Diamond: Automating Data Management and Storage for Wide-area, Reactive Applications

Irene Zhang Niel Lebeck Pedro Fonseca
Brandon Holt Raymond Cheng
Ariadna Norberg Arvind Krishnamurthy Henry M. Levy

UNIVERSITY *of* WASHINGTON

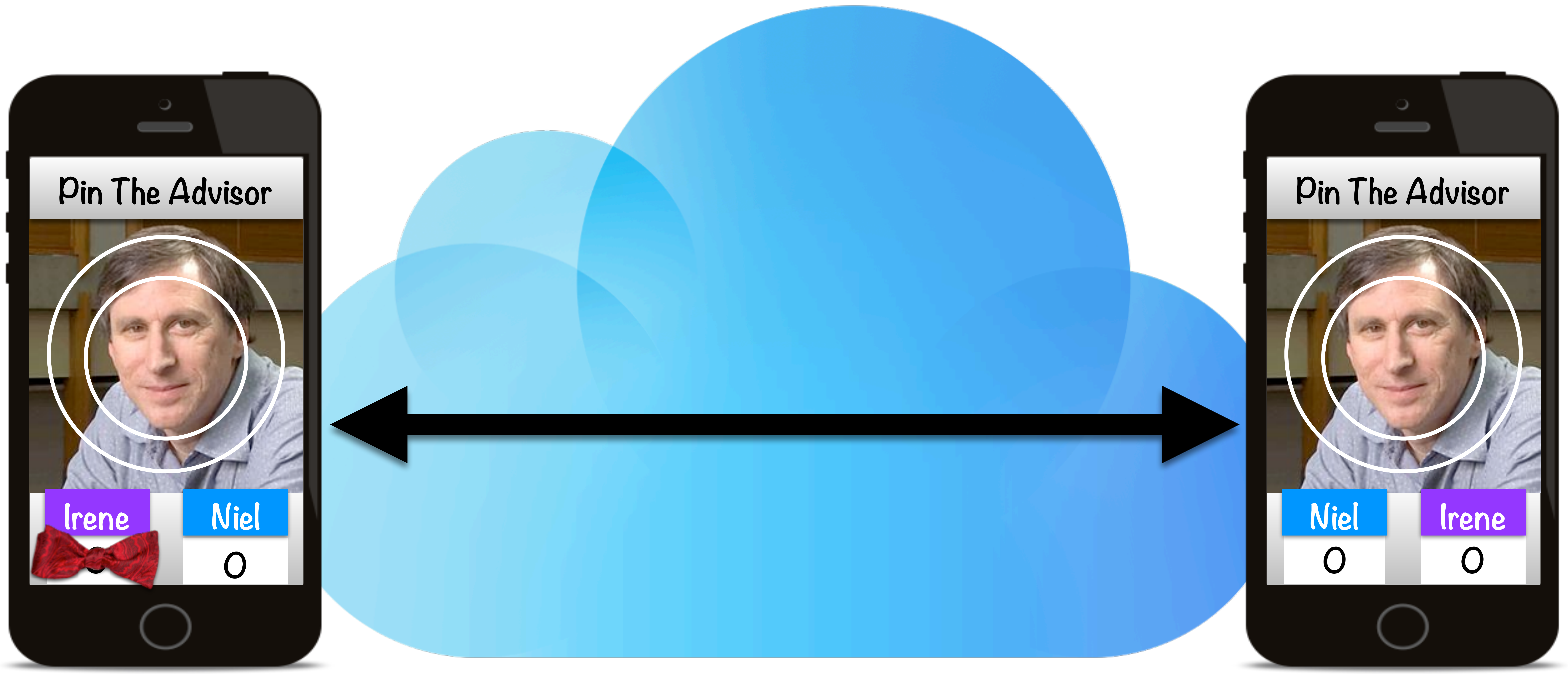
Diamond: Automating Data Management and Storage for Wide-area, Reactive Applications

Irene Zhang Niel Lebeck Pedro Fonseca
Brandon Holt Raymond Cheng
Ariadna Norberg Arvind Krishnamurthy Henry M. Levy

