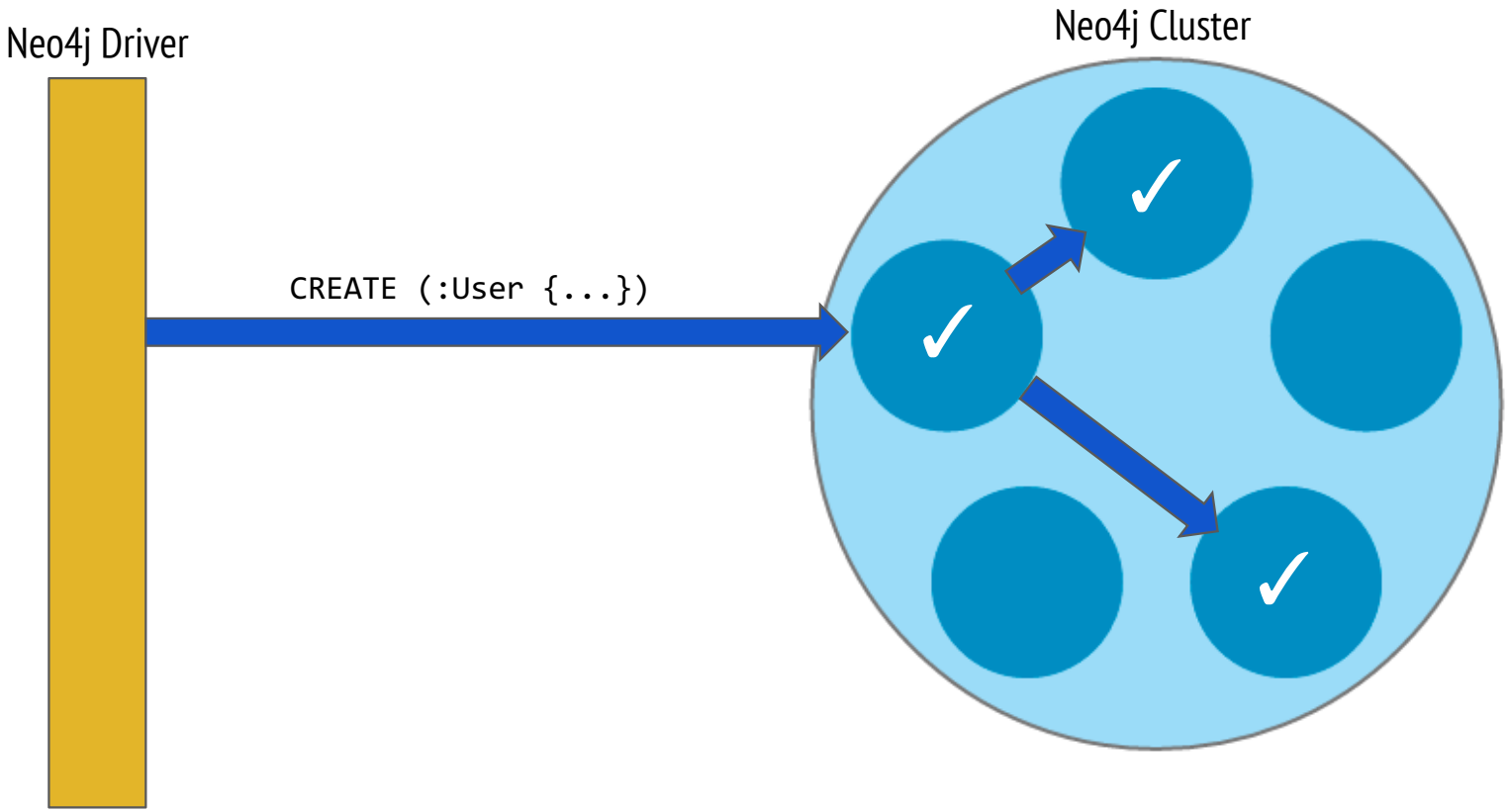
Writing to the Core Cluster



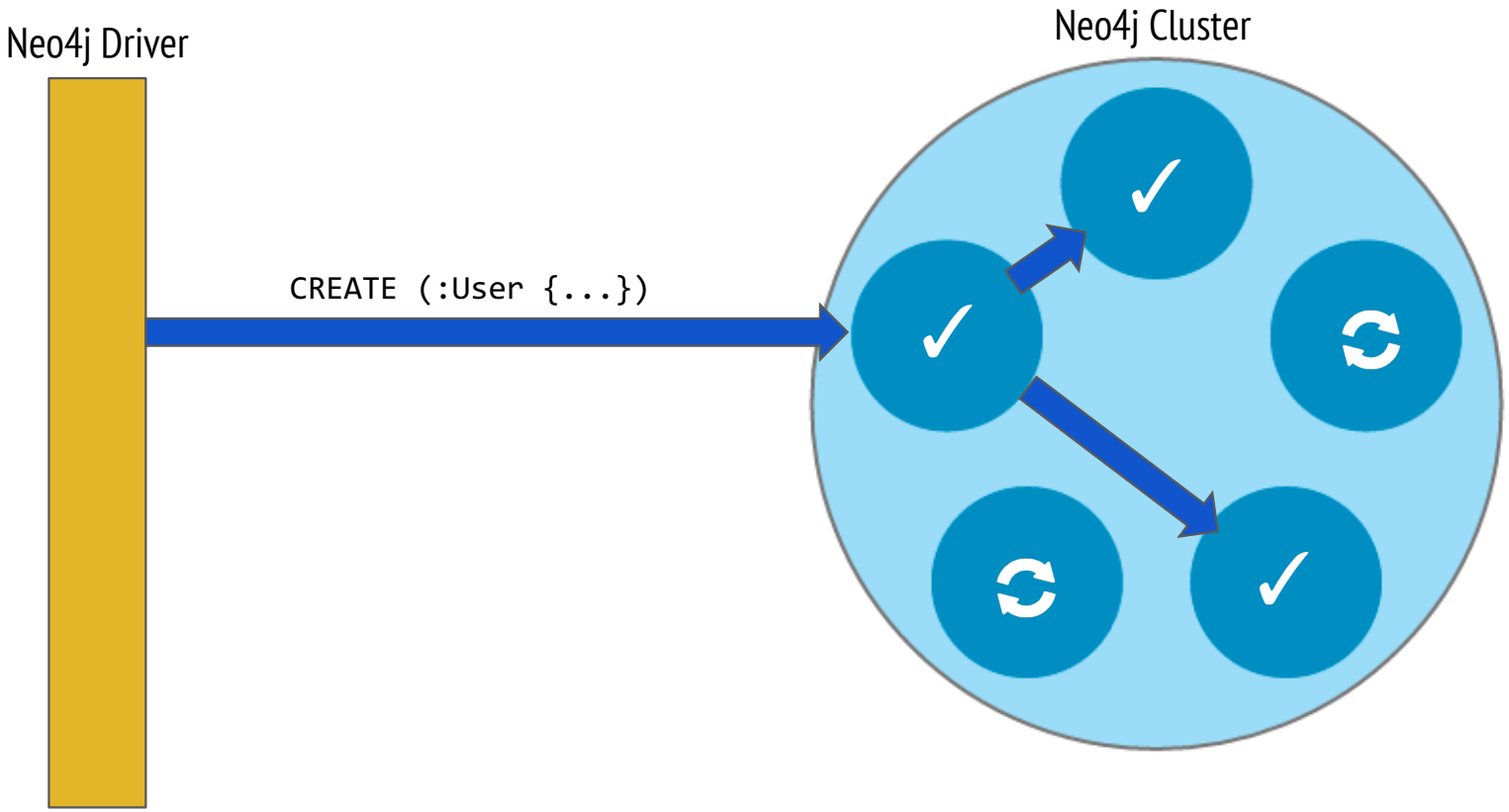
Writing to the Core Cluster



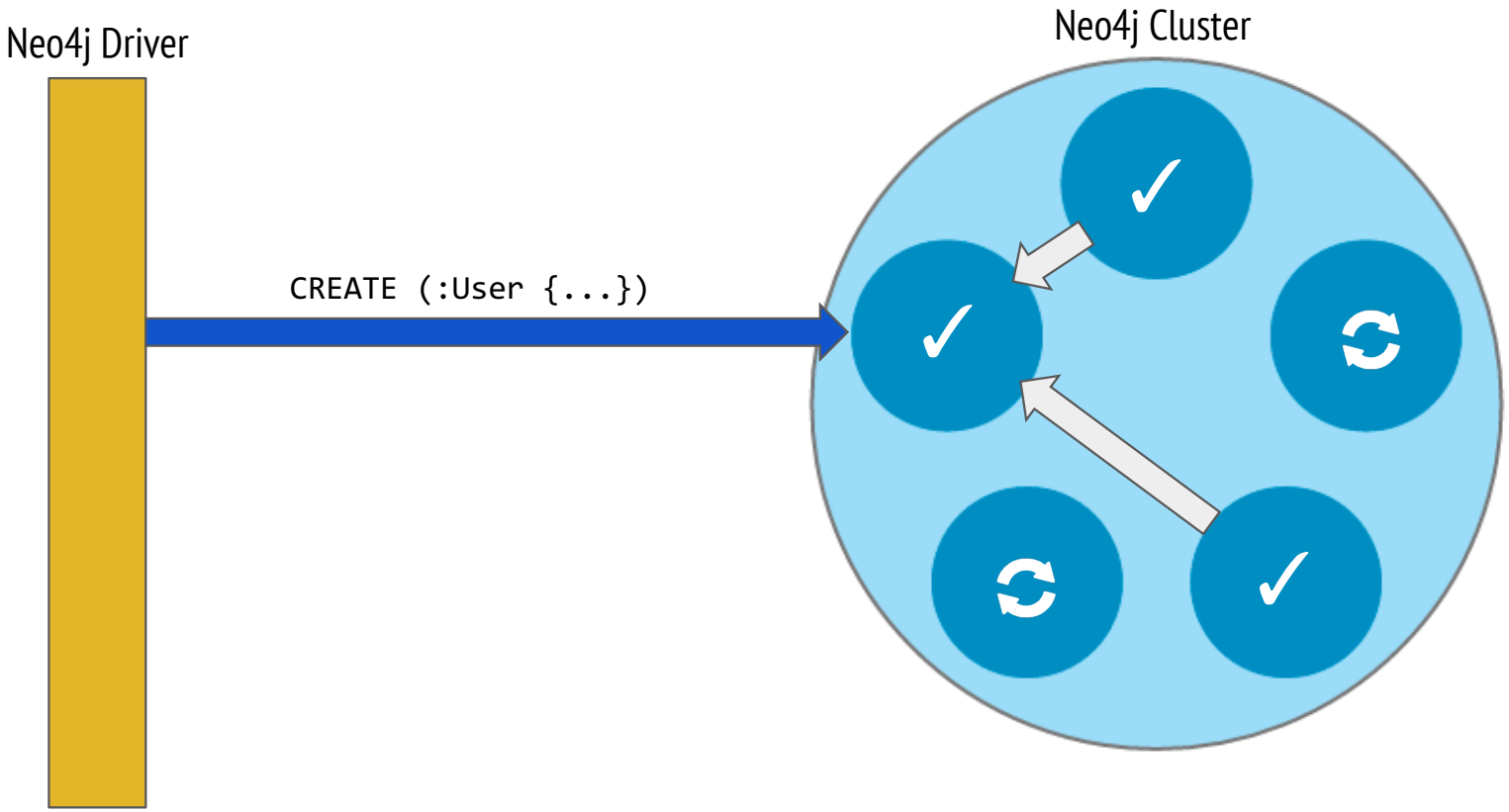
Writing to the Core Cluster



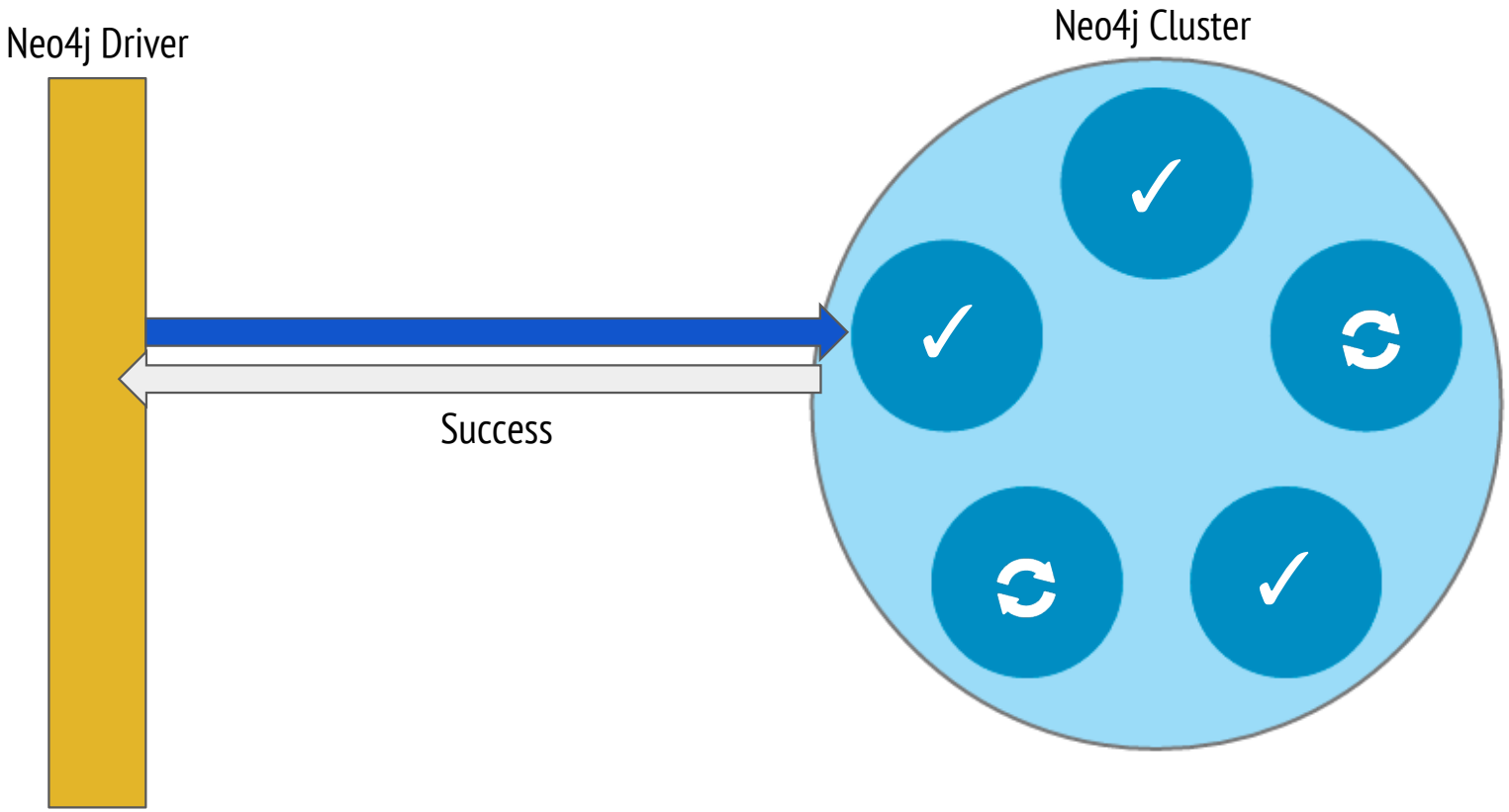
Writing to the Core Cluster



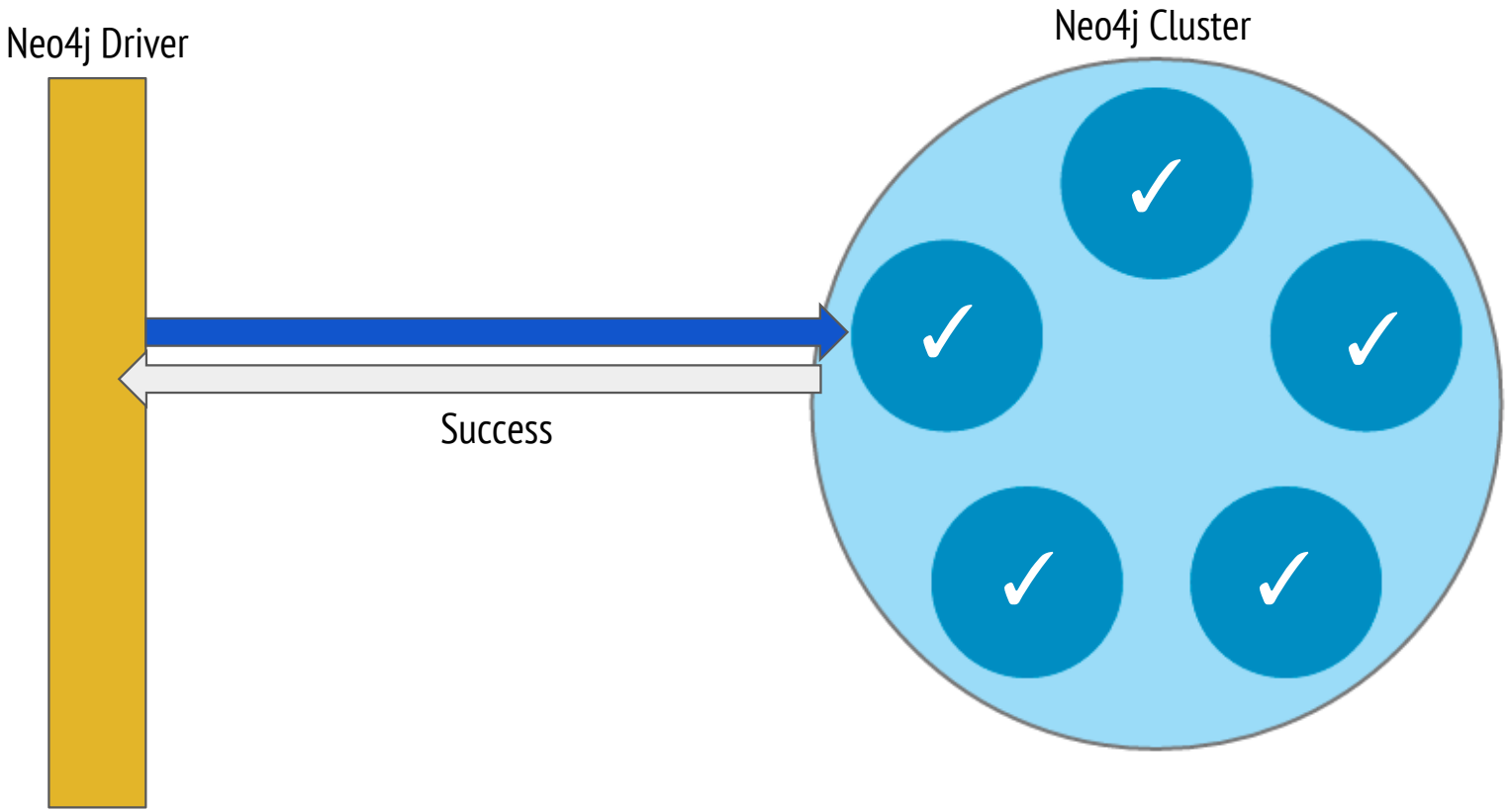
Writing to the Core Cluster



Writing to the Core Cluster

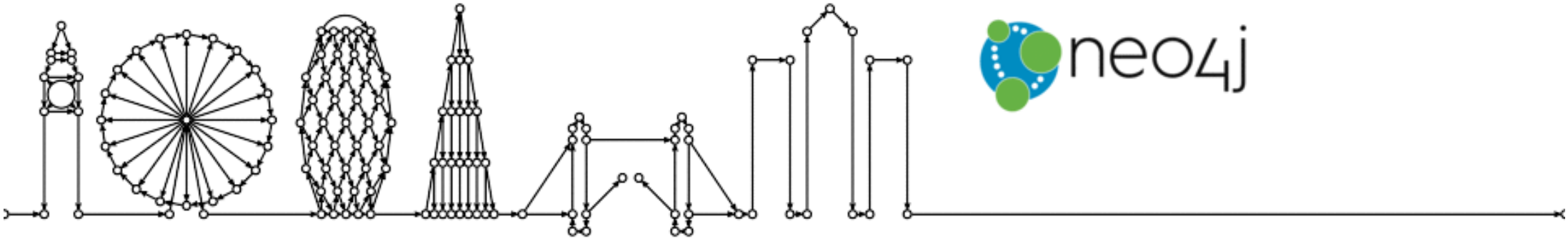


Writing to the Core Cluster



Raft Protocol

Non-Blocking Consensus for Humans



In Search of an Understandable Consensus Algorithm (Extended Version)

Diego Ongaro and John Ousterhout
Stanford University

EASY TO UNDERSTAND

Abstract

Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems. In order to enhance understandability, Raft separates the key elements of consensus, such as leader election, log replication, and safety, and it enforces a stronger degree of coherency to reduce the number of states that must be considered. Results from a user study demonstrate that Raft is easier for students to learn than Paxos. Raft also includes a new mechanism for changing the cluster membership, which uses overlapping majorities to guarantee safety.