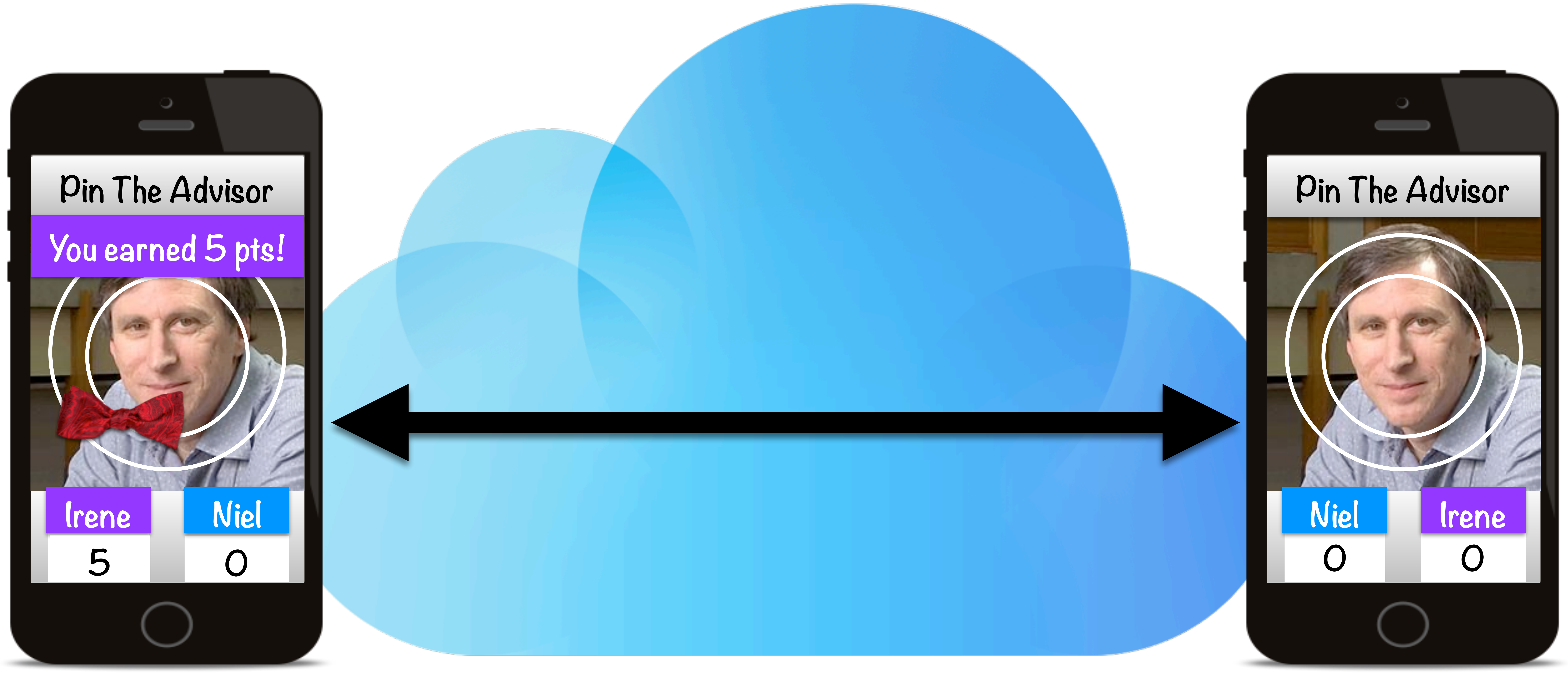
UNIVERSITY *of* WASHINGTON

Reactive applications <u>automatically</u> propagate updates across mobile devices and the cloud.

Reactive applications <u>automatically</u> propagate updates across mobile devices and the cloud.



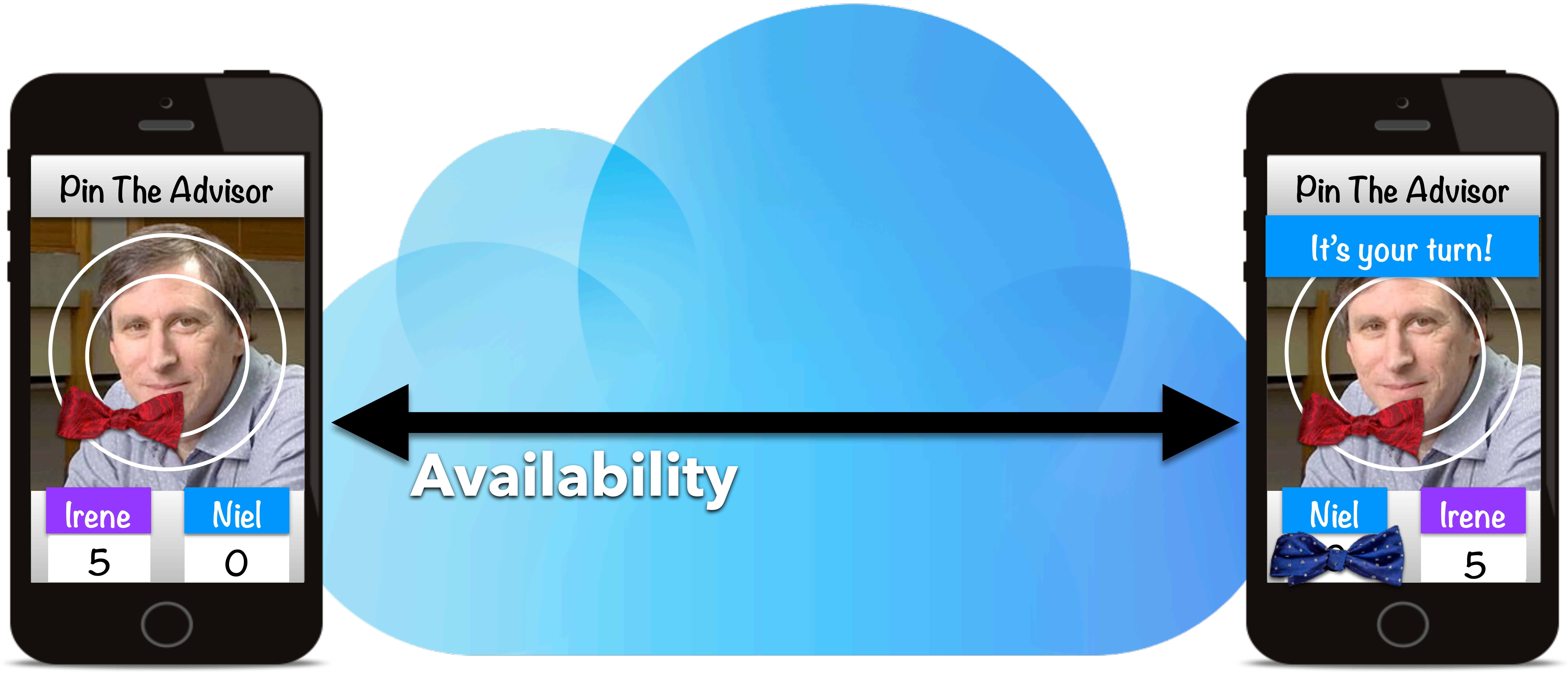
Reactive applications <u>automatically</u> propagate updates across mobile devices and the cloud.



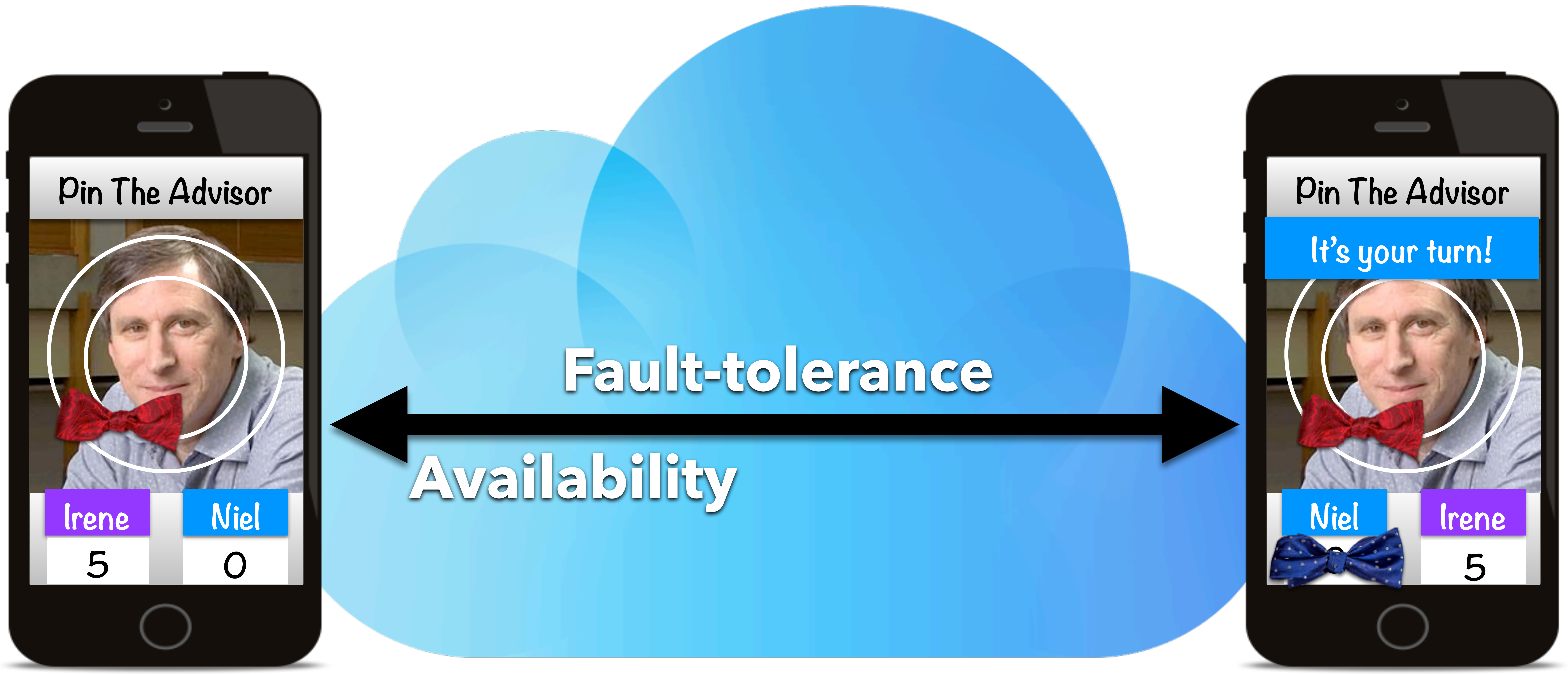
Reactive applications <u>automatically</u> propagate updates across mobile devices and the cloud.