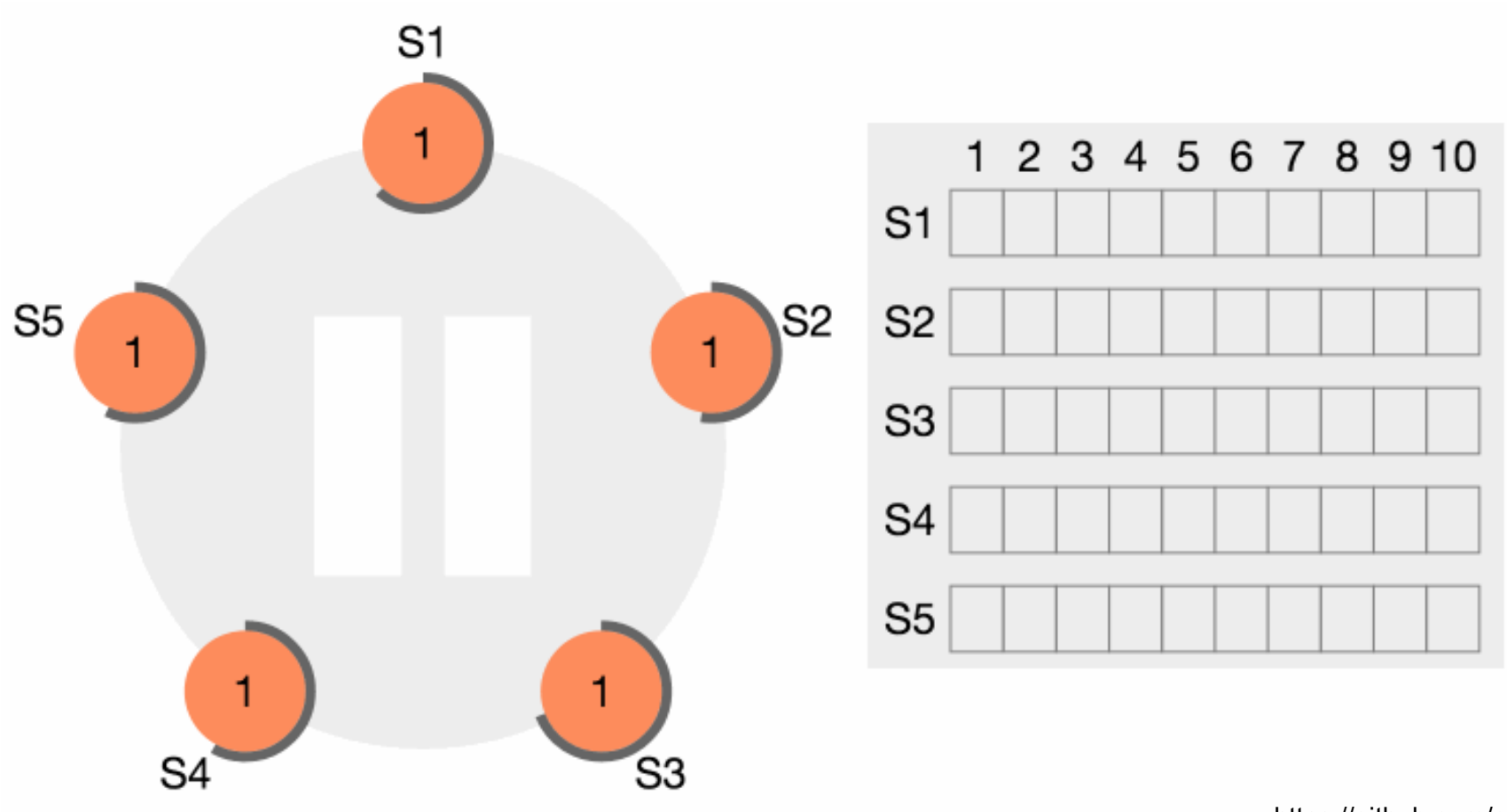
1 Introduction

state space reduction (relative to Paxos, Raft reduces the degree of nondeterminism and the ways servers can be inconsistent with each other). A user study with 43 students at two universities shows that Raft is significantly easier to understand than Paxos: after learning both algorithms, 33 of these students were able to answer questions about Raft better than questions about Paxos.

Raft is similar in many ways to existing consensus algorithms (most notably, Oki and Liskov's Viewstamped Replication [29, 22]), but it has several novel features:

- **Strong leader:** Raft uses a stronger form of leadership than other consensus algorithms. For example, log entries only flow from the leader to other servers. This simplifies the management of the replicated log and makes Raft easier to understand.
- **Leader election:** Raft uses randomized timers to

Raft Protocol



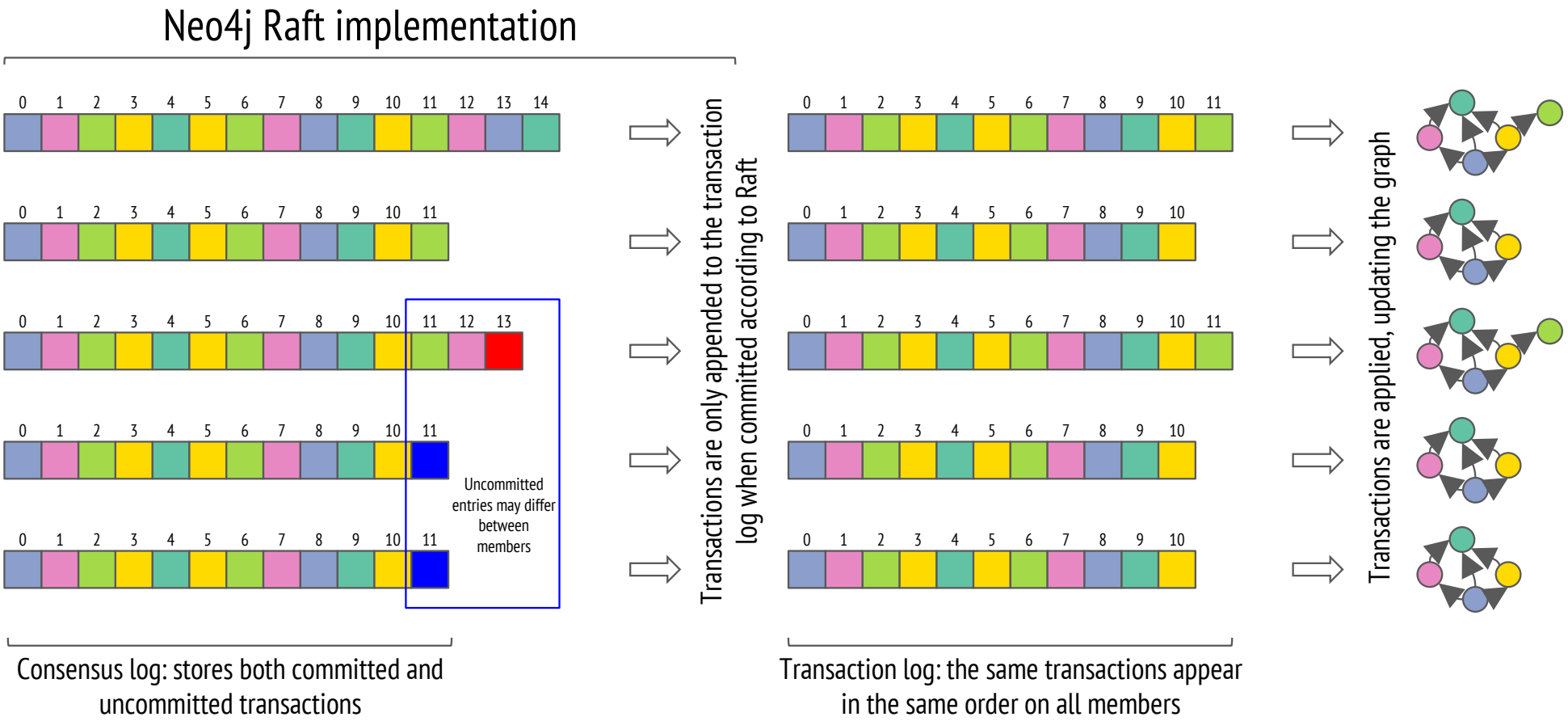
https://github.com/ongardie/raftscope

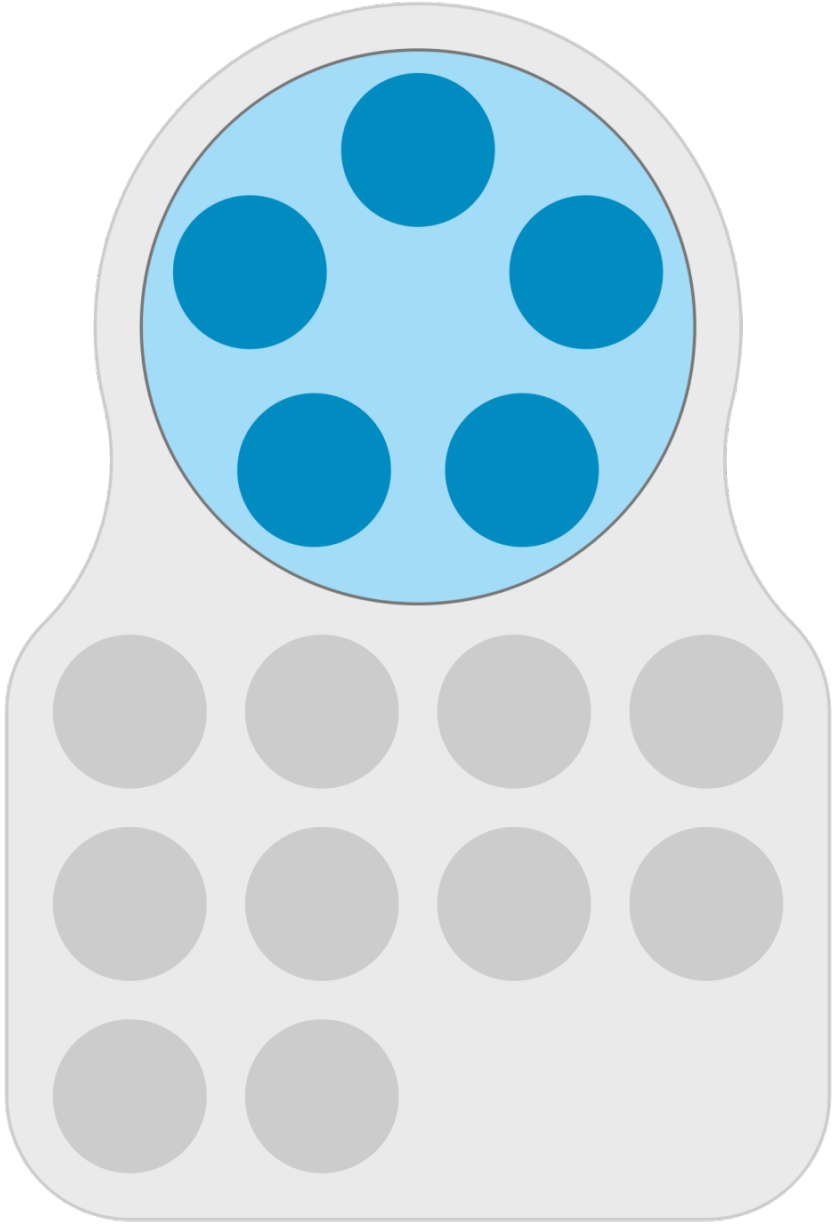
Raft in a Nutshell

- Raft keeps logs tied together (geddit?)
- Logs contain entries for both the database and the cluster membership
- Entries are appended and subsequently committed if a simple majority agree
- Implication: majority agree with the log as proposed
- Anyone can call an election: highest term (logical clock) wins, followed by highest committed, followed by highest appended.
- Appended, but not committed, entries can be truncated, but this is safe (translates as transaction aborted)

https://github.com/ongardie/raftscope

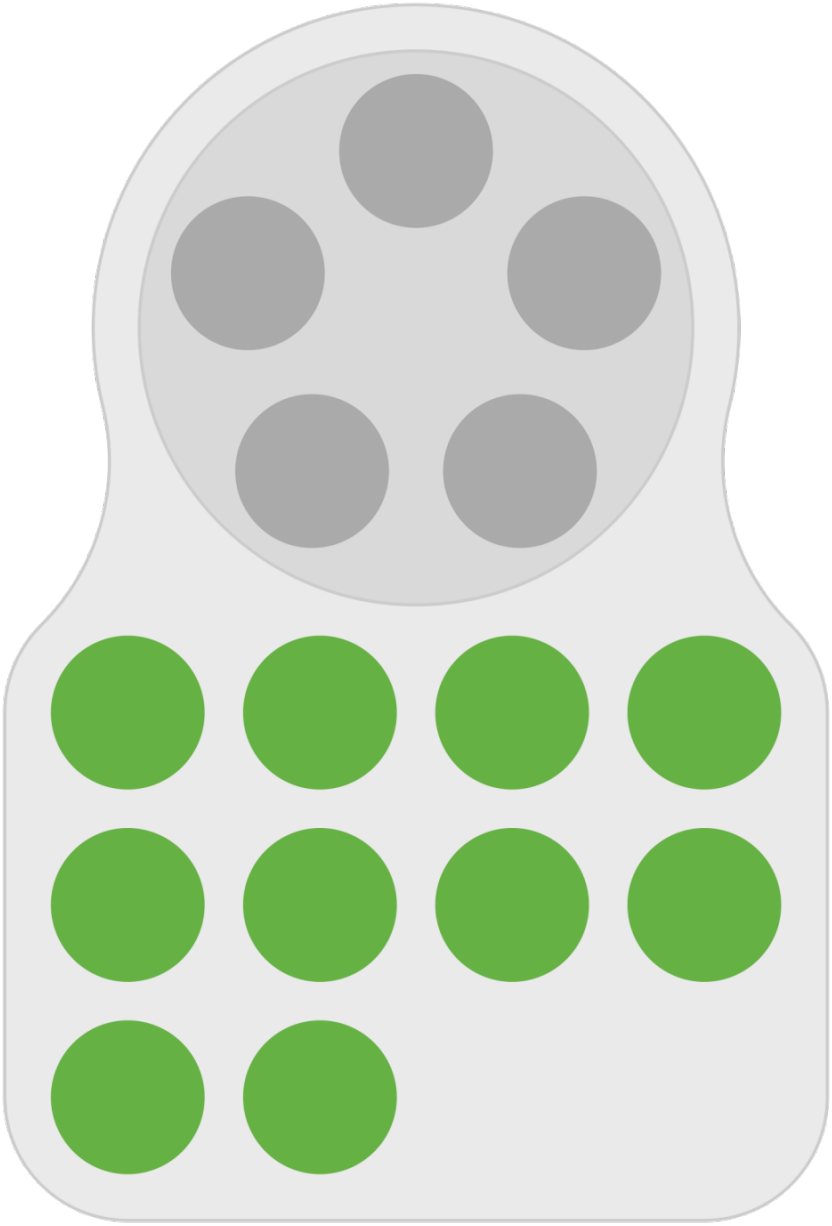
Consensus Log → Committed Transactions → Updated Graph





Core

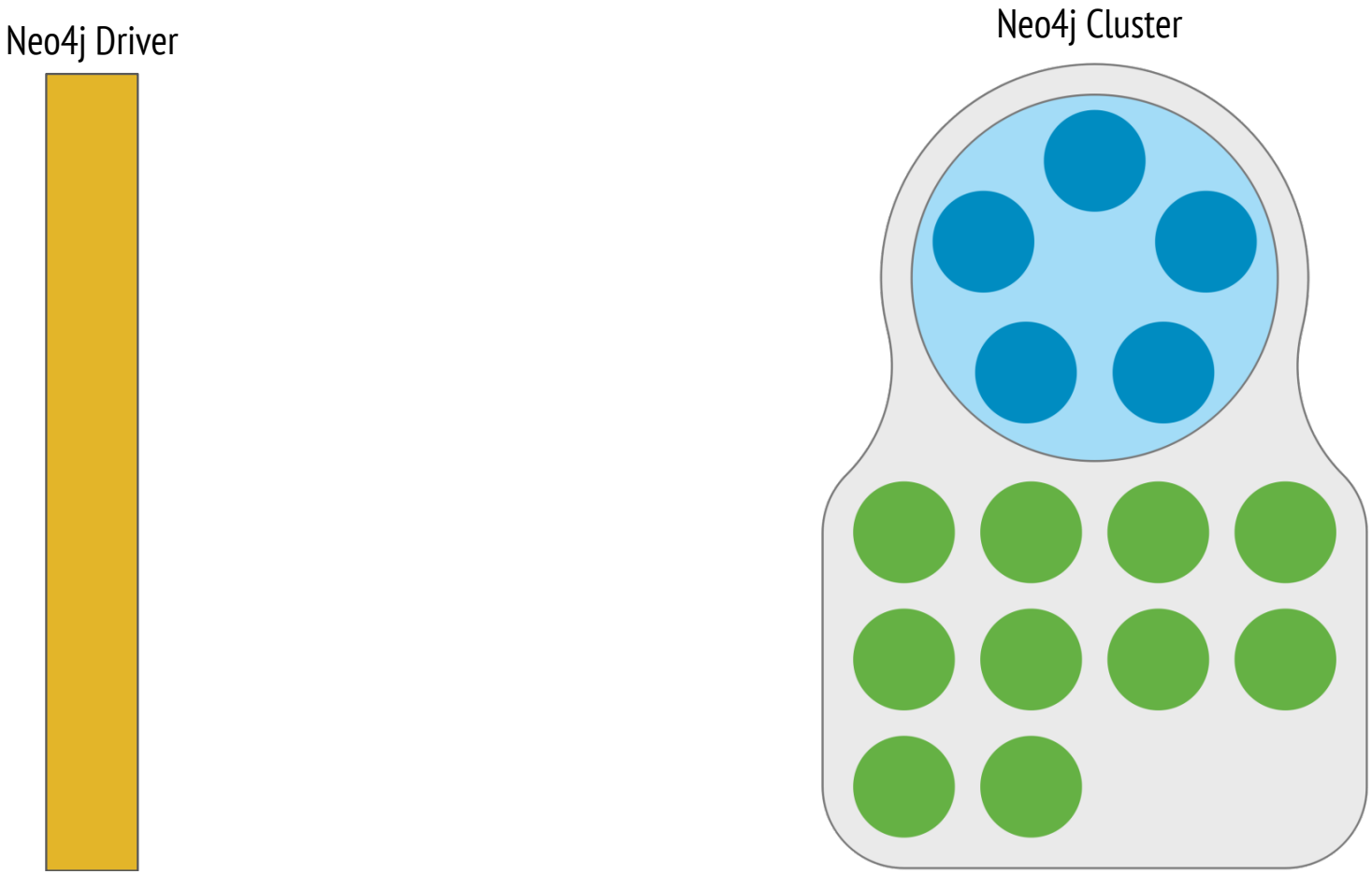
- Small group of Neo4j databases
- Fault-tolerant Consensus Commit
- Responsible for data safety



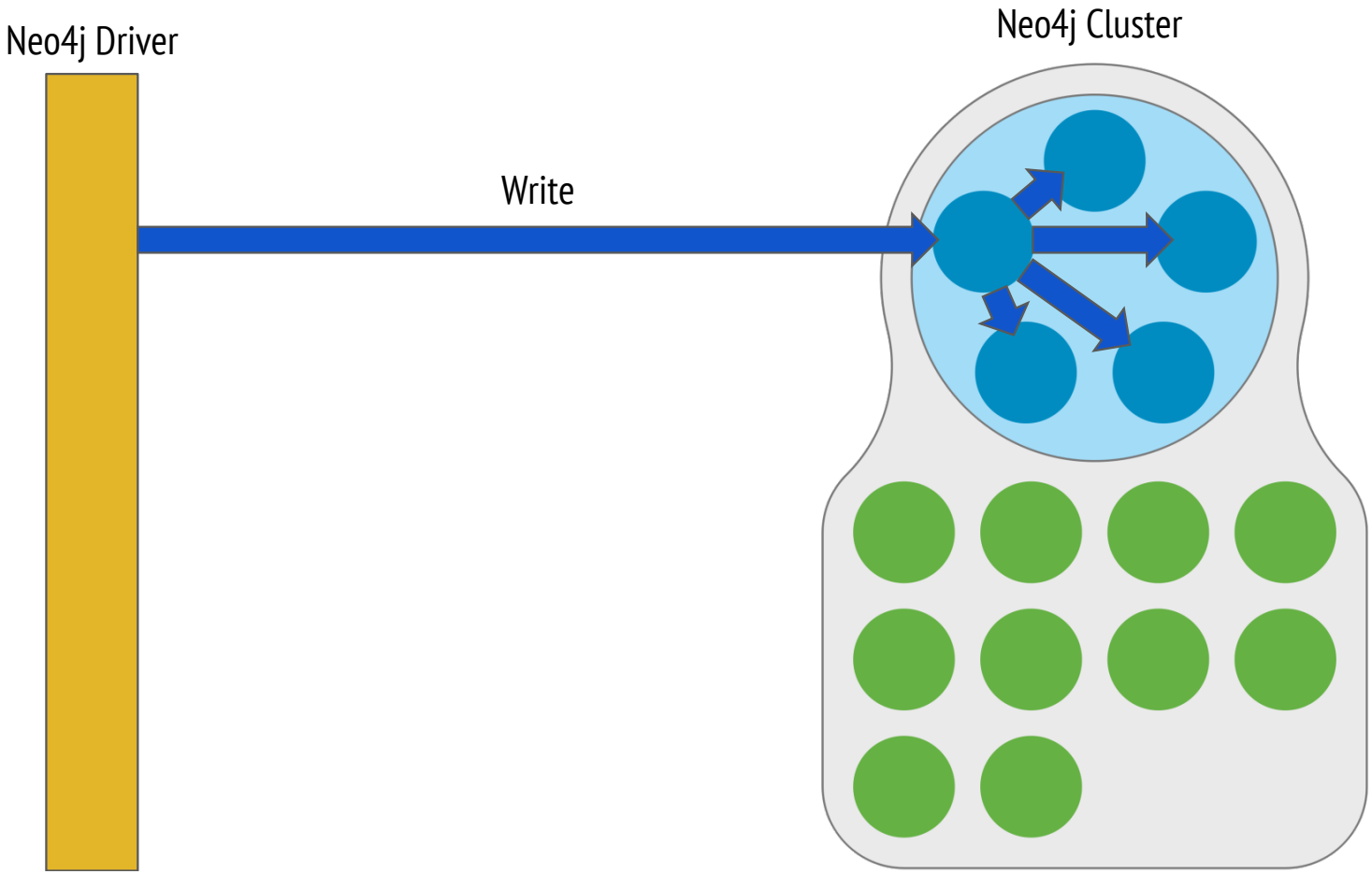
Read Replicas

- For massive query throughput
- Read-only replicas
- Not involved in Consensus Commit
- Disposable, suitable for auto-scaling