Pin The Advisor
Irene
5
Niel
0
Pin The Advisor
It's your turn!
Niel
Irene
5

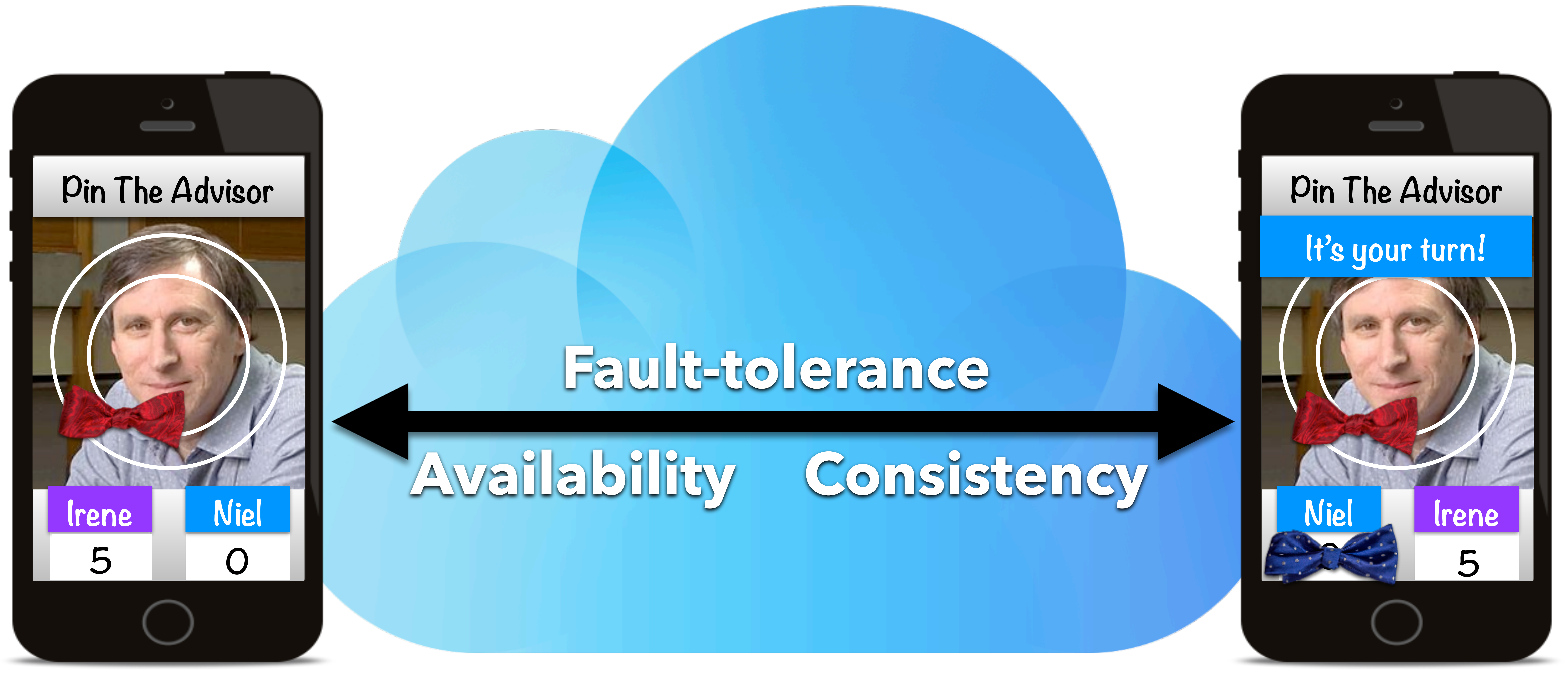
Reactive applications automatically propagate updates across mobile devices and the cloud.



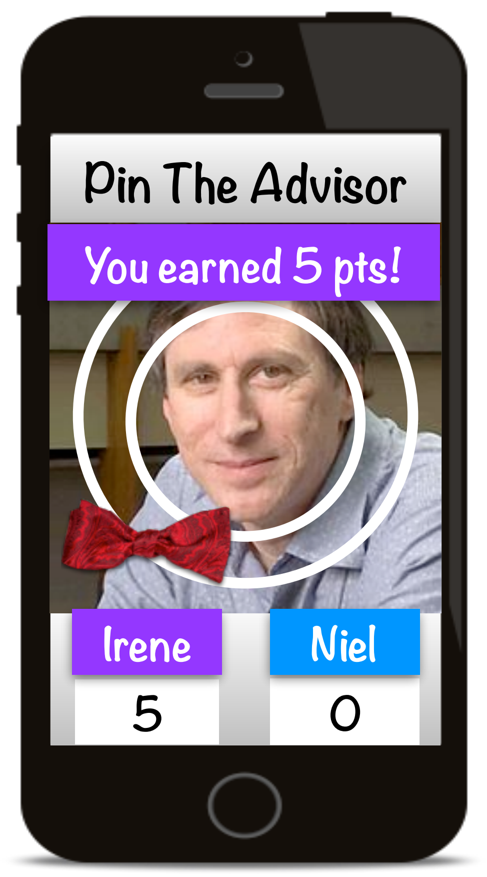
Reactive applications automatically propagate updates across mobile devices and the cloud.

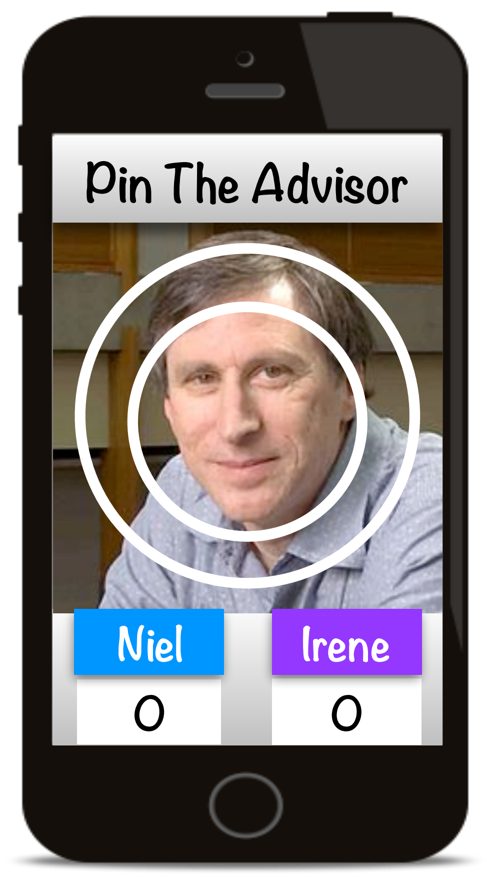


Reactive applications <u>automatically</u> propagate updates across mobile devices and the cloud.

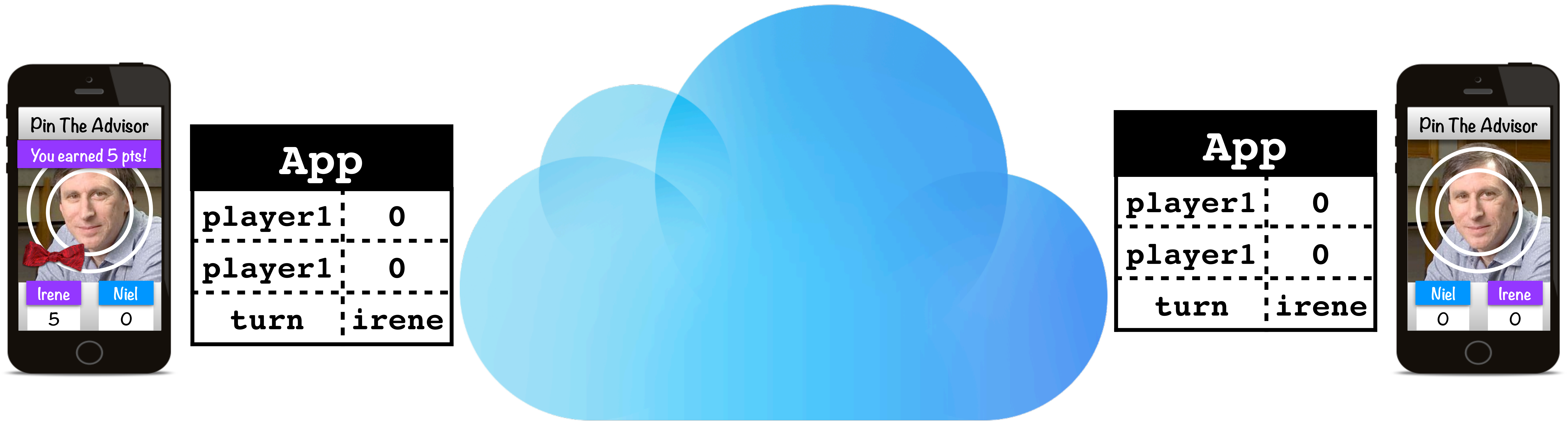


Which poses a challenge for app programmers.





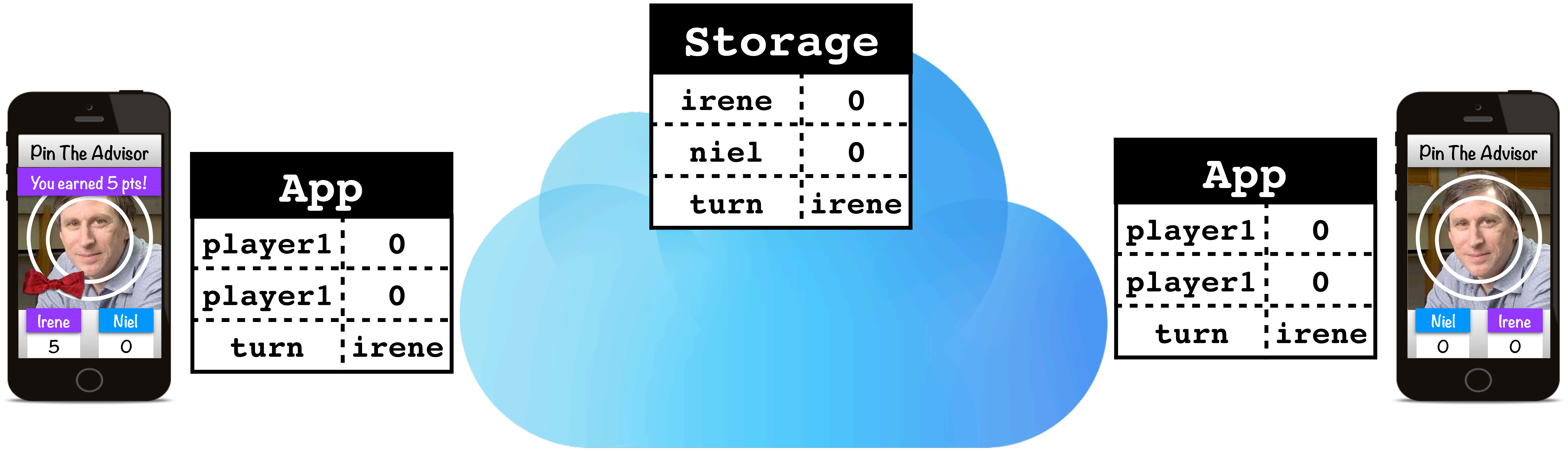
Which poses a challenge for app programmers.



App	
player1	0
player1	0
turn	irene

App	
player1	0
player1	0
turn	irene

Which poses a challenge for app programmers.