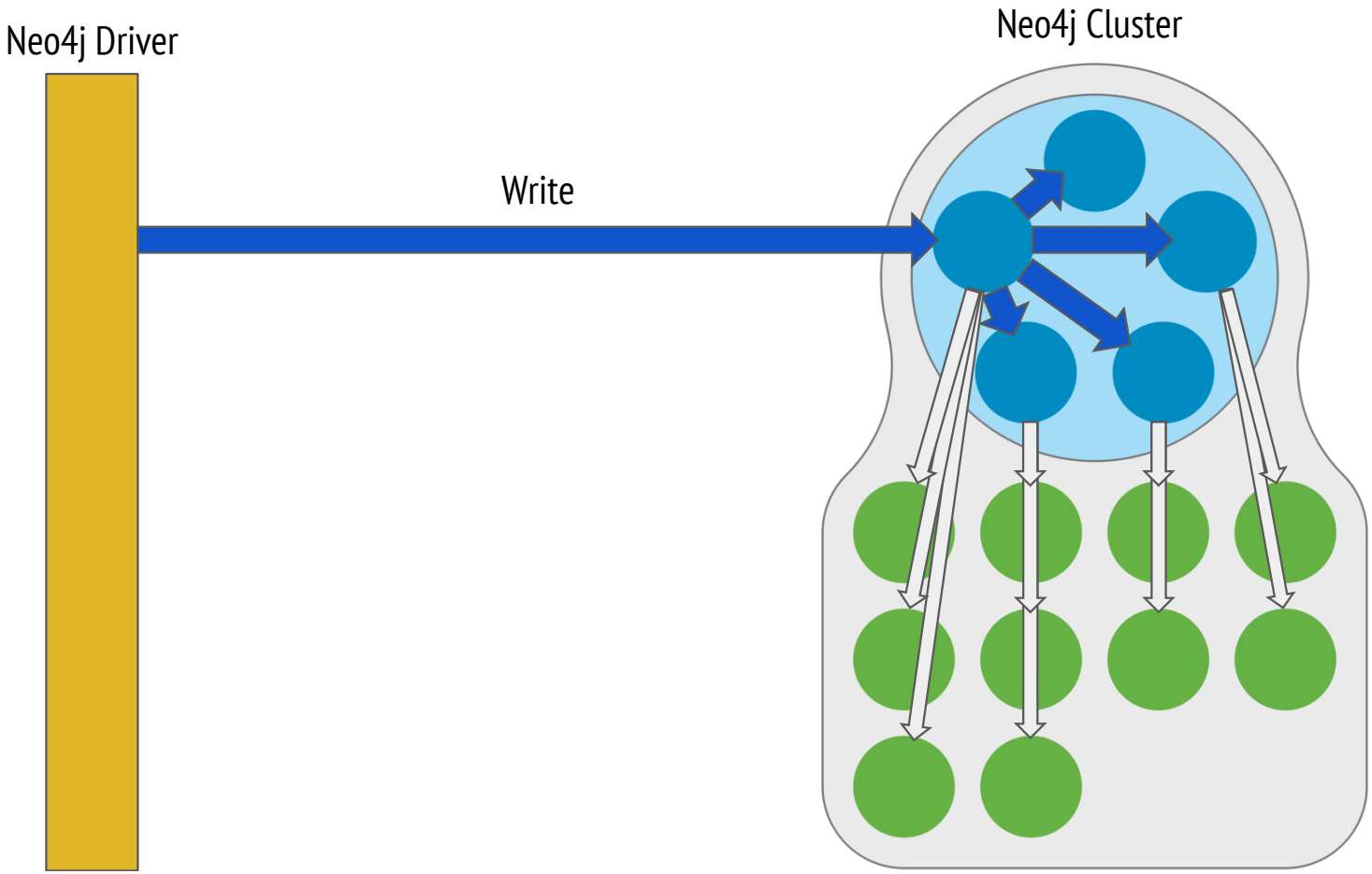
Propagating updates to the Read Replicas



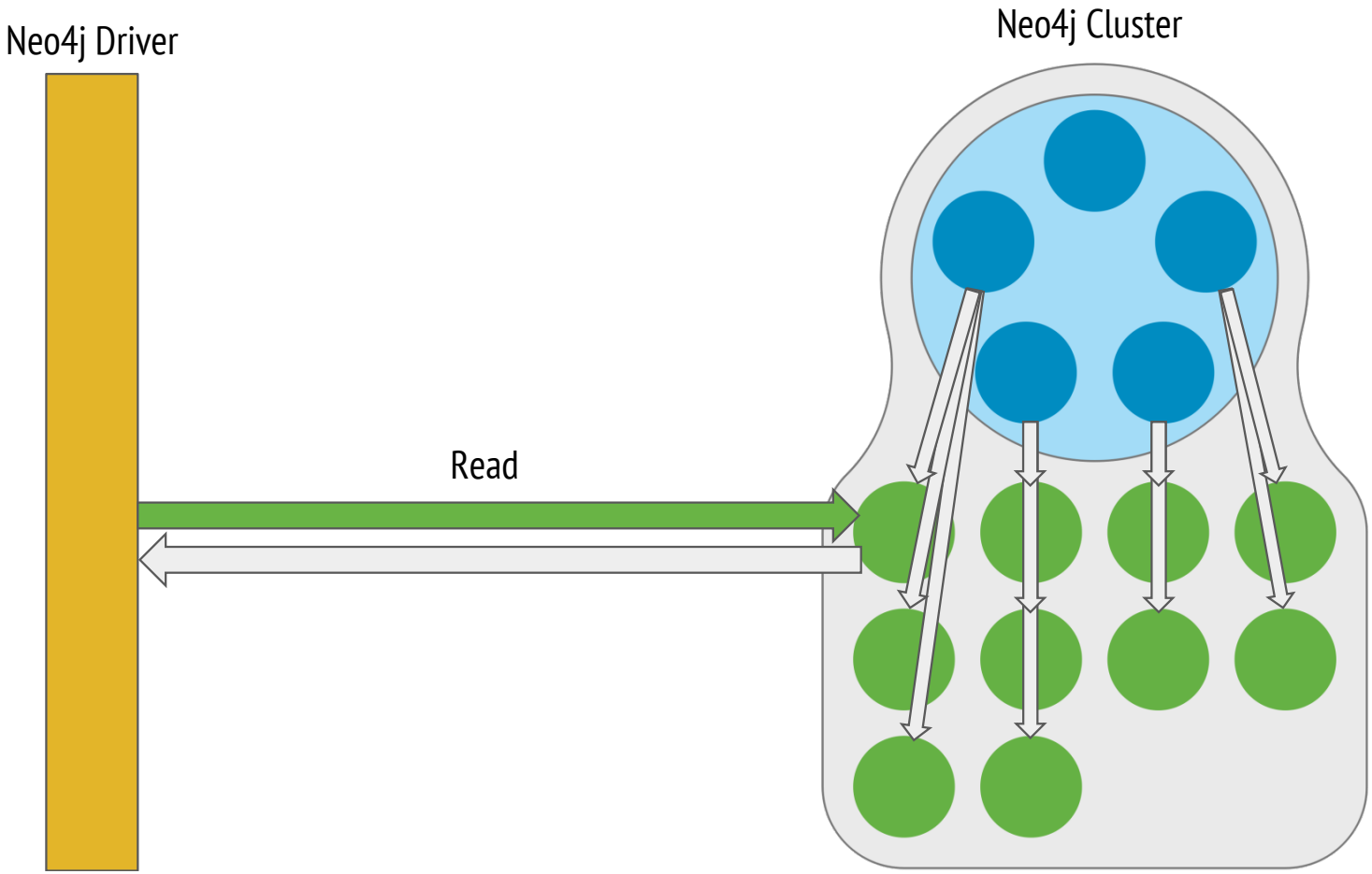
Propagating updates to the Read Replicas

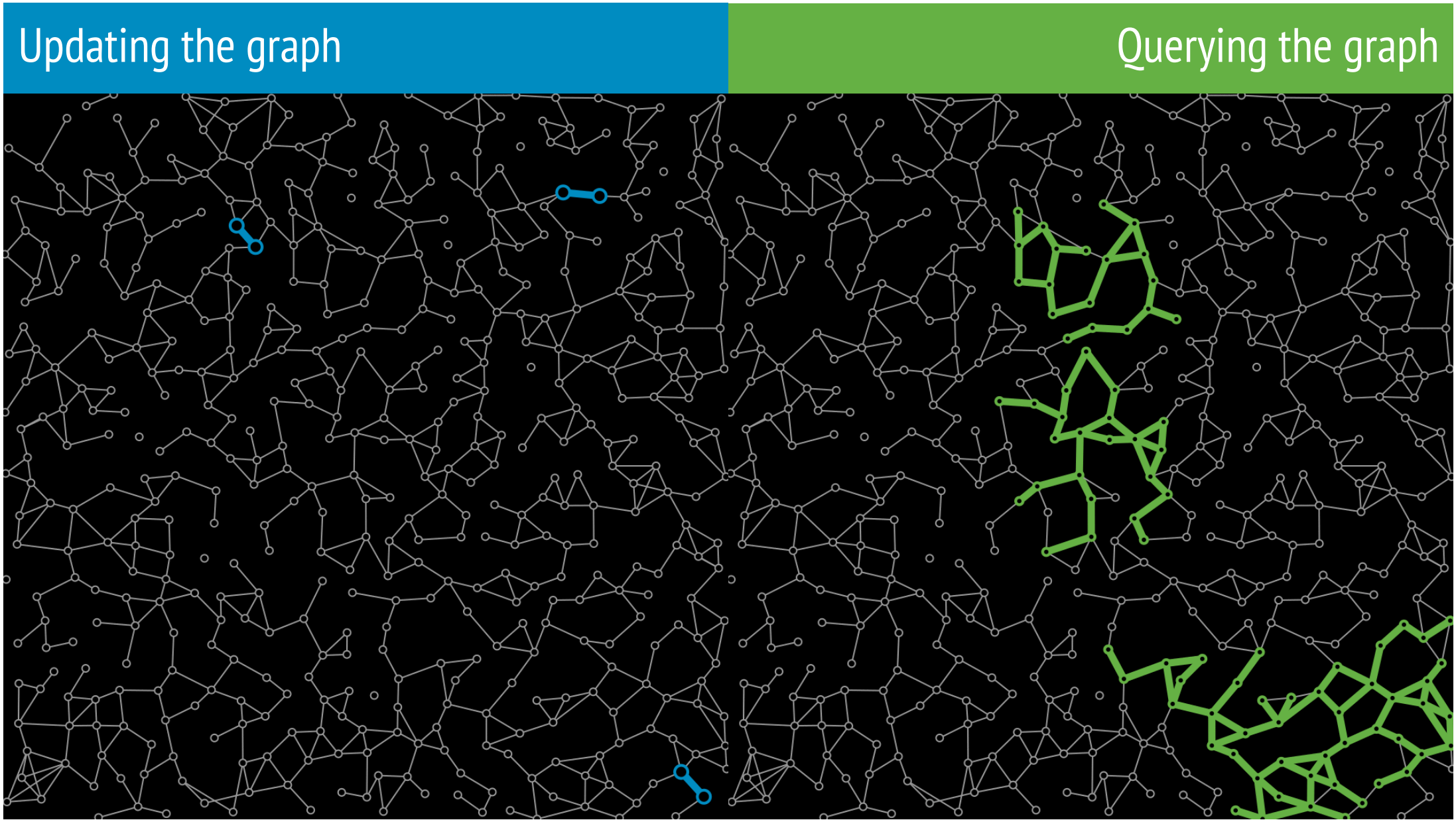


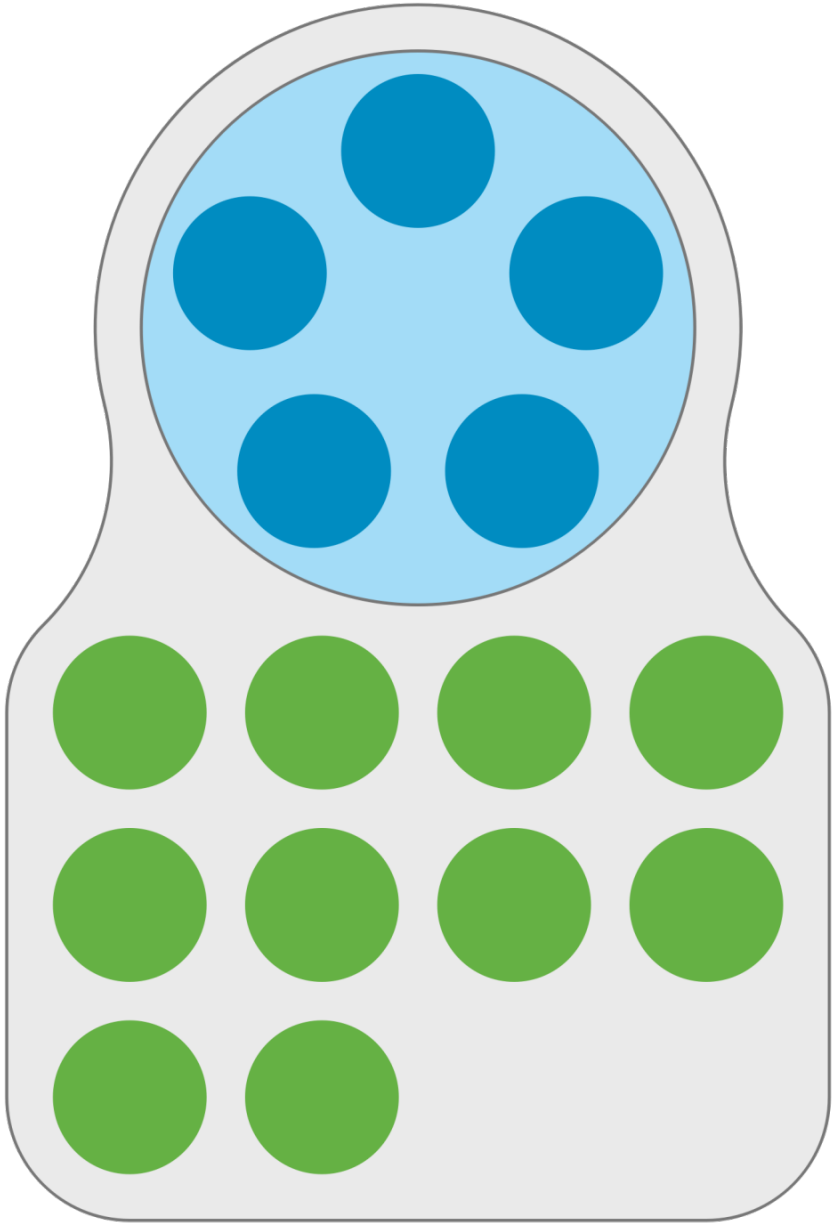
Propagating updates to the Read Replicas



Reading from the Read Replicas




Updating the graph
Querying the graph



Core Updating the graph

Read Replicas Queries, analysis, reporting

```
ESTATE=$(neo-workbench estate add database -p Local -b core-block -s 3)

neo-workbench estate add database -p Local -b edge-block -s 10 $ESTATE

neo-workbench database install -m Core \
  --package-uri file:///Users/jim/Downloads/neo4j-enterprise-3.1.1-unix.tar.gz \
  -b core-block $ESTATE

neo-workbench database install -m Read_Replica \
  --package-uri file:///Users/jim/Downloads/neo4j-enterprise-3.1.1-unix.tar.gz \
  -b edge-block $ESTATE

neo-workbench database start $ESTATE
```

SKILLSCAST

Neo4j Container Orchestration with Kubernetes, Docker Swarm, Mesos

cloud | graph-databases | Graphs | data-mining | bigdata | neo4j

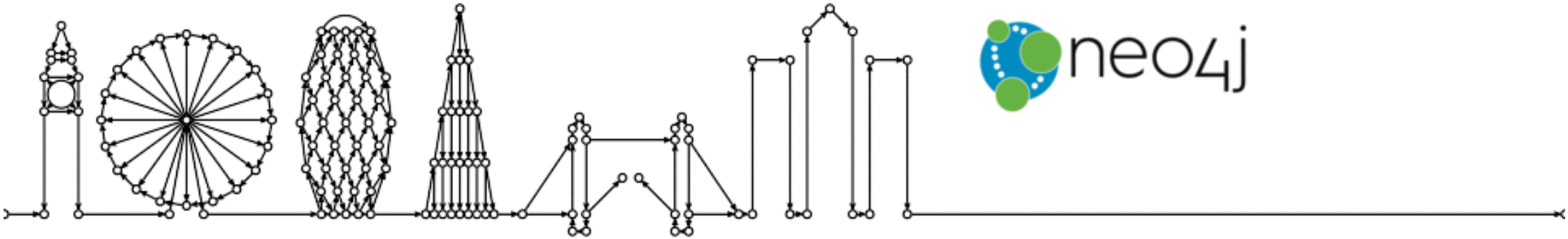
:sysinfo

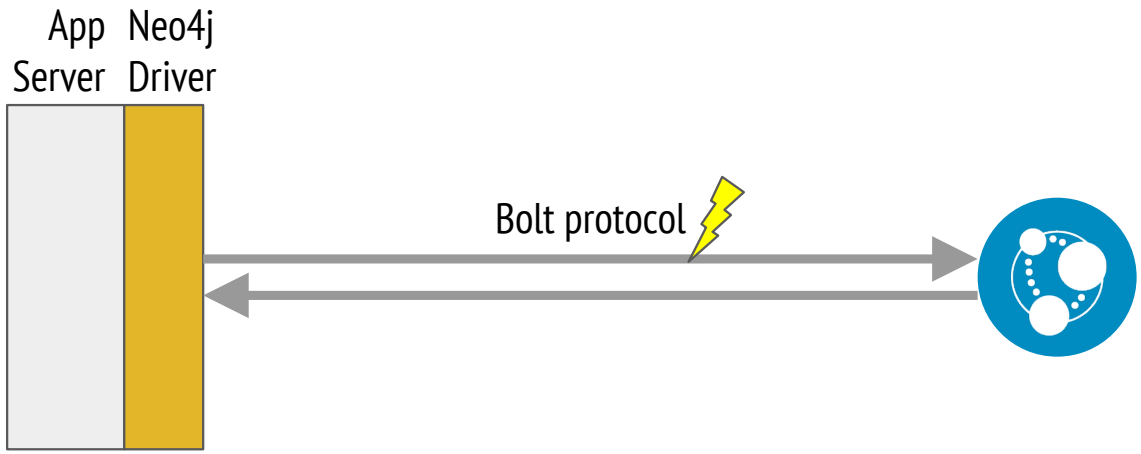
Causal Cluster Members

Role	Addresses	Actions
LEADER	bolt://localhost:26000, http://localhost:7474, https://localhost:22000	Open
FOLLOWER	bolt://localhost:26001, http://localhost:7475, https://localhost:22001	Open
FOLLOWER	bolt://localhost:26002, http://localhost:7476, https://localhost:22002	Open
FOLLOWER	bolt://localhost:26003, http://localhost:7477, https://localhost:22003	Open
FOLLOWER	bolt://localhost:26004, http://localhost:7478, https://localhost:22004	
READ_REPLICA	bolt://localhost:26006, http://localhost:7480, https://localhost:22006	Open
READ_REPLICA	bolt://localhost:26007, http://localhost:7481, https://localhost:22007	Open
READ_REPLICA	bolt://localhost:26008, http://localhost:7482, https://localhost:22008	Open
READ_REPLICA	bolt://localhost:26009, http://localhost:7483, https://localhost:22009	Open
READ_REPLICA	bolt://localhost:26010, http://localhost:7484, https://localhost:22010	Open
READ_REPLICA	bolt://localhost:26012, http://localhost:7486, https://localhost:22012	Open

Building an App

Computer science meets technology




App
Server
Neo4j
Driver
Bolt protocol

Java

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
</dependency>
```

Python

```
pip install neo4j-driver
```

.NET

```
PM> Install-Package Neo4j.Driver
```

JavaScript

```
npm install neo4j-driver
```

Developer: Languag

https://neo4j.com/developer/language-guides/

Blog Support Contact Us Download Neo4j

neo4j PRODUCTS SOLUTIONS PARTNERS CUSTOMERS LEARN DEVELOPERS Search

Language Guides

Developer → Develop with Neo4j

- Get Started
- Cypher Query Language
- Data Modeling
- Working with Data
- Drivers & Language Guides
  - Java
  - Spring Framework
  - .NET
  - JavaScript
  - Python

Language Guides

These guides and tutorials are designed to provide detailed examples of how to integrate Neo4j with your preferred programming language. Neo4j **officially supports the drivers for .Net, Java, JavaScript and Python** for the binary Bolt protocol. Our community contributors provide drivers for all major programming languages for all protocols and APIs. In this section, we provide an introduction and a consistent example application for several languages and Neo4j drivers.

https://neo4j.com/developer/language-guides

bolt://

```
GraphDatabase.driver( "bolt://aServer" )
```

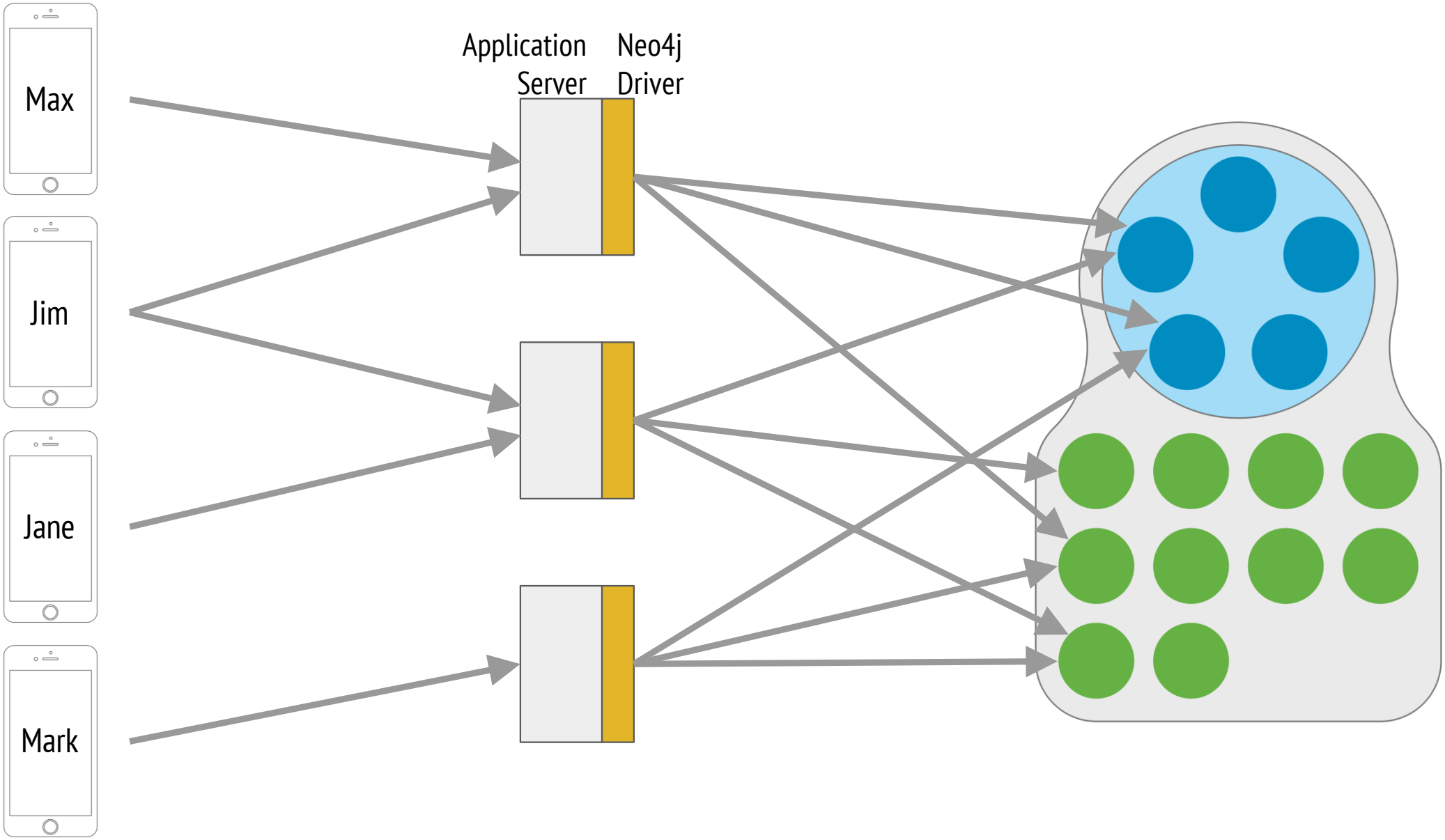
bolt+routing://