App

player1	0
player1	0
turn	irene

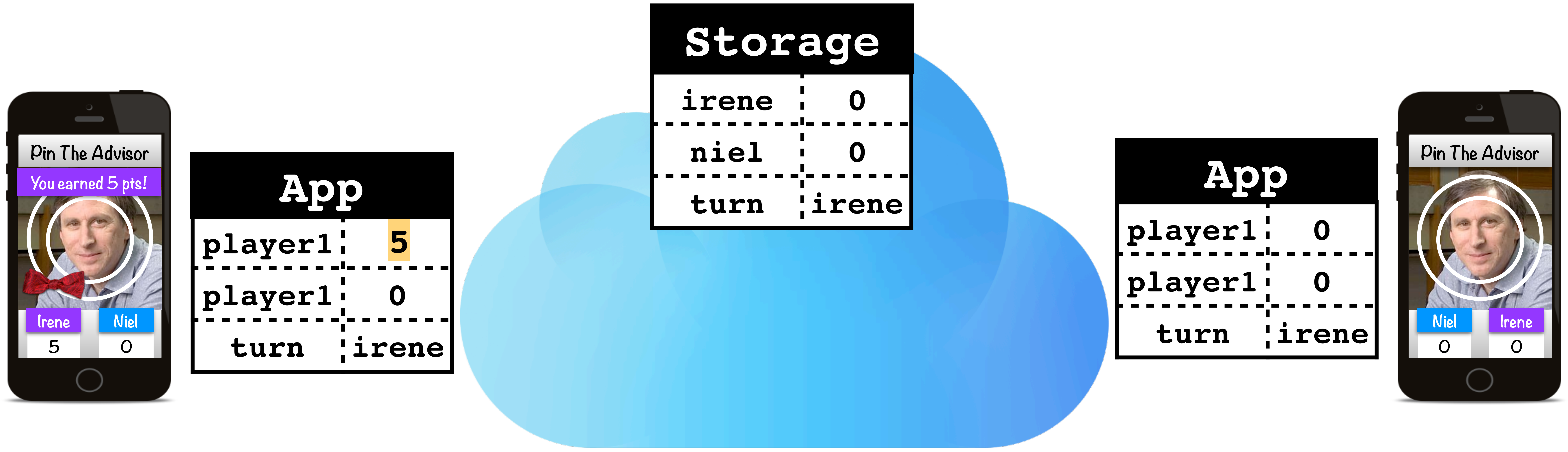
Storage

irene	0
niel	0
turn	irene

App

player1	0
player1	0
turn	irene

Which poses a challenge for app programmers.



App

player1	5
player1	0
turn	irene

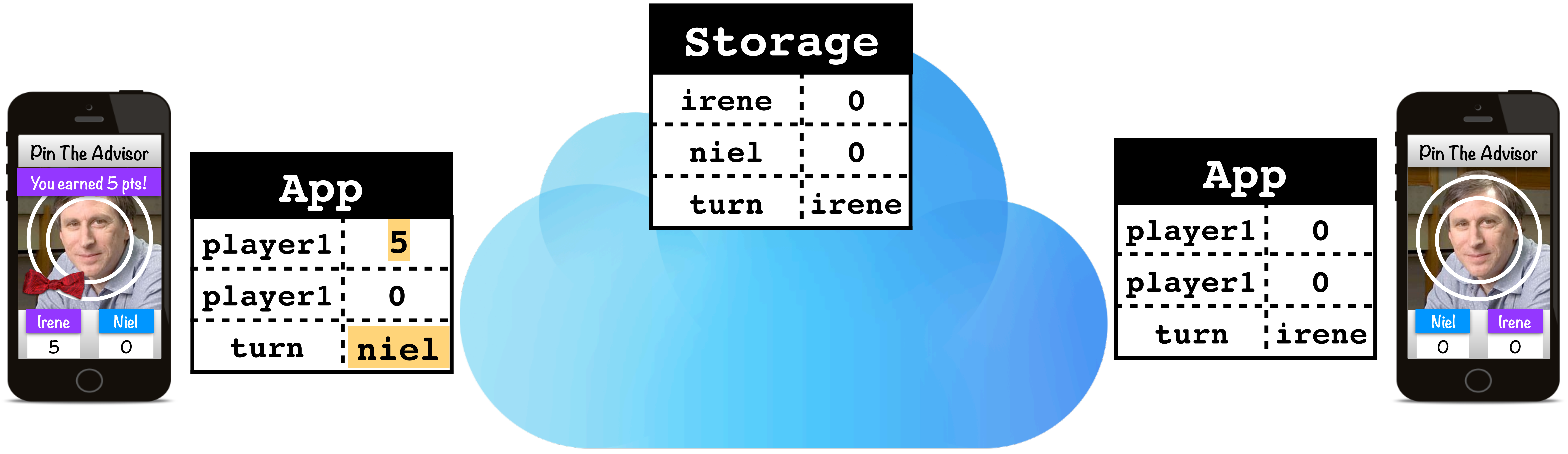
Storage

irene	0
niel	0
turn	irene

App

player1	0
player1	0
turn	irene

Which poses a challenge for app programmers.



App

player1	5
player1	0
turn	niel

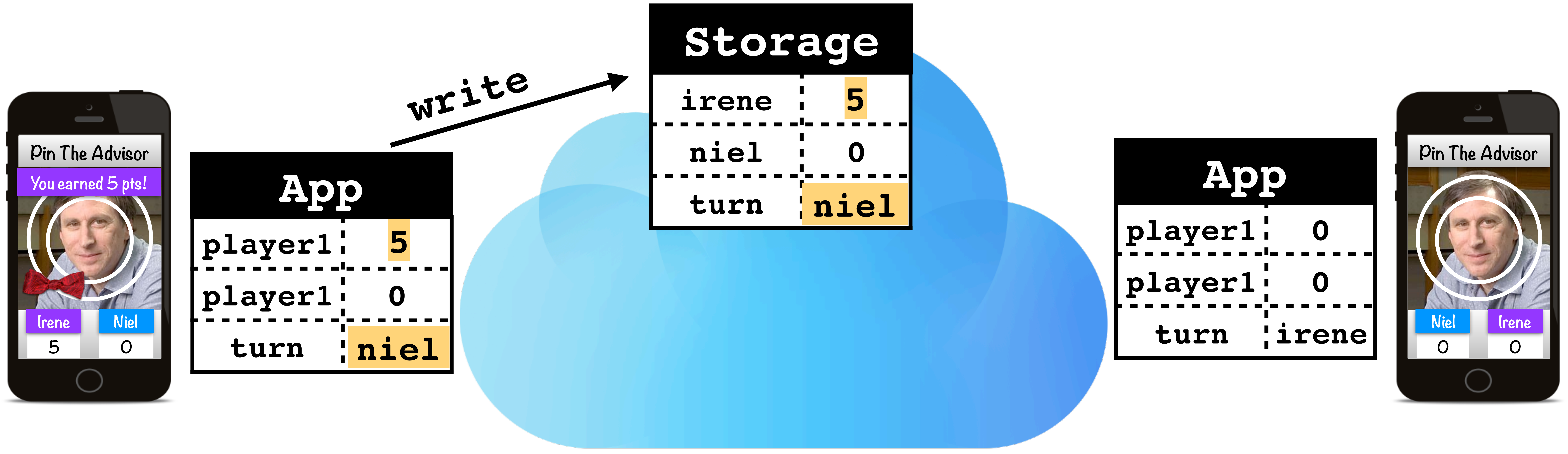
Storage

irene	0
niel	0
turn	irene

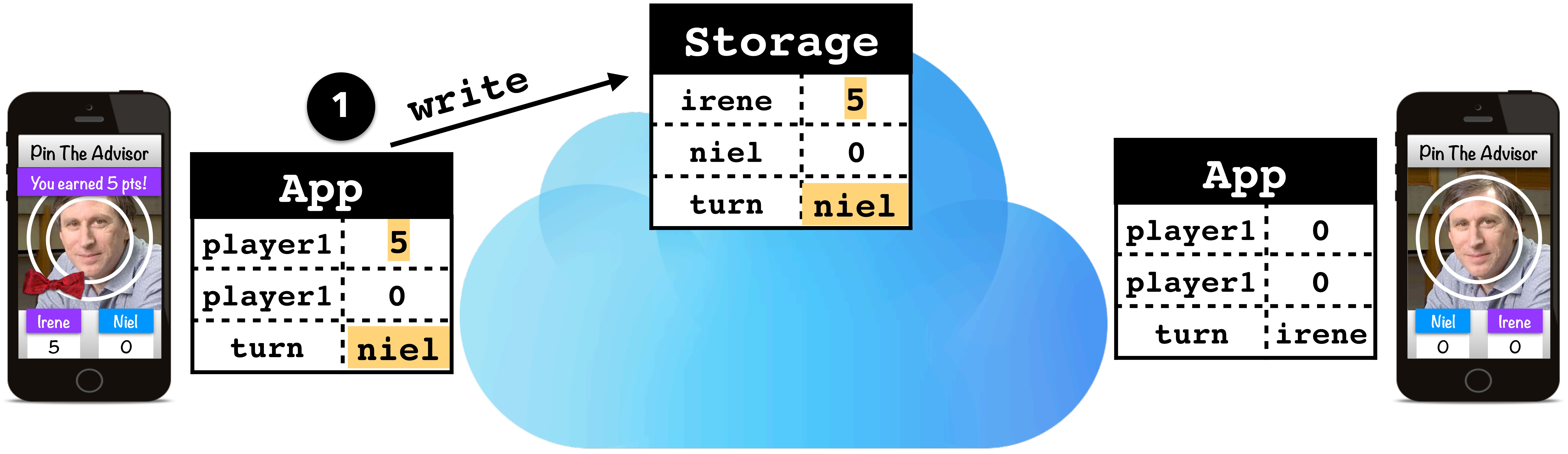
App

player1	0
player1	0