```
GraphDatabase.driver( "bolt+routing://aCoreServer" )
```

bolt+routing://

```
GraphDatabase.driver( "bolt+routing://aCoreServer" )
```

Bootstrap: specify any core server to route load across the whole cluster


Max
Jim
Jane
Mark
Application Server
Neo4j Driver

Routed write statements

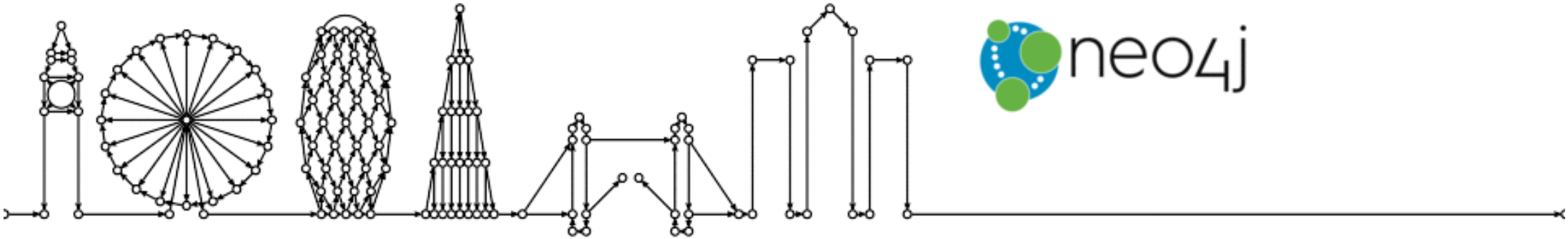
```
driver = GraphDatabase.driver( "bolt+routing://aCoreServer" );

try ( Session session = driver.session( AccessMode.WRITE ) )
{
   try ( Transaction tx = session.beginTransaction() )
   {
       tx.run( "MERGE (user:User {userId: {userId}})",
               parameters( "userId", userId ) );

       tx.success();
   }
}
```

Routed read queries

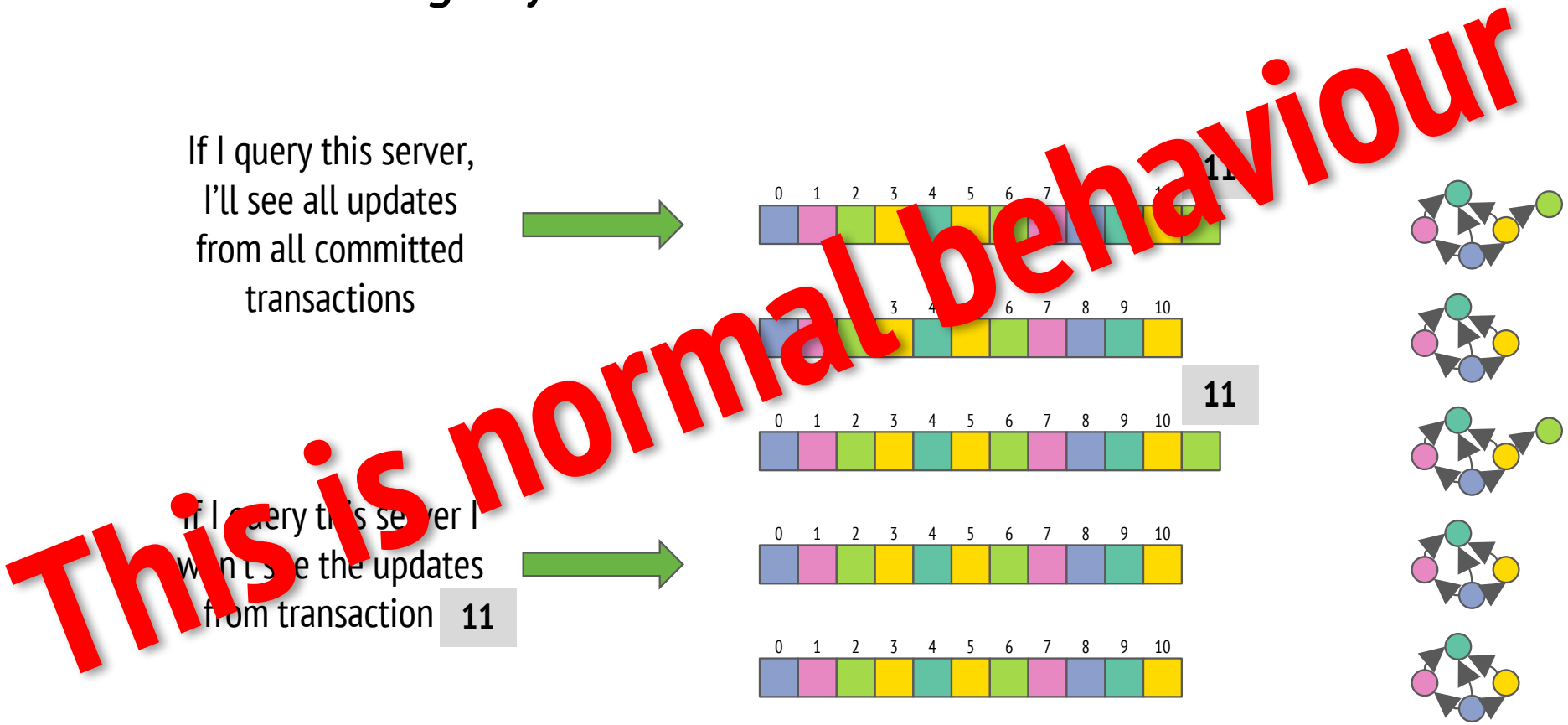
```
driver = GraphDatabase.driver( "bolt+routing://aCoreServer" );

try ( Session session = driver.session( AccessMode.READ ) )
{
   try ( Transaction tx = session.beginTransaction() )
   {
       tx.run( "MATCH (user:User {userId: {userId}})-[*]-(:Product) RETURN *",
               parameters( "userId", userId ) );

       tx.success();
   }
}
```

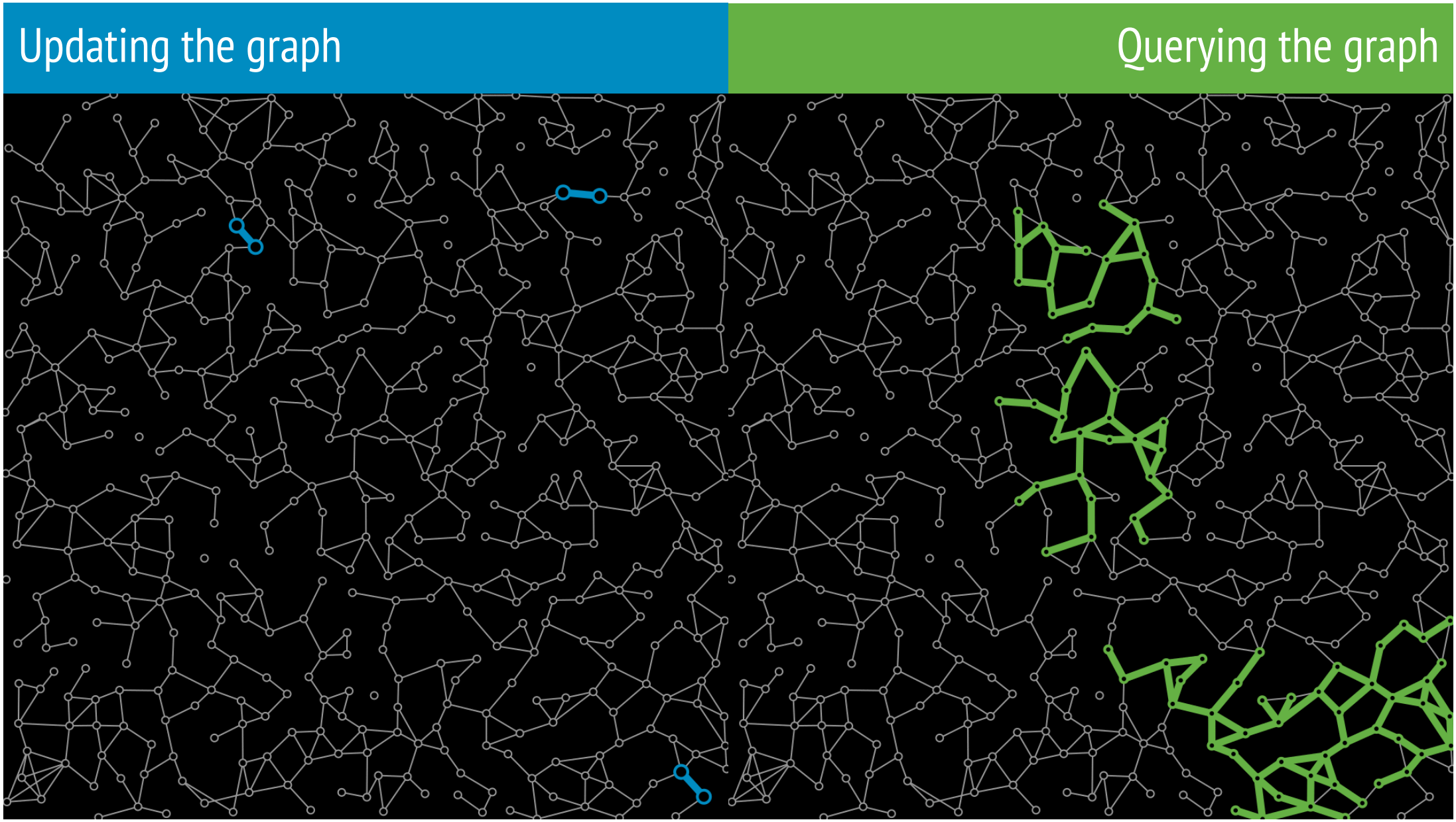
Consistency models

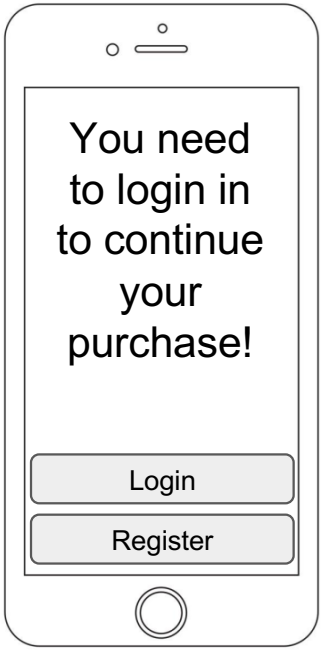
Can you read what you write?

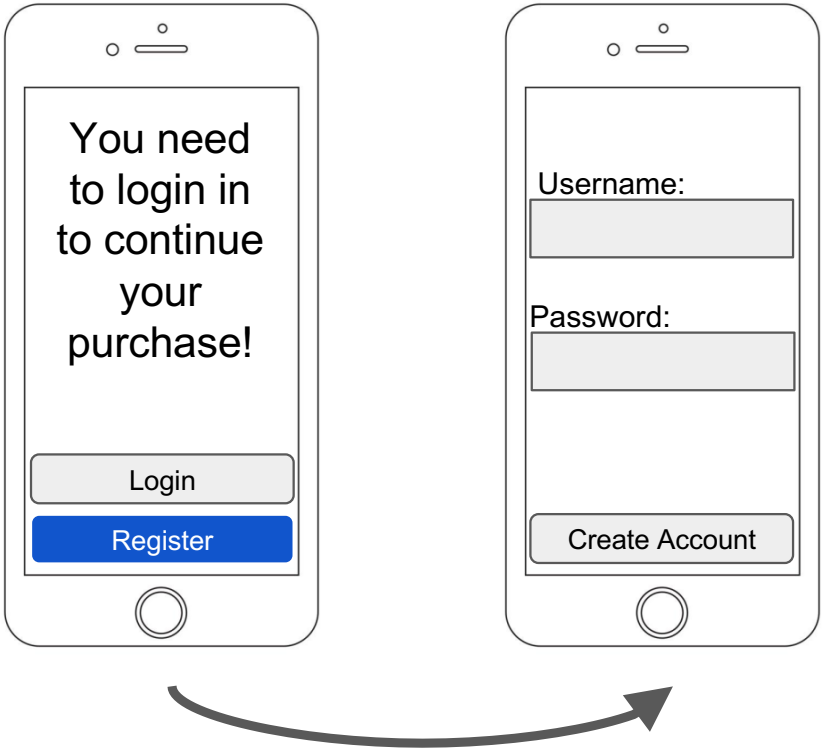


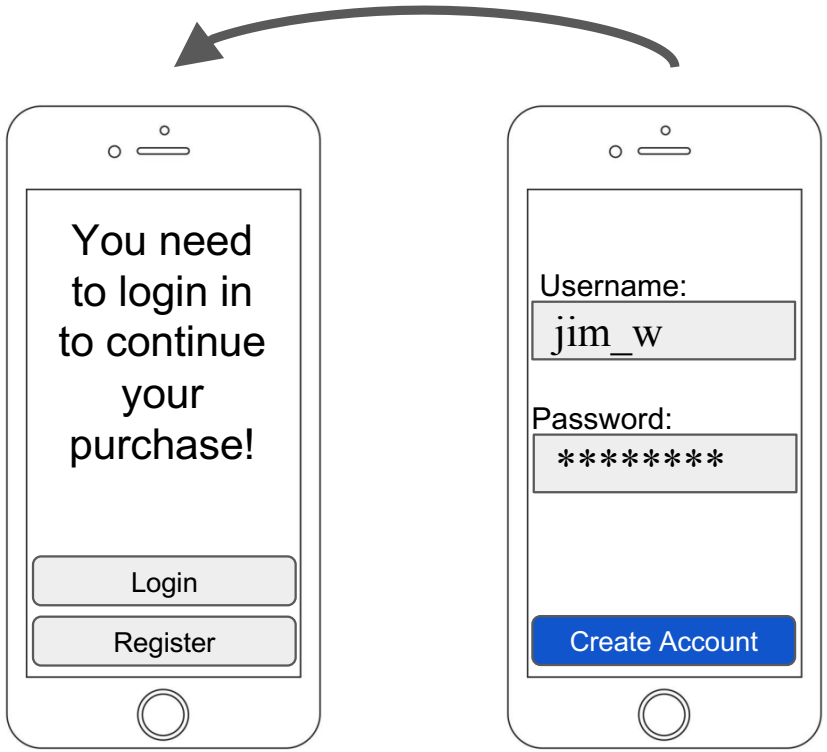
Cluster members slightly “ahead” or “behind” of each other

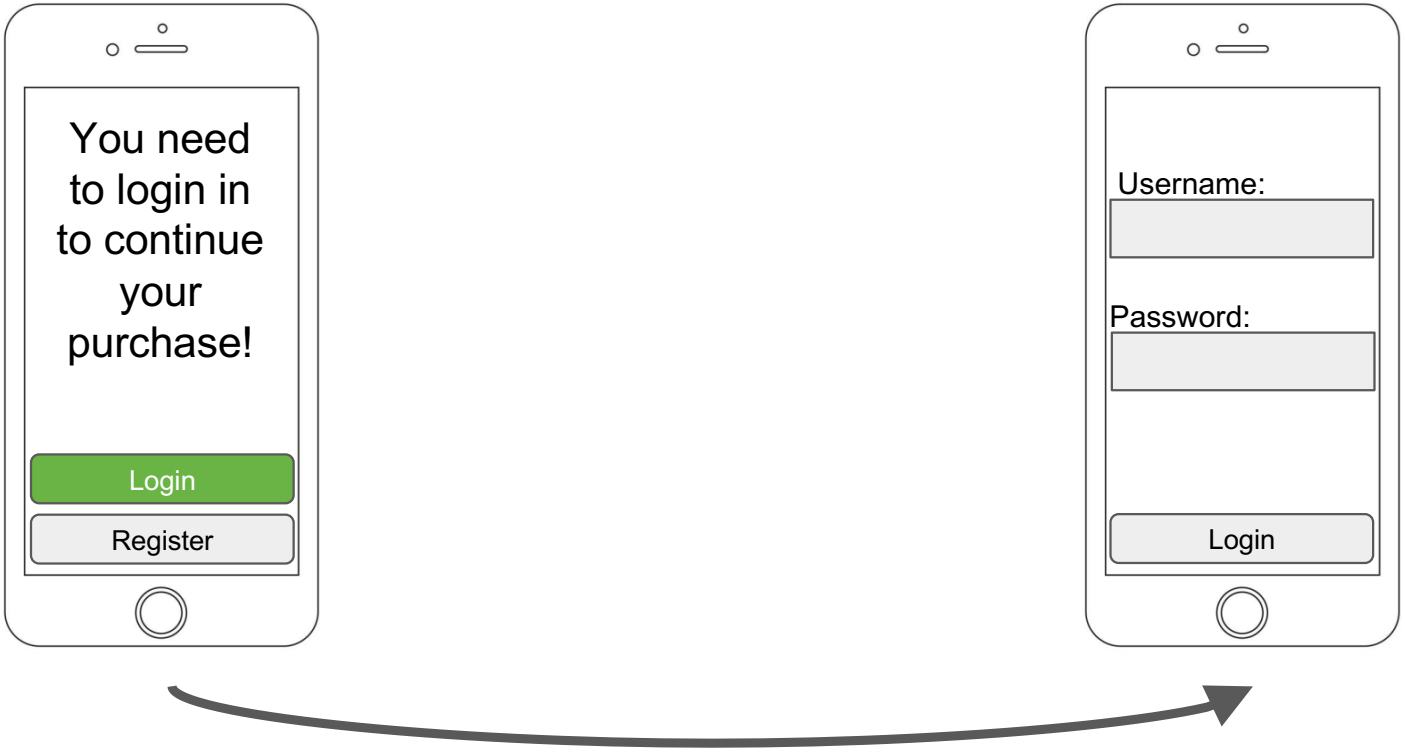


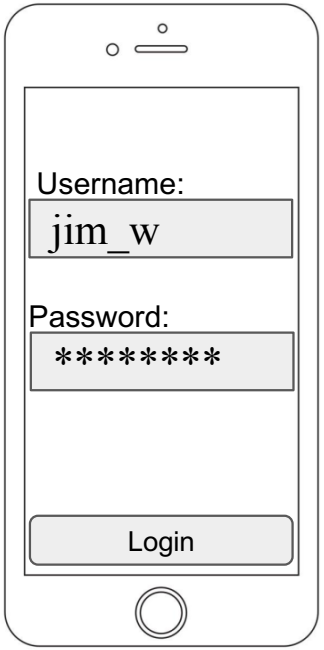

Updating the graph
Querying the graph

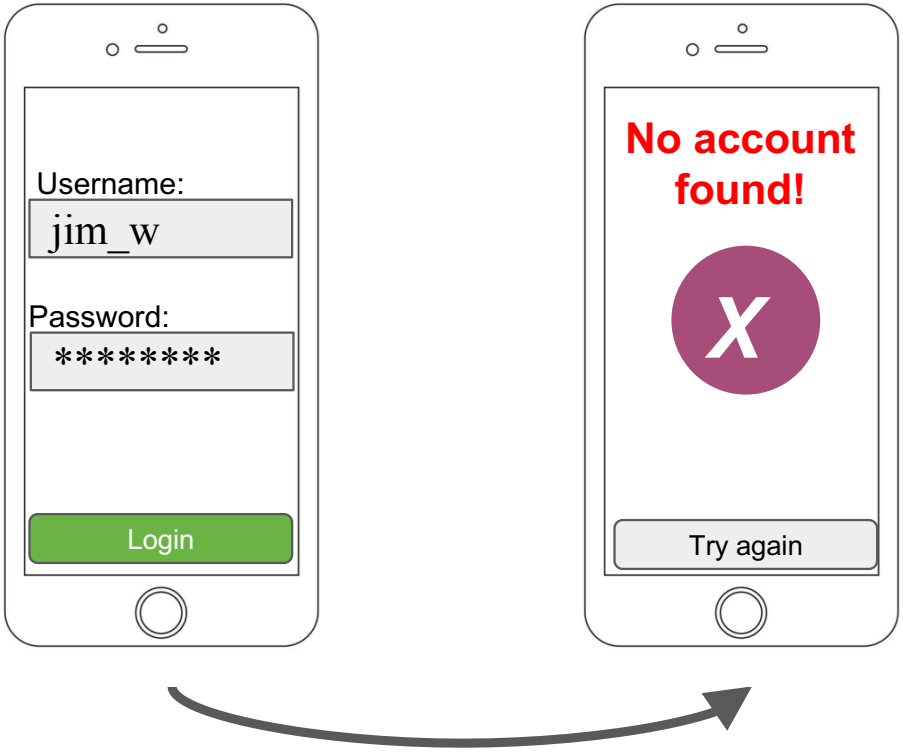

You need to login in to continue your purchase!
Login
Register


You need to login in to continue your purchase!
Login
Register
Username:
Password:
Create Account

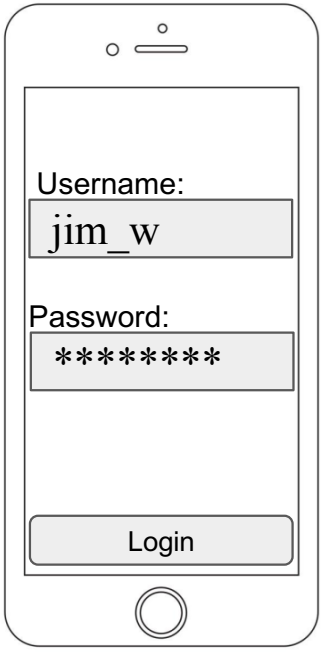

You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Create Account


You need to login in to continue your purchase!
Login
Register
Username:
Password:
Login

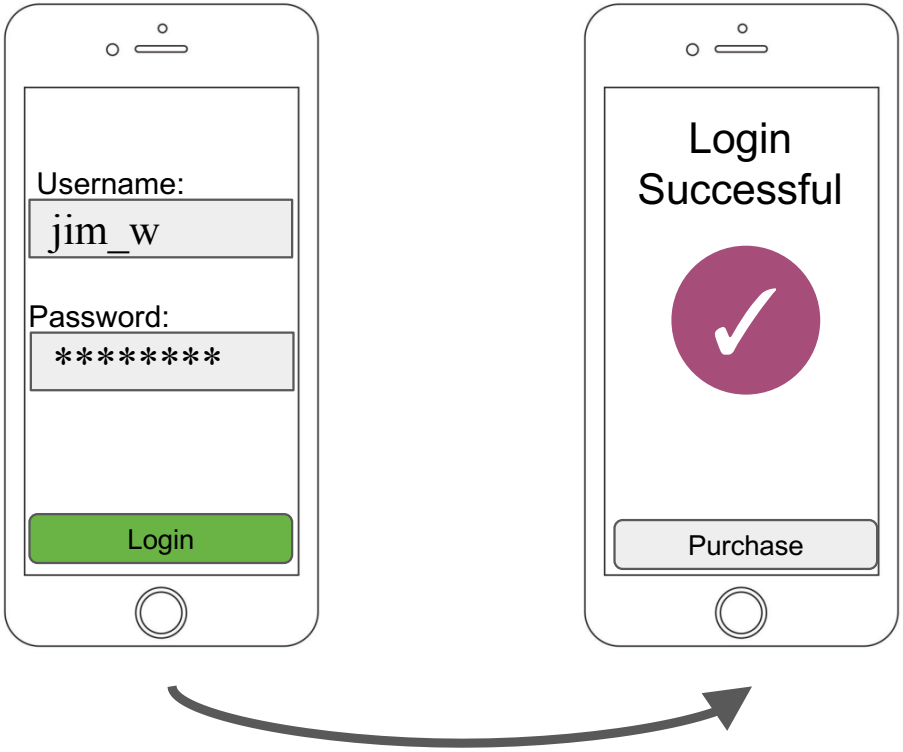

Username:
jim_w
Password:
****
Login


Username:
jim_w
Password:
****
Login
No account found!
X
Try again

A few moments later...

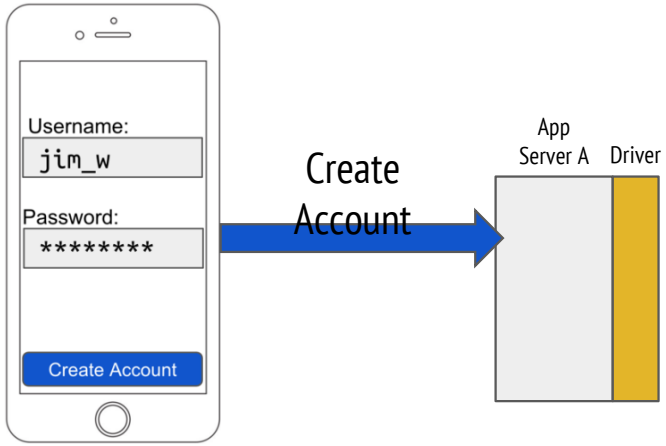


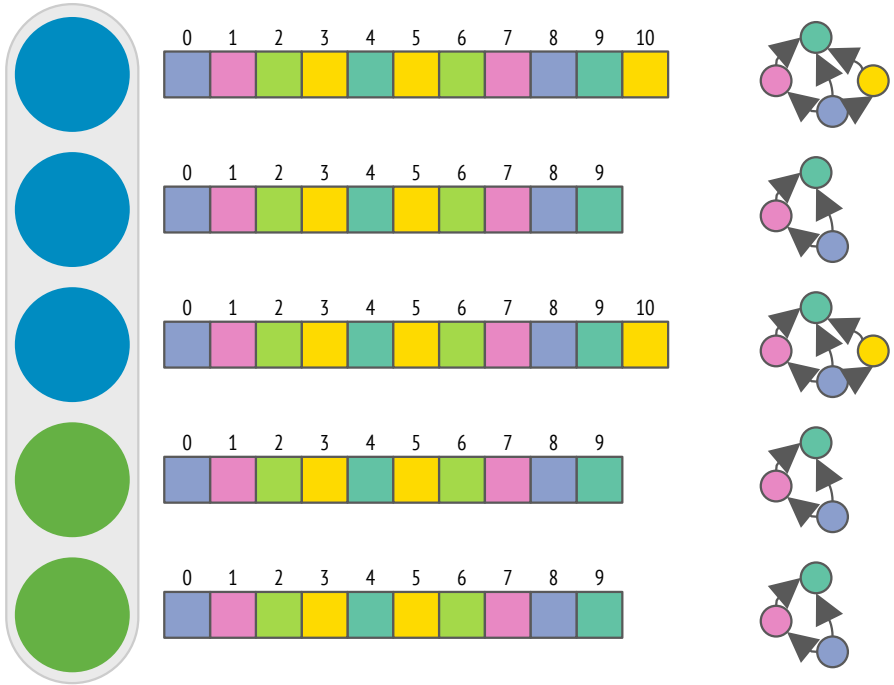
A few moments later...

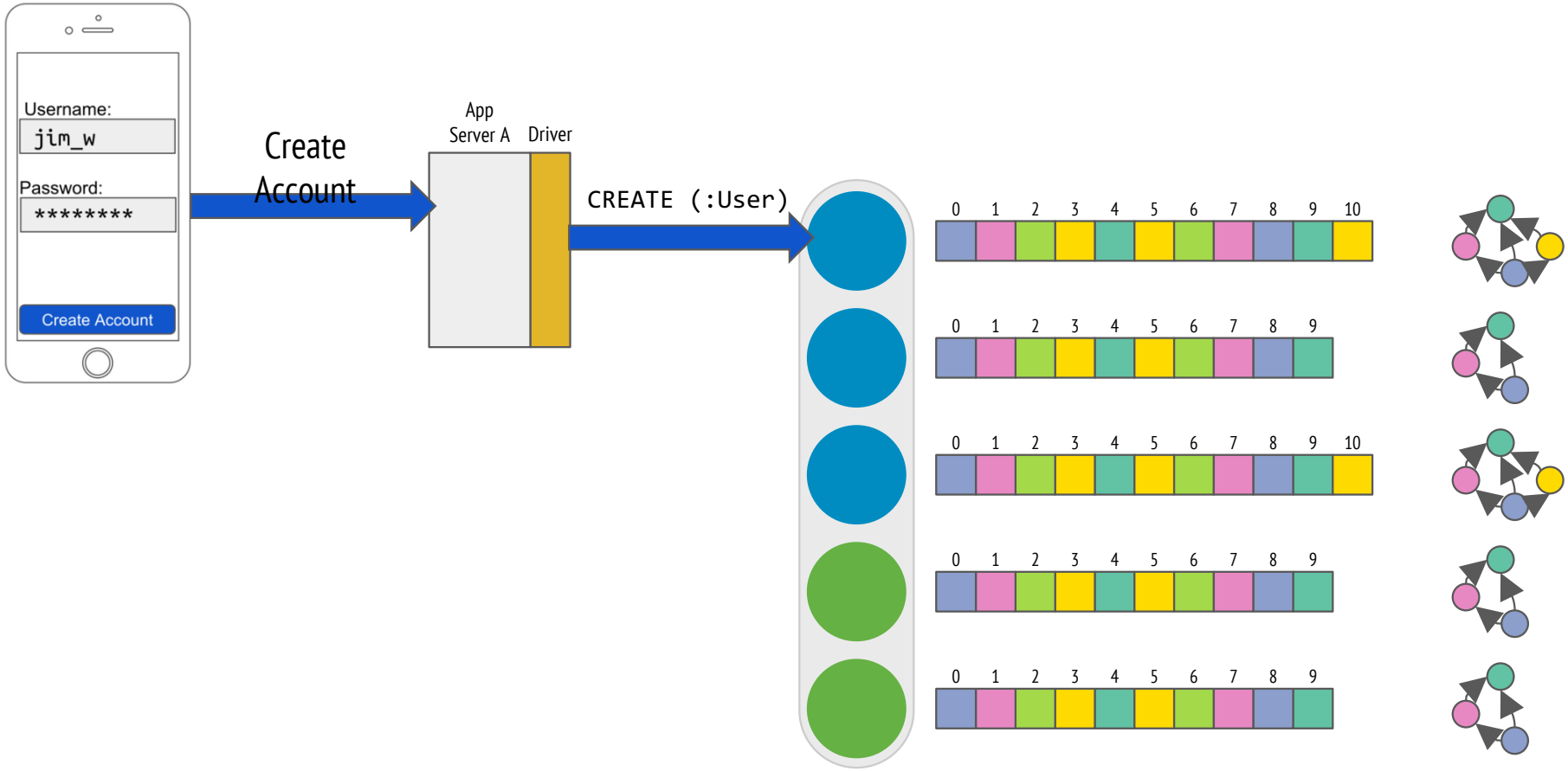


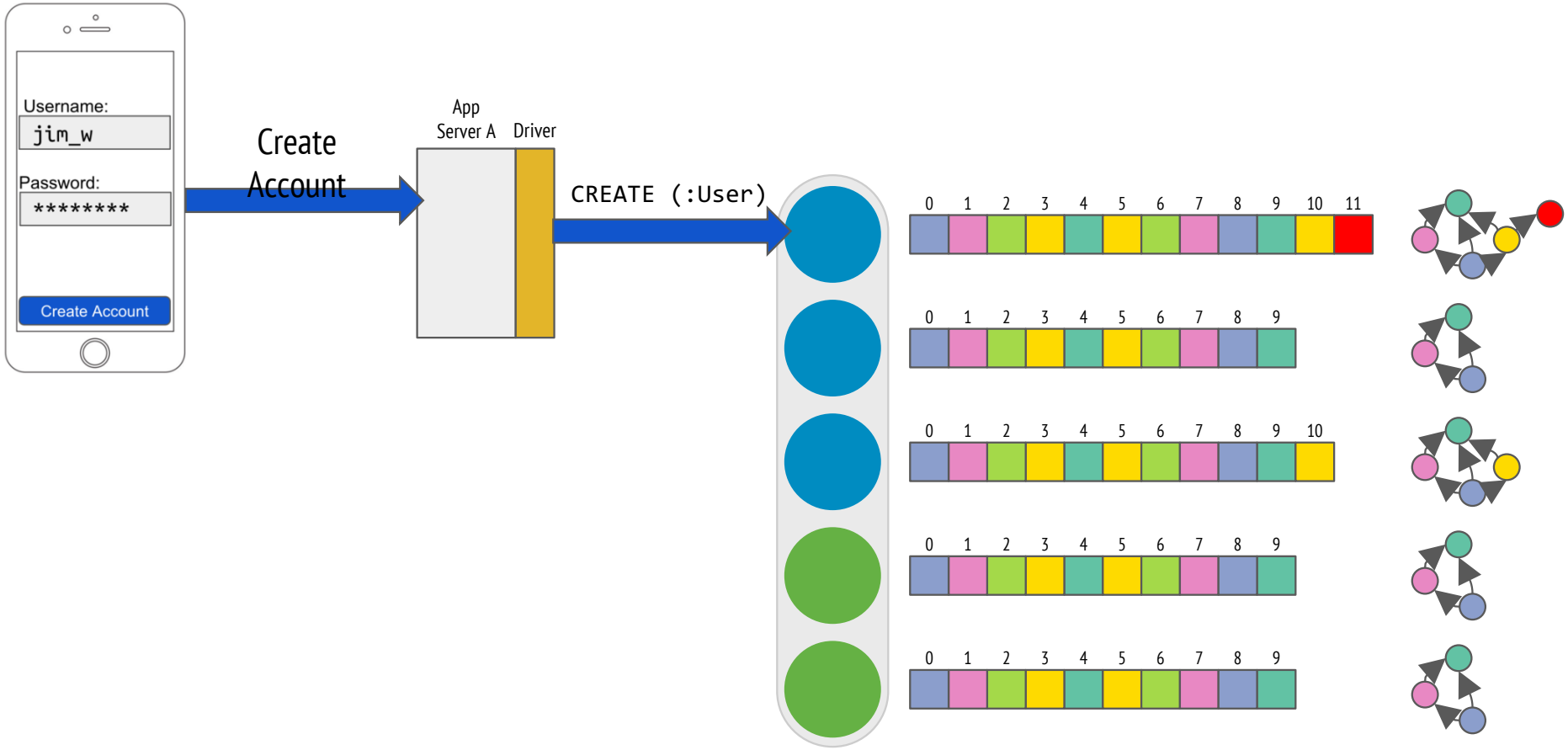
Q Why didn’t this work?

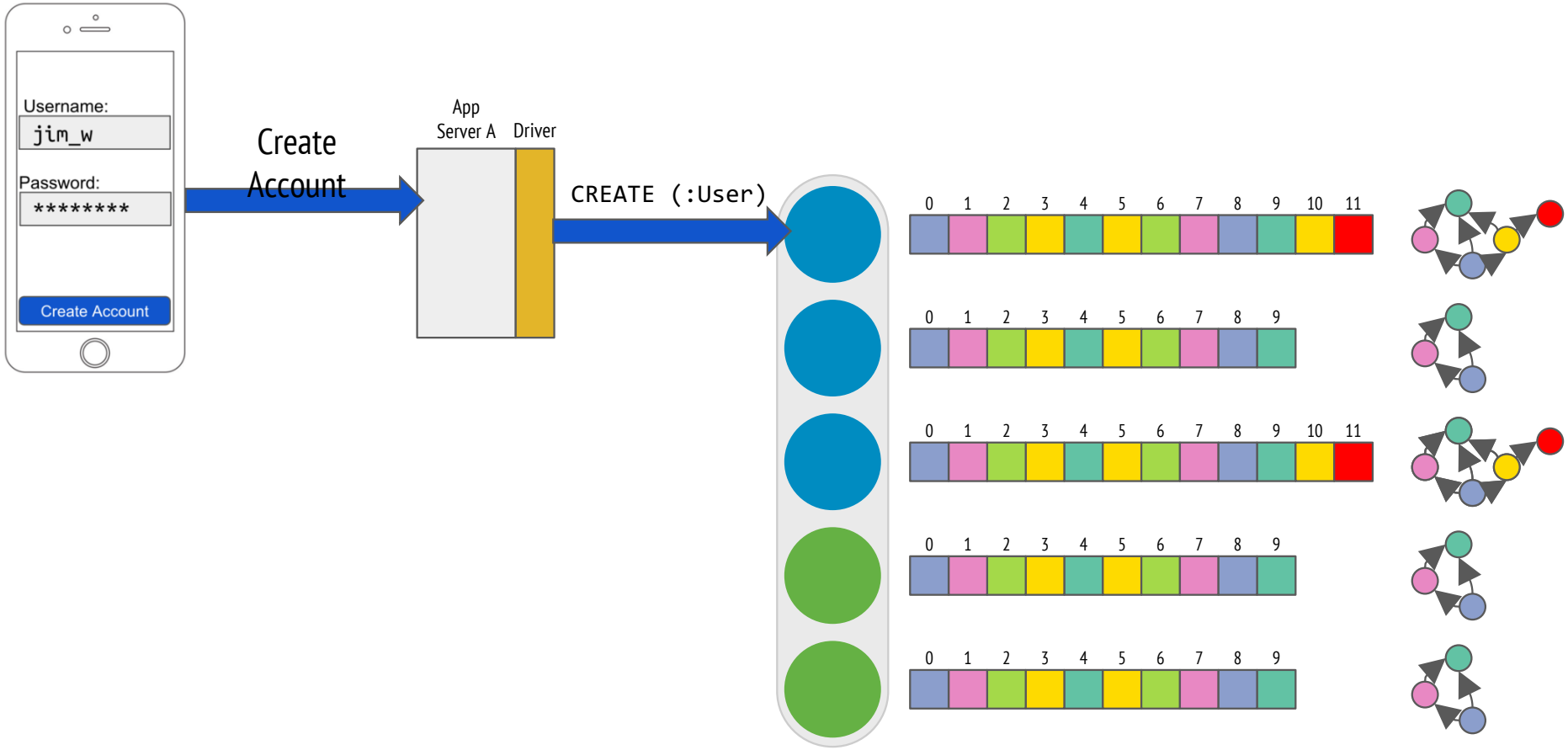
A Eventual Consistency

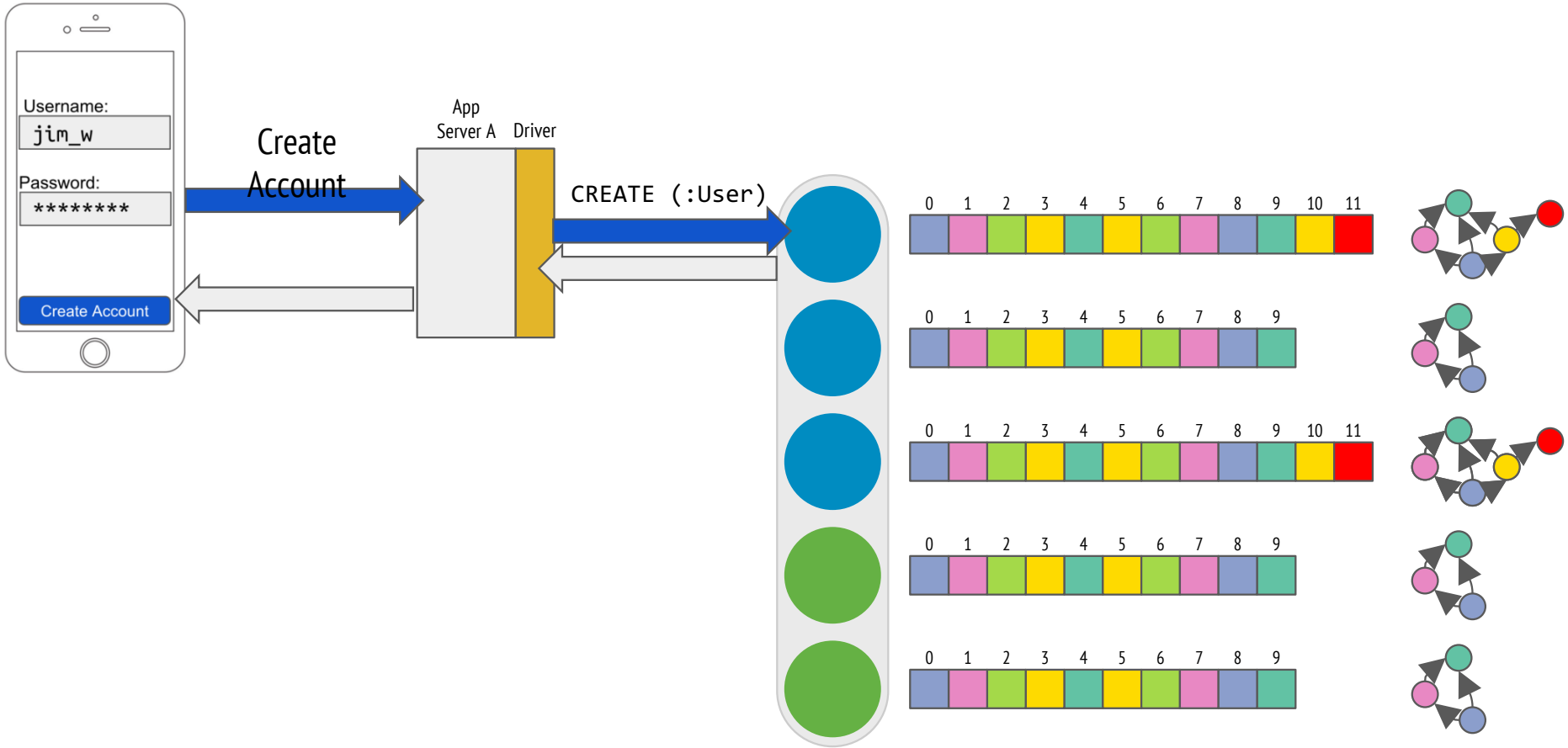

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver

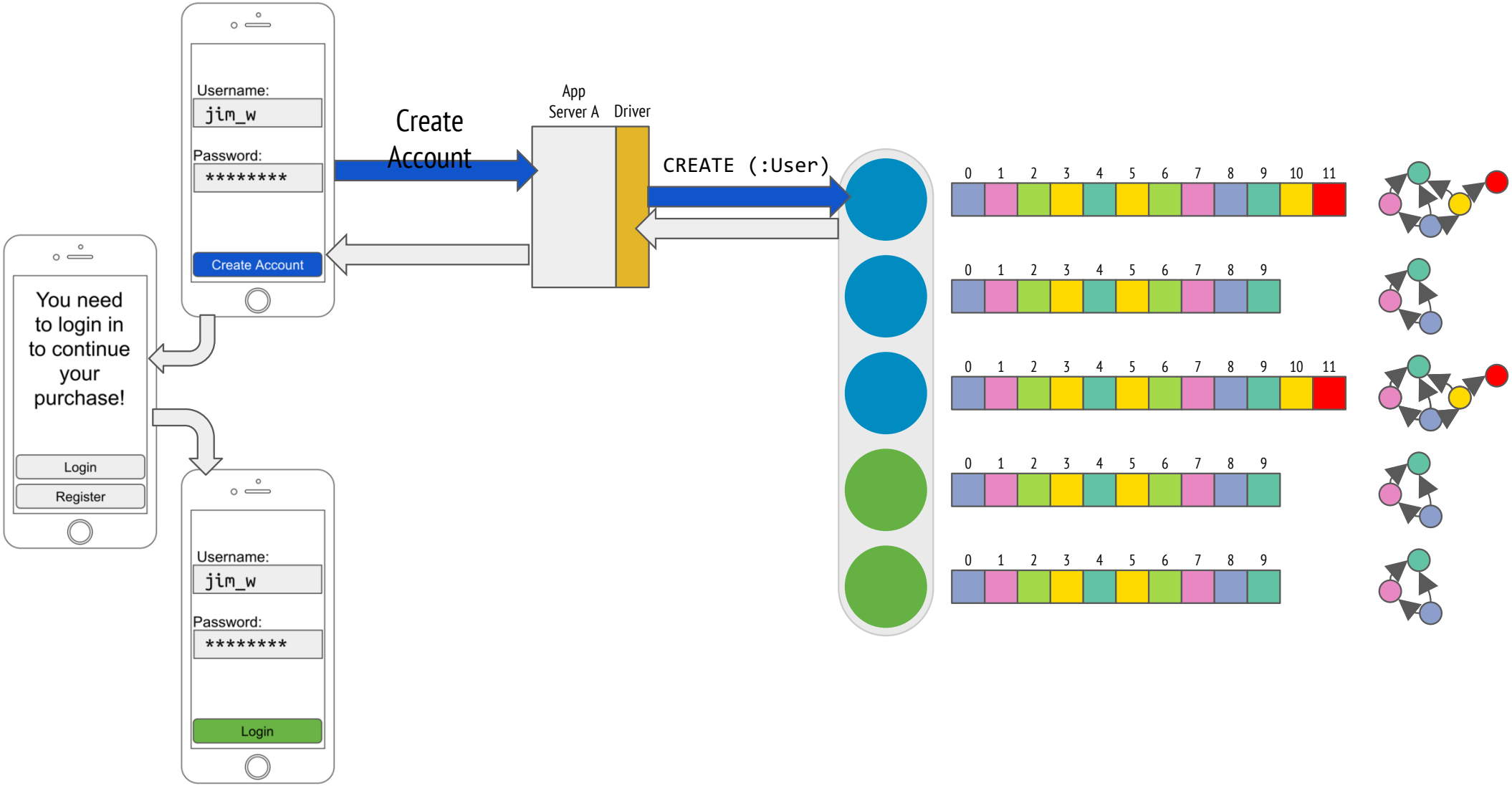

0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

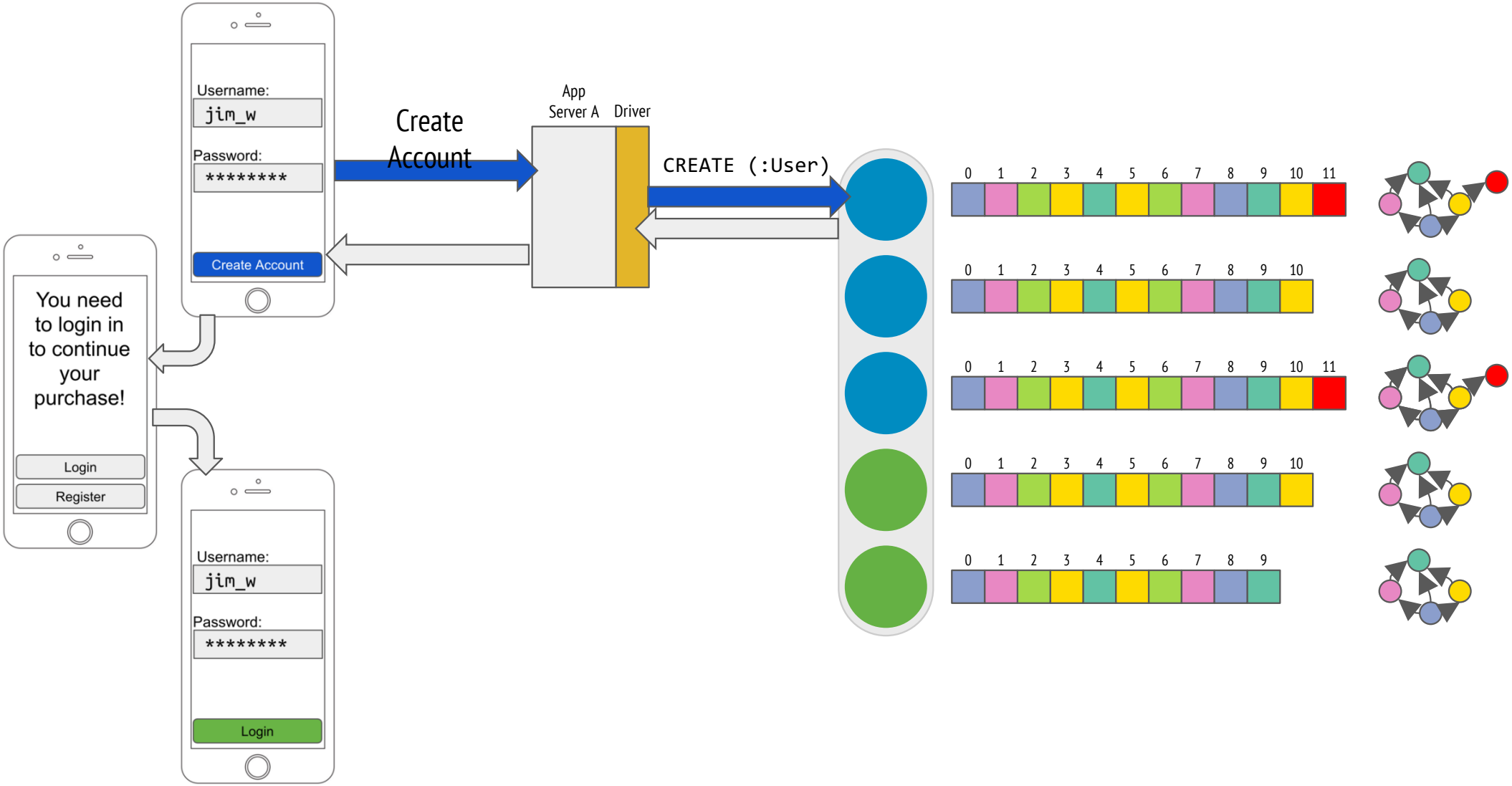

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

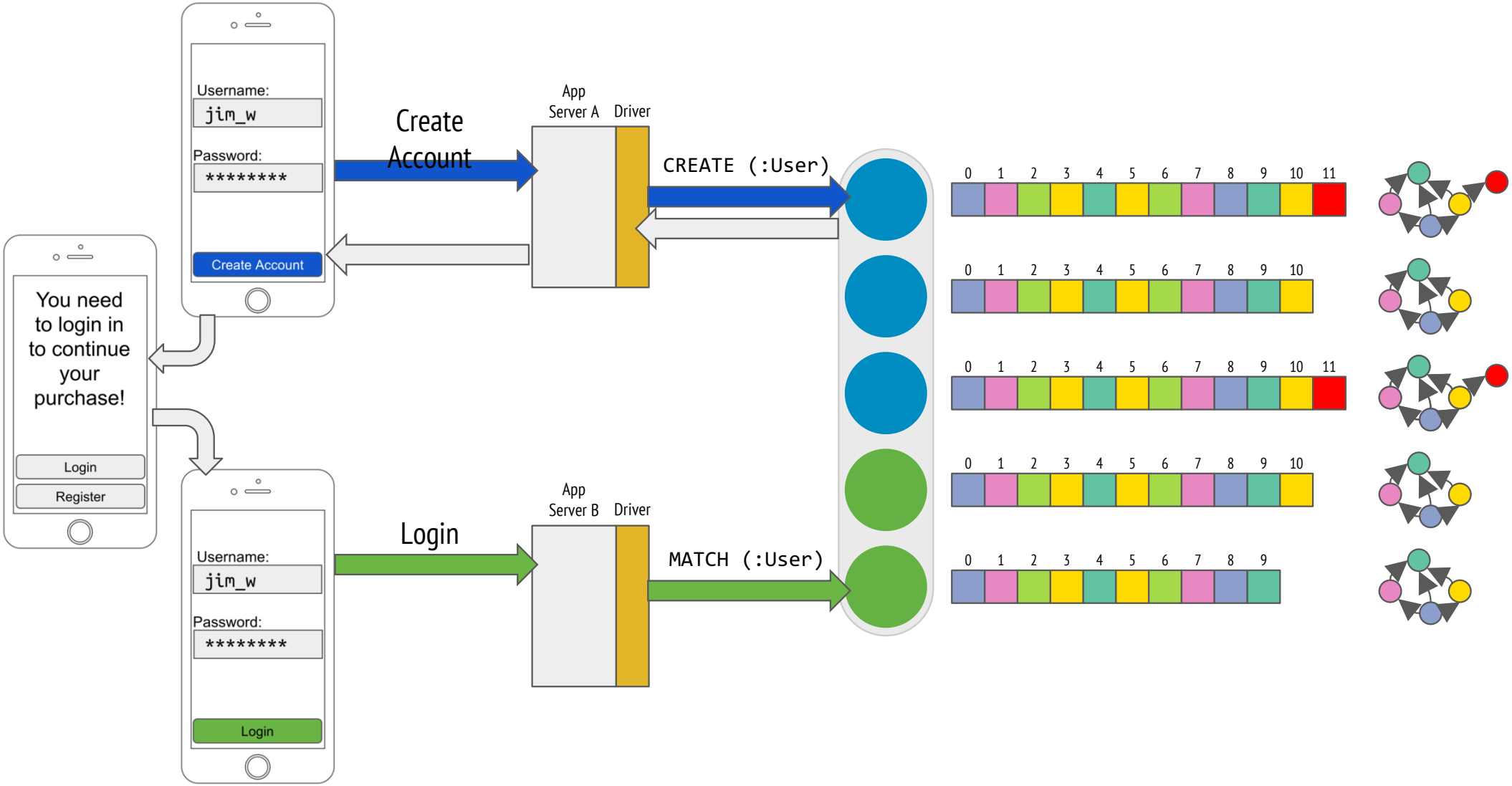

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

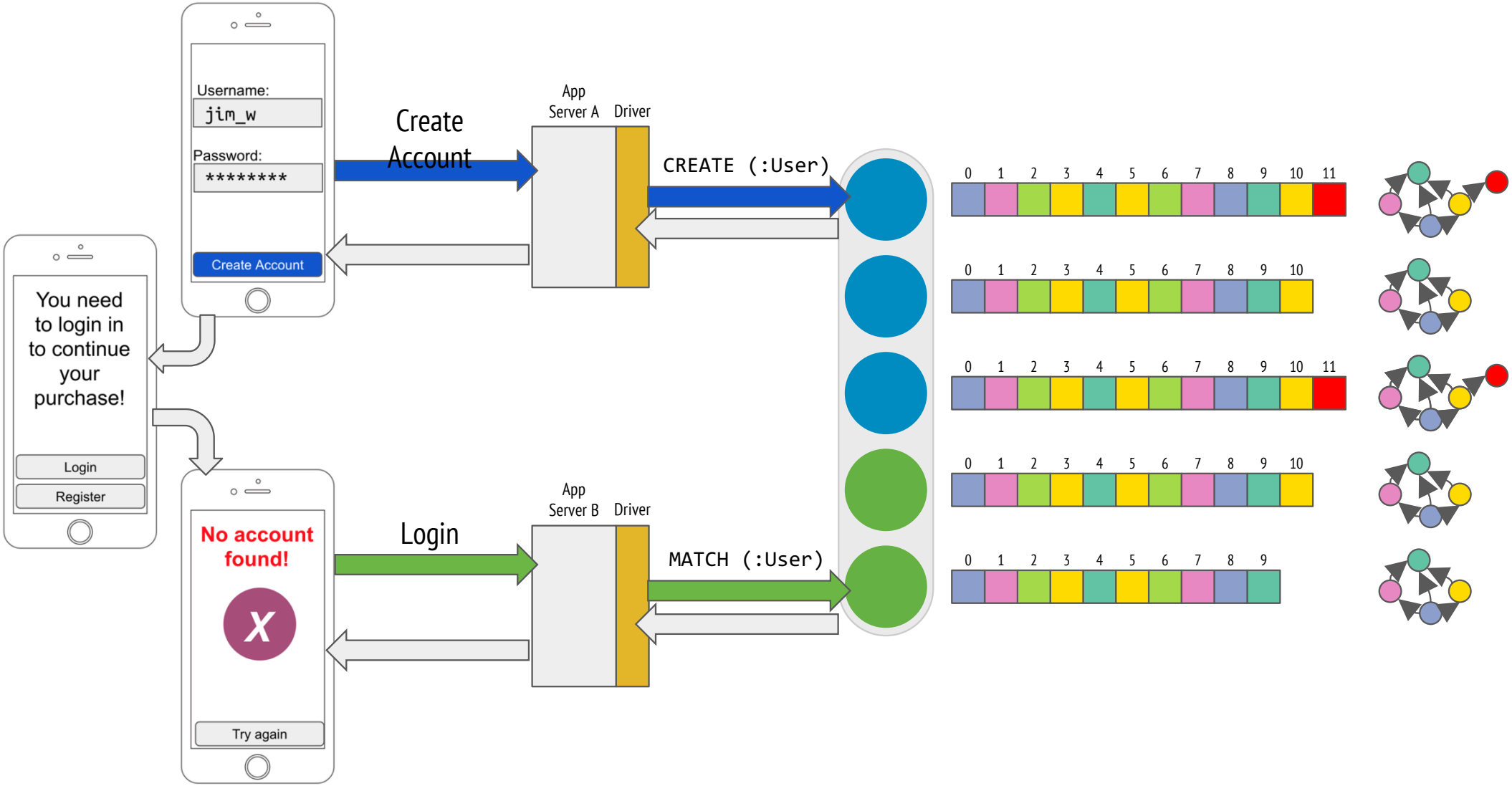

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

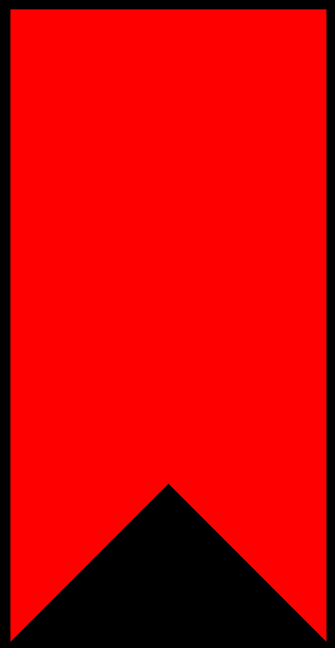

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
No account found!
X
Try again
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9

Bookmark



Session token

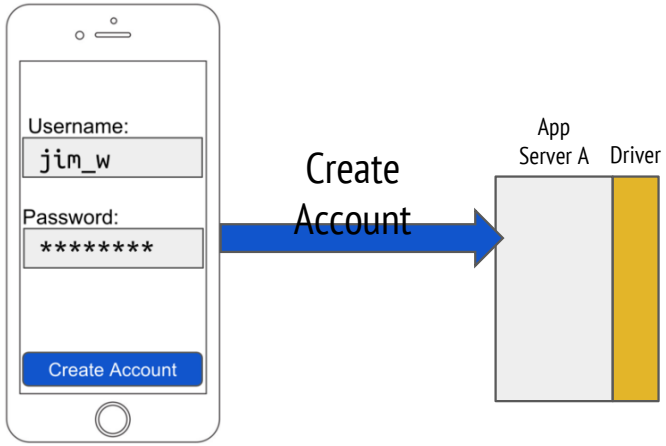
String (for portability)

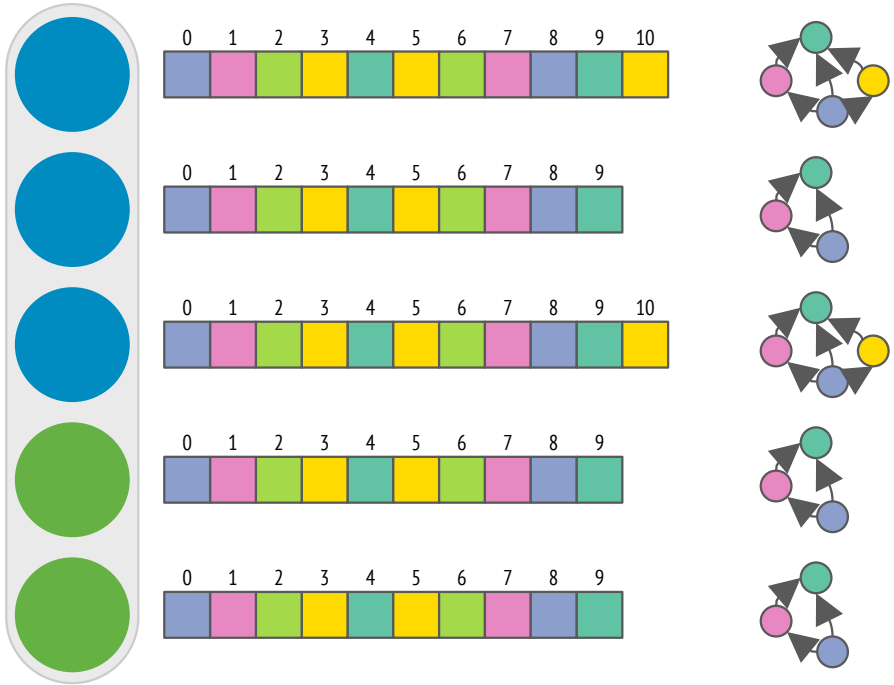
Opaque to application

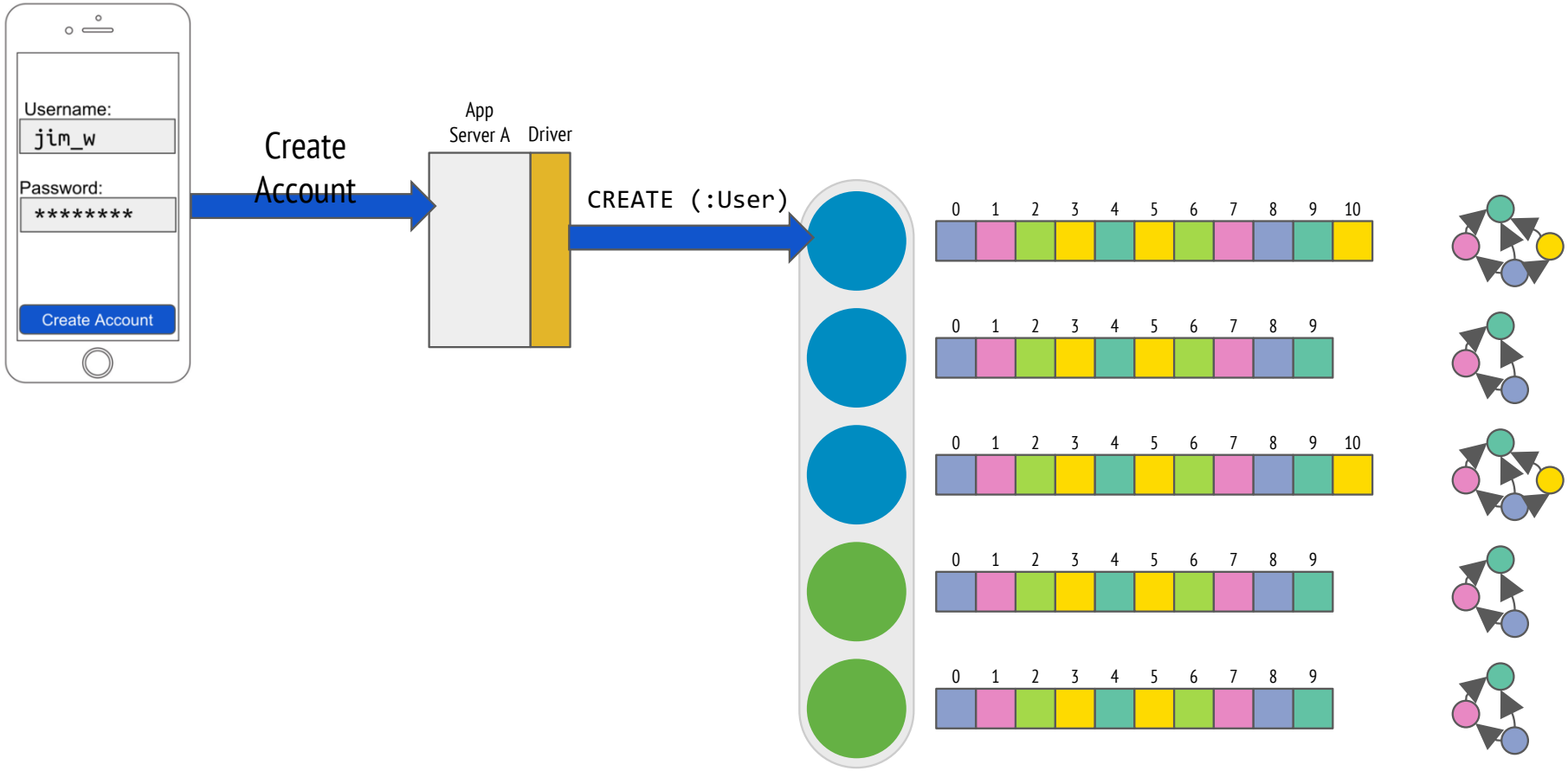
Represents ultimate user's most recent view of the graph

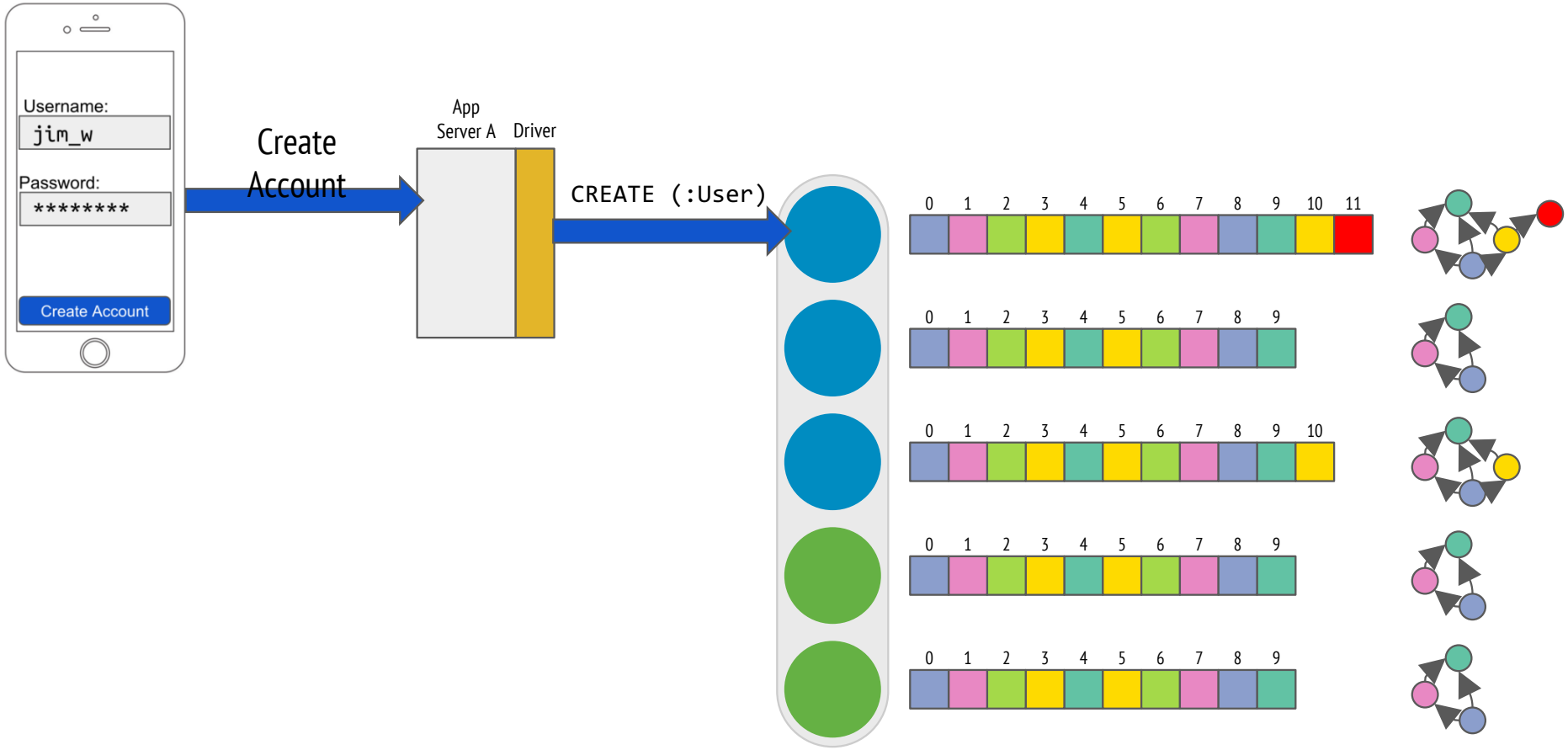
More capabilities to come

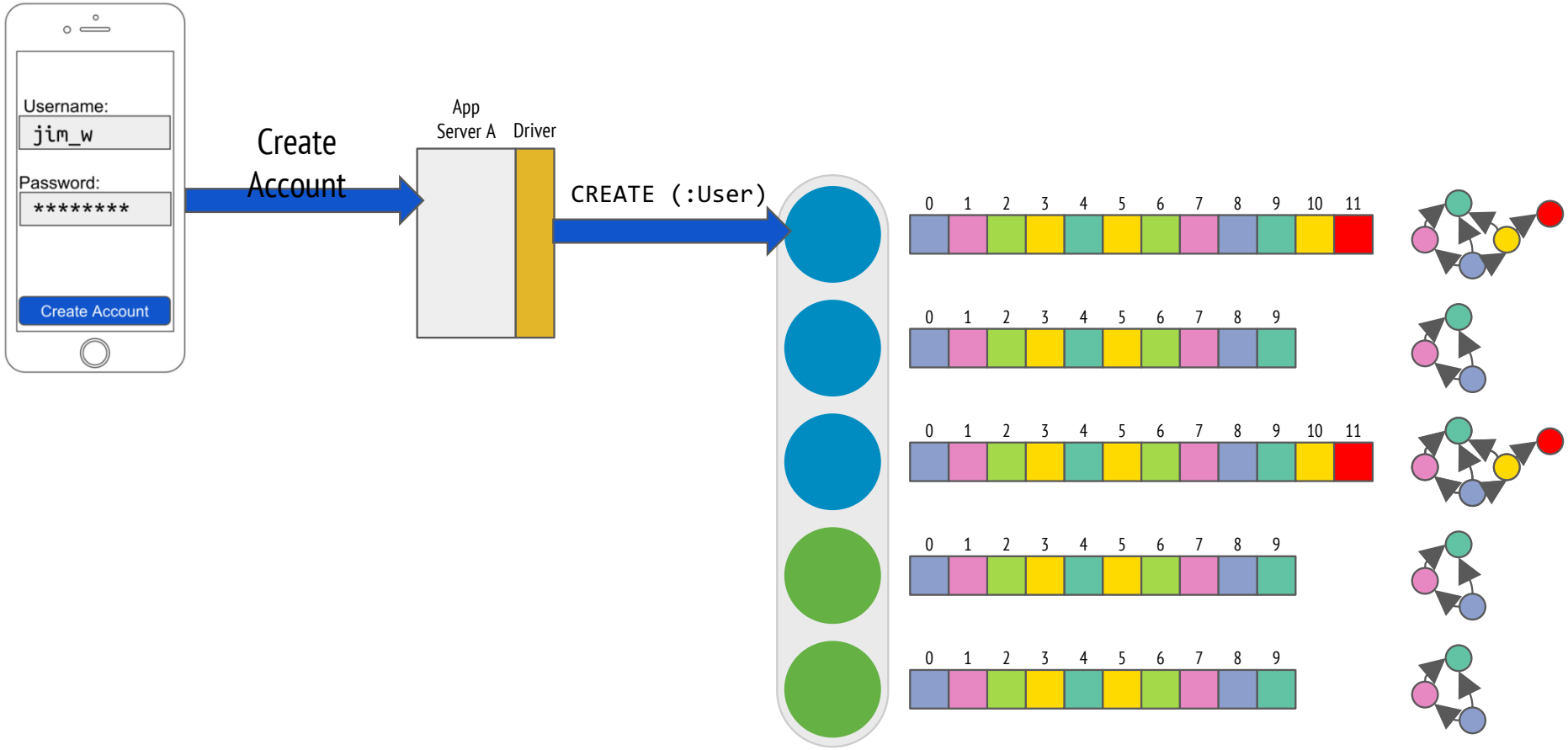
Let’s try again, with Causal Consistency

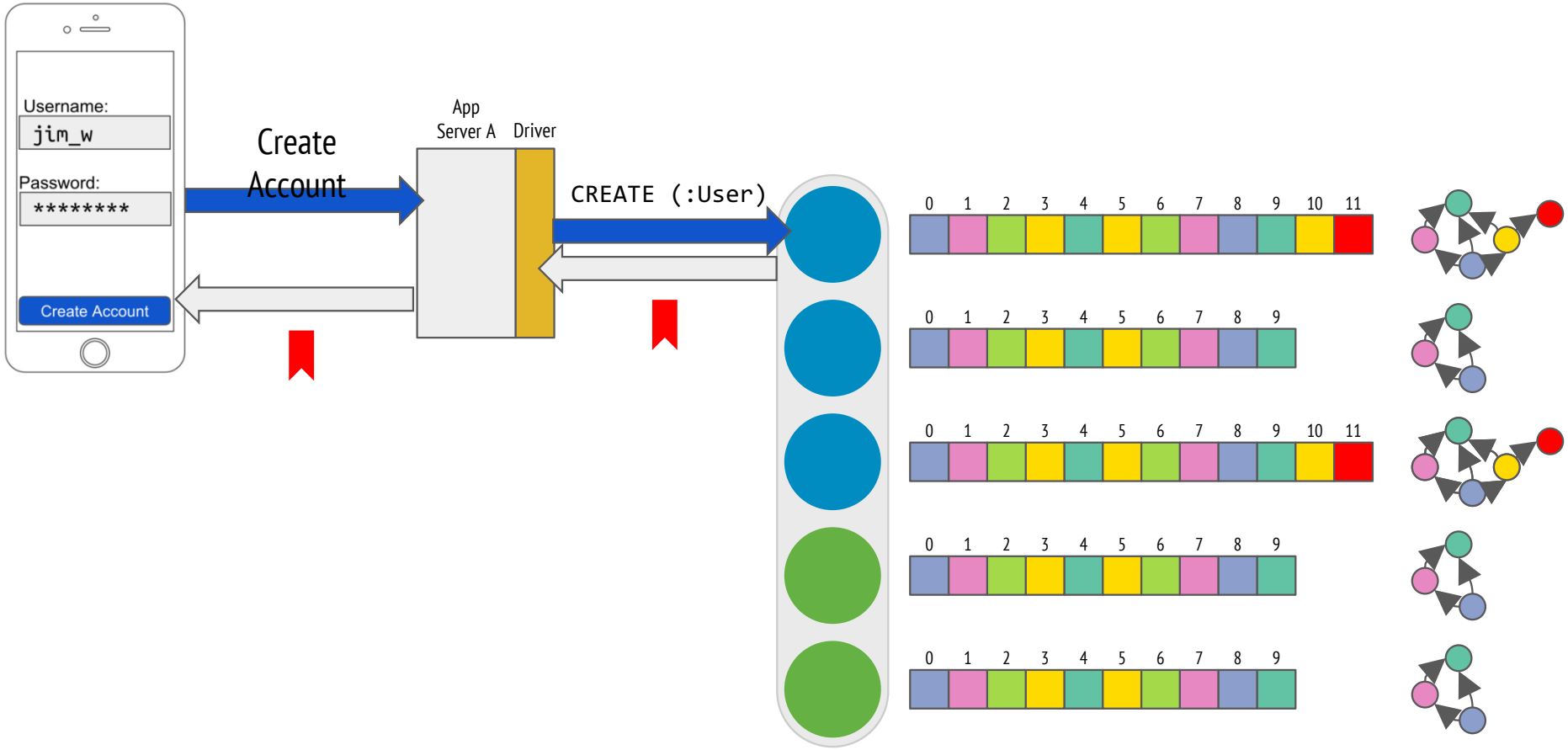

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver

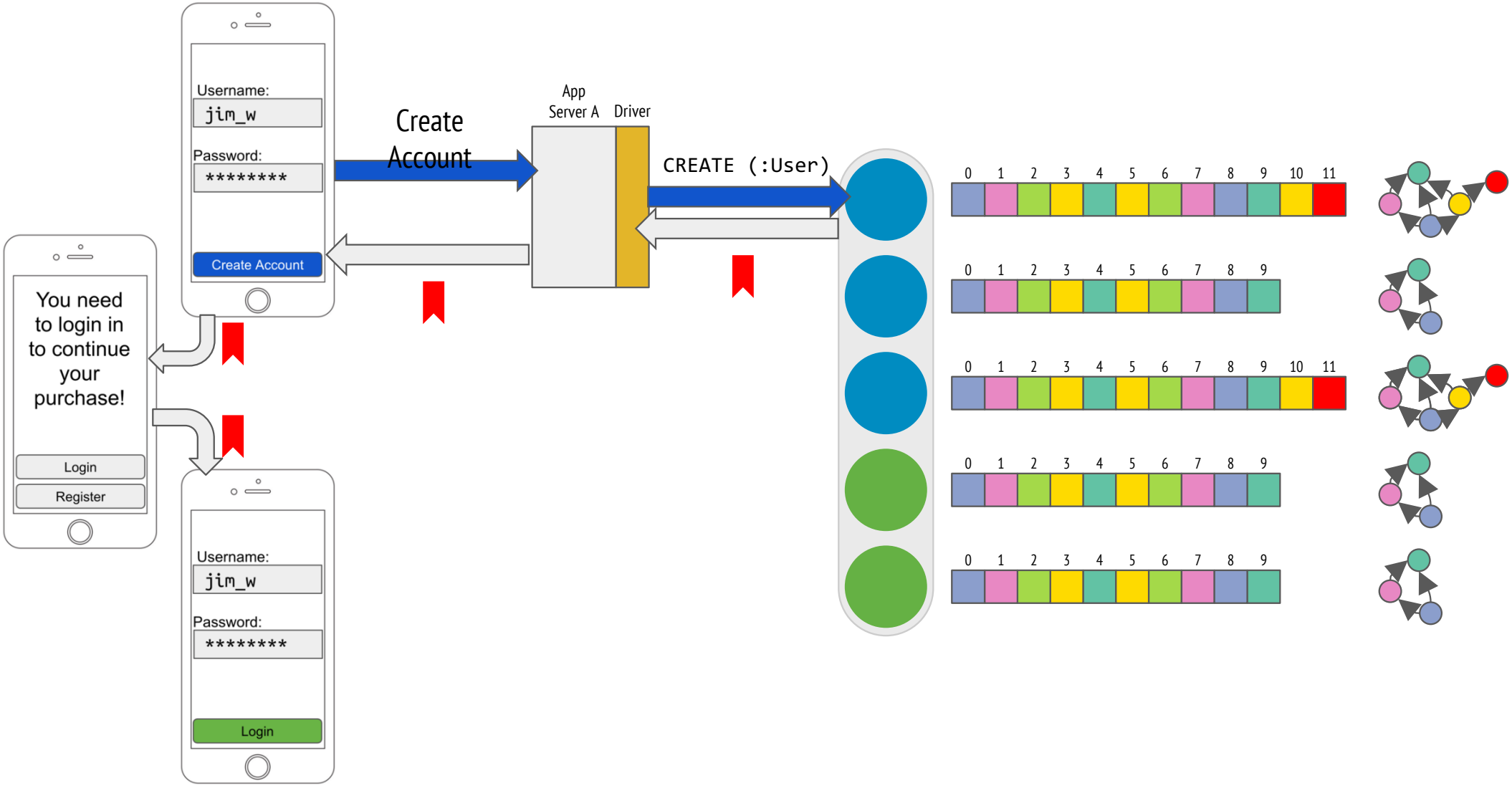

0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

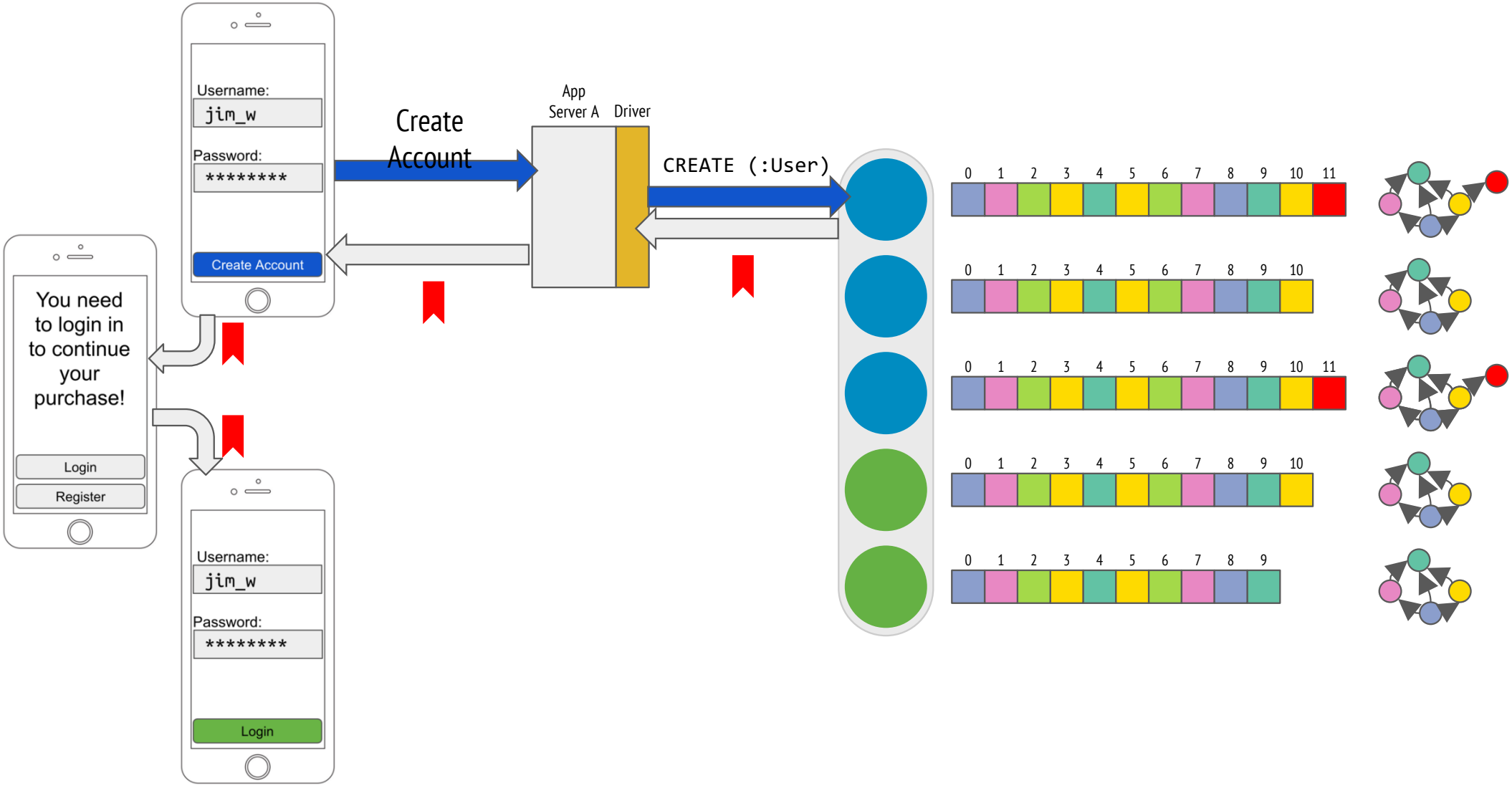

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

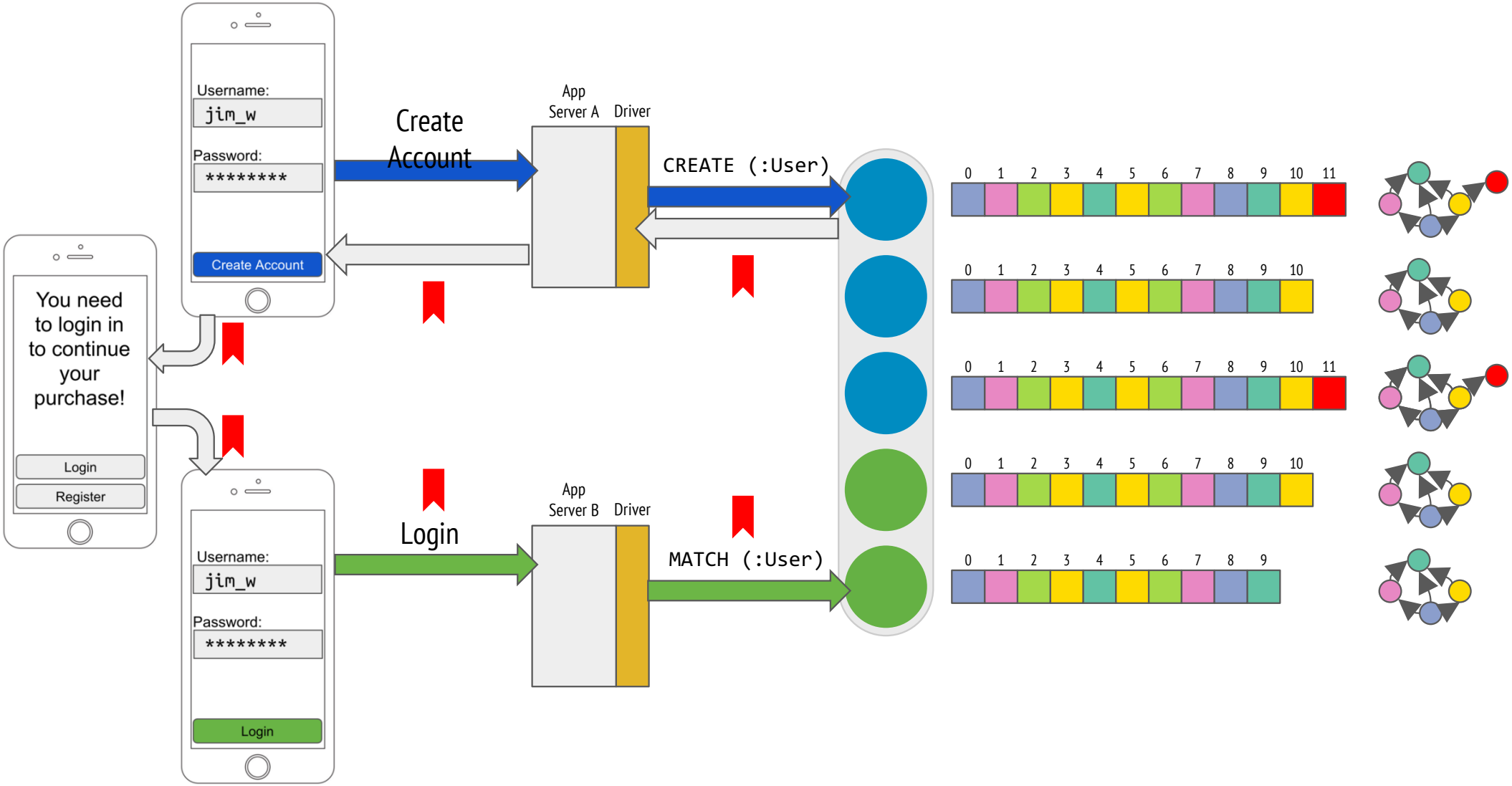

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

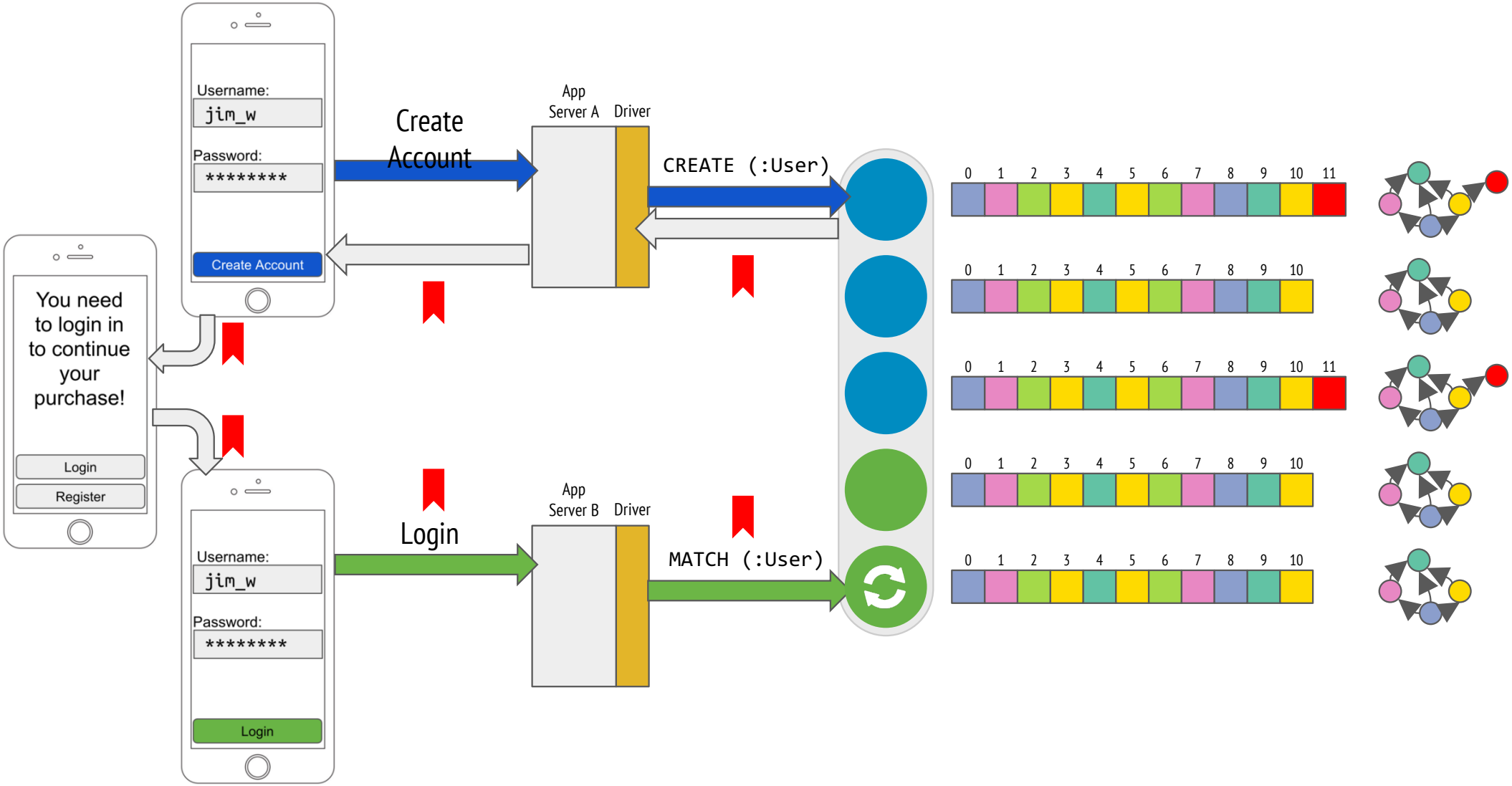

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

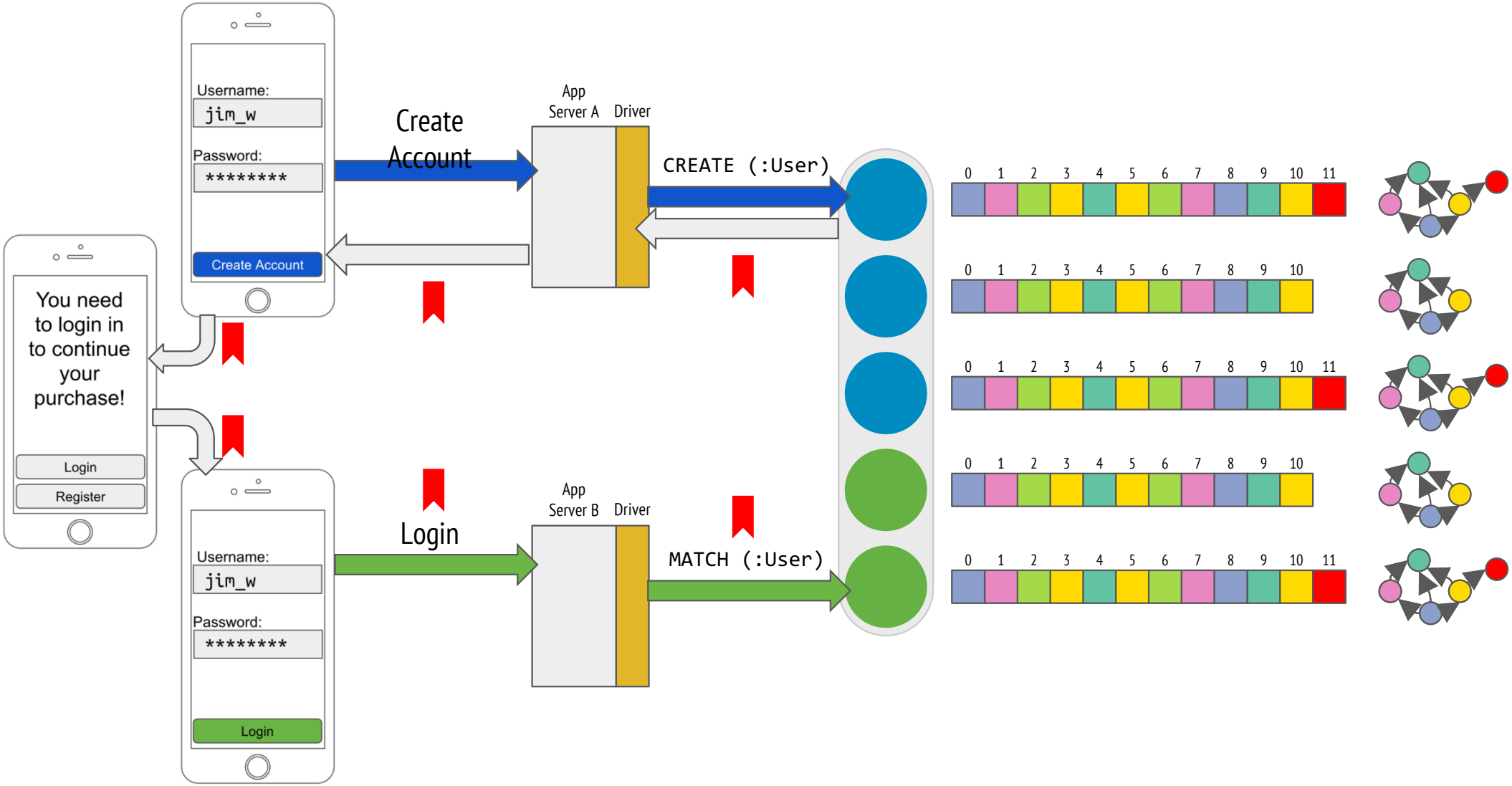

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

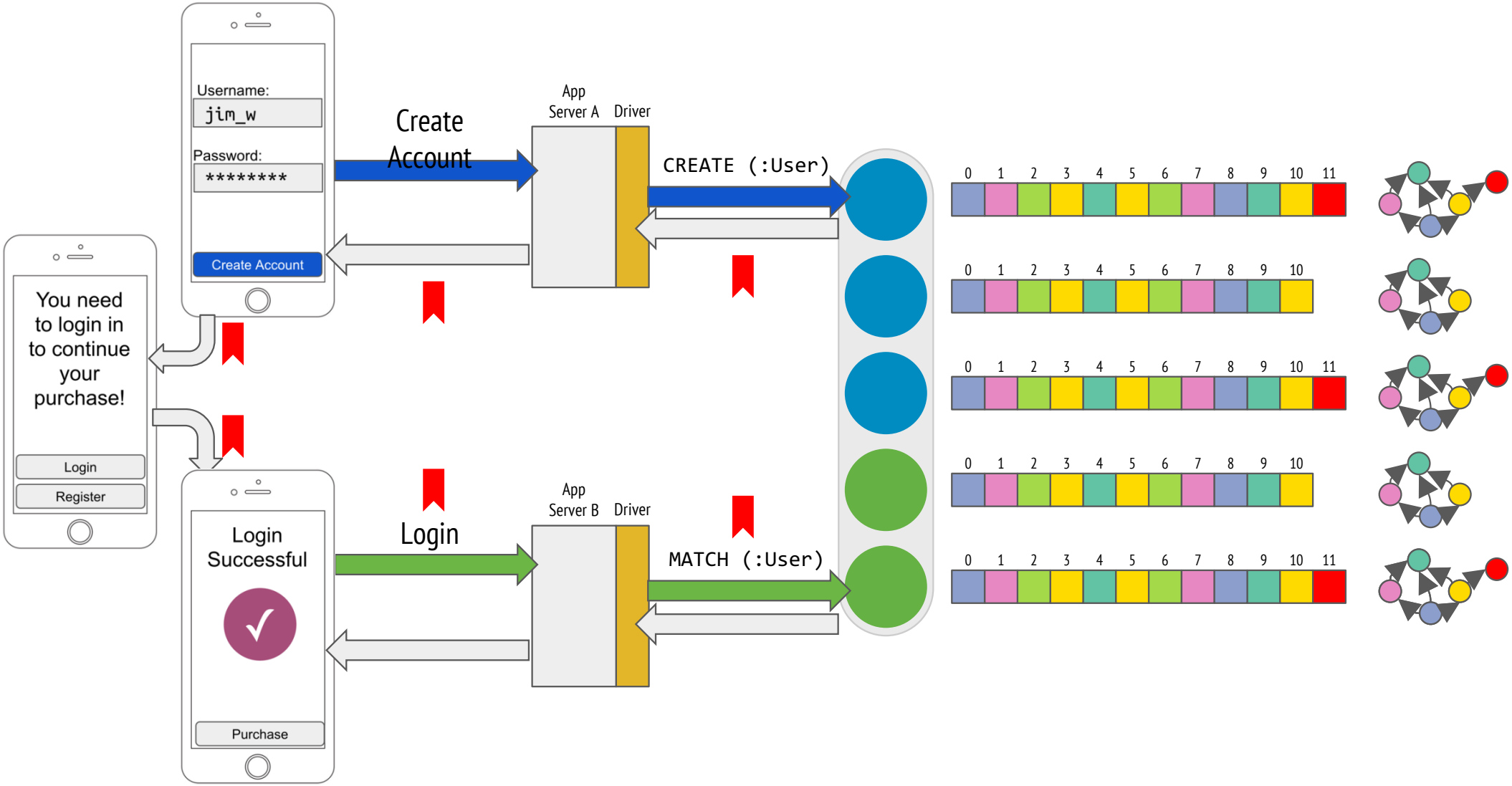

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9

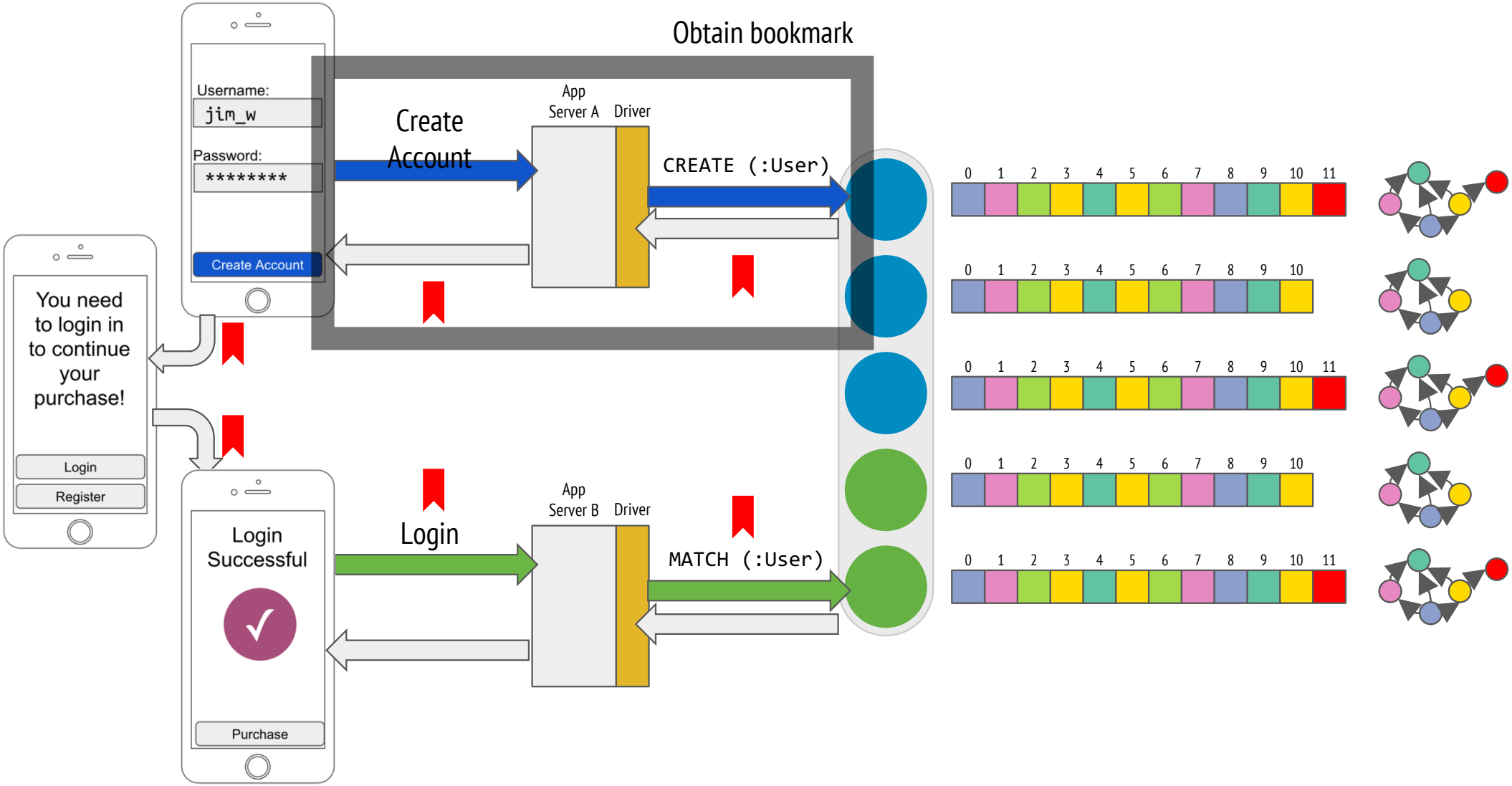

Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Username:
jim_w
Password:
****
Login
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Login Successful
Purchase
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11

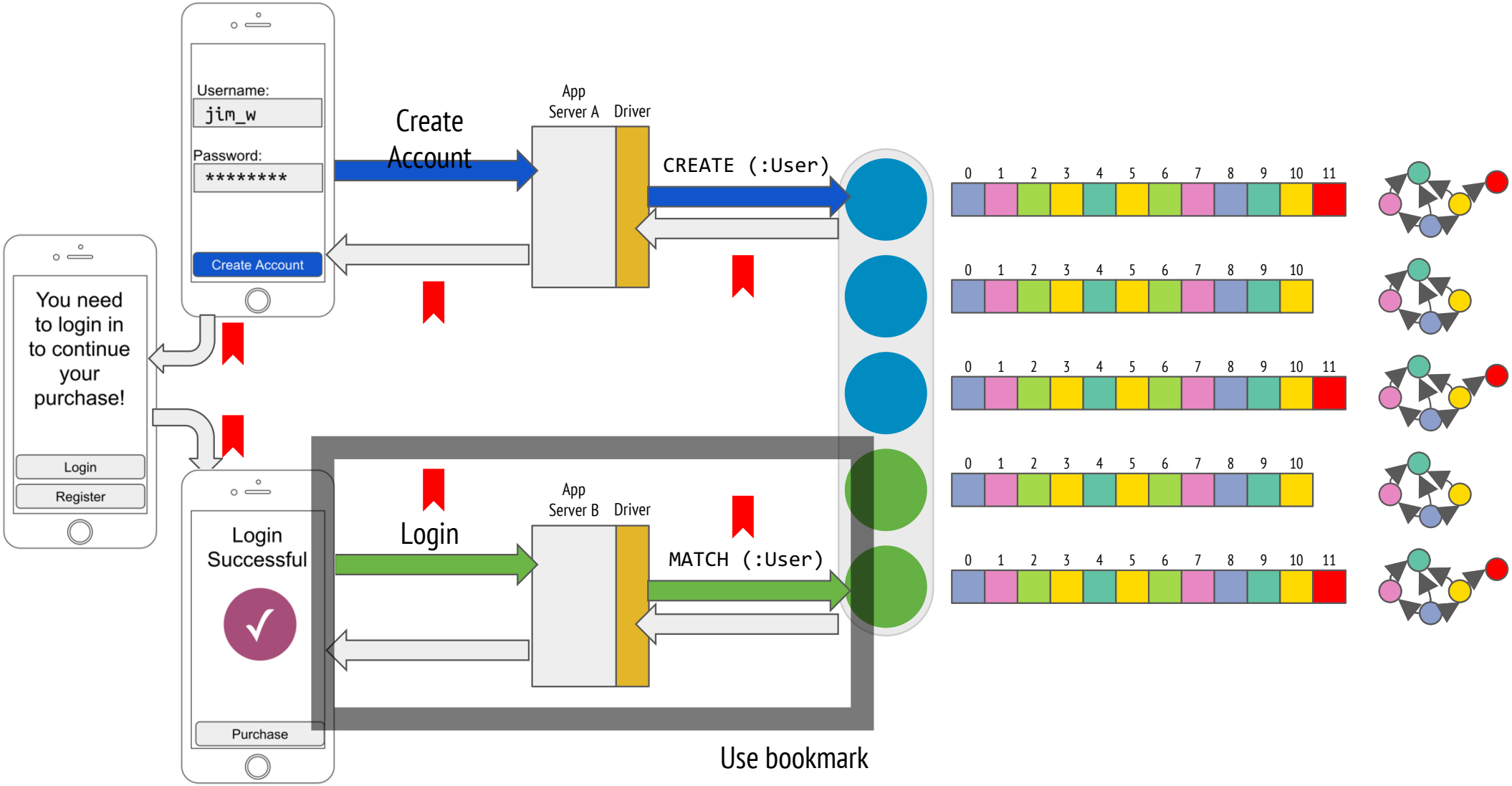
Obtain bookmark

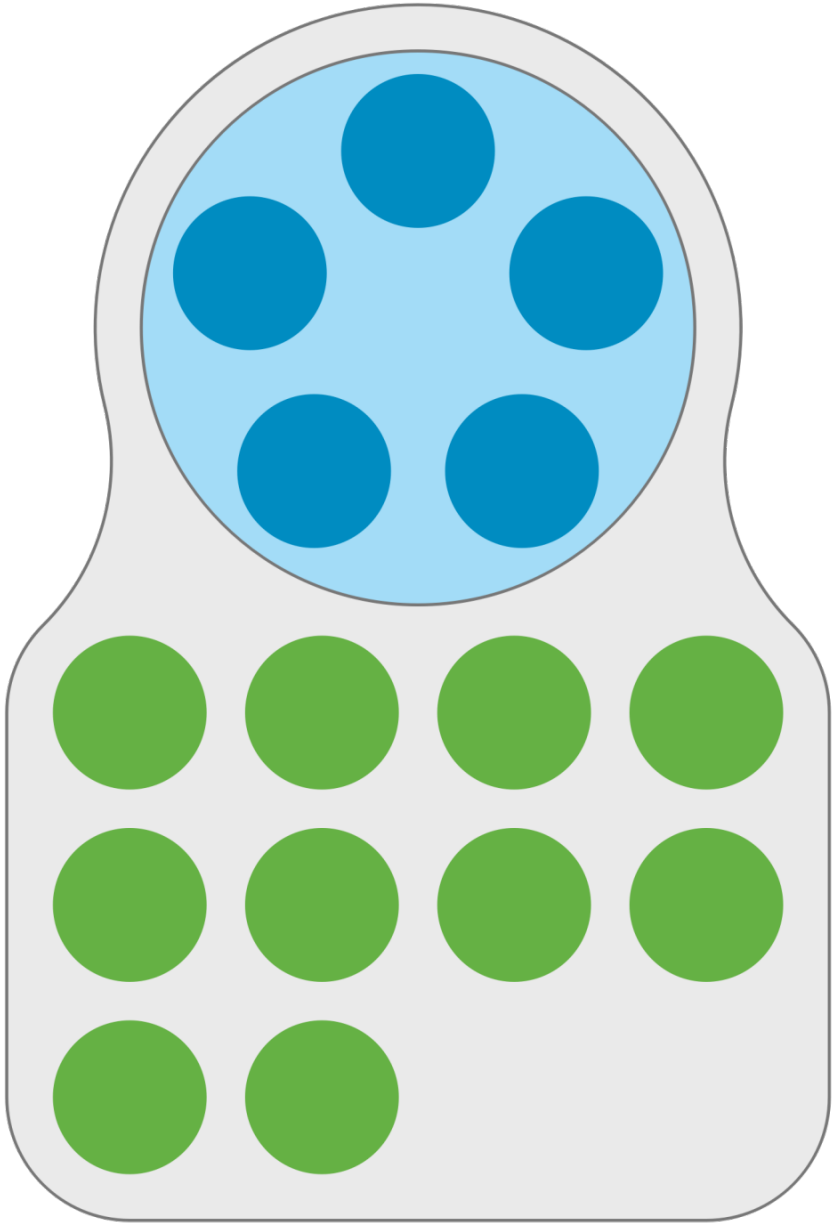
```
try ( Session session = driver.session( AccessMode.WRITE ) )
{
    try ( Transaction tx = session.beginTransaction() )
    {
        tx.run( "CREATE (user:User {userId: {userId}, passwordHash:
{passwordHash})",
                parameters( "userId", userId, "passwordHash", passwordHash ) );

        tx.success();
    }

    String bookmark = session.lastBookmark();
}
```


Obtain bookmark
Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Login Successful
Purchase
Login
App Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11

Use a bookmark

```
try ( Session session = driver.session( AccessMode.READ ) )
{
    try ( Transaction tx = session.beginTransaction( bookmark ) )
    {
        tx.run( "MATCH (user:User {userId: {userId}}) RETURN *",
                parameters( "userId", userId ) );

        tx.success();
    }
}
```


Username:
jim_w
Password:
****
Create Account
Create Account
App Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Login Successful
Purchase
Login
App Server B
Driver
MATCH (:User)
Use bookmark
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11



Core Updating the graph

Read Replicas Queries, analysis, reporting

Thank you for listening

@jimwebber