turn	irene

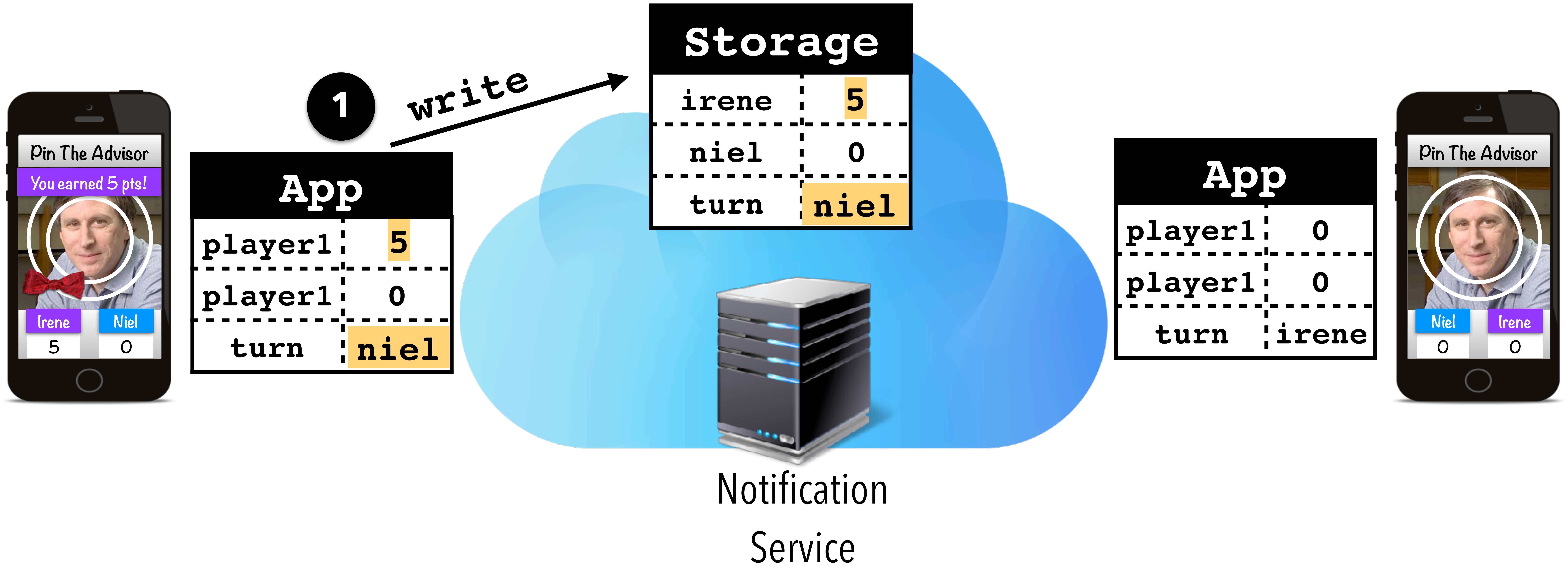
Which poses a challenge for app programmers.



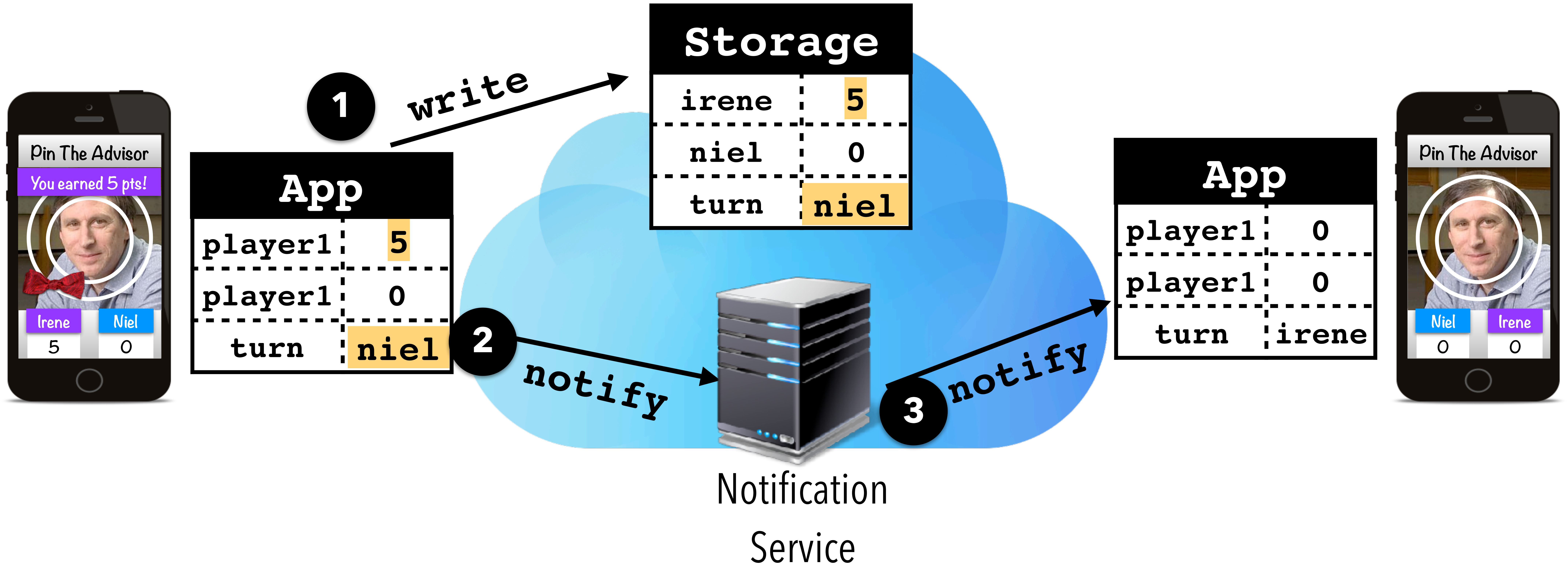
Which poses a challenge for app programmers.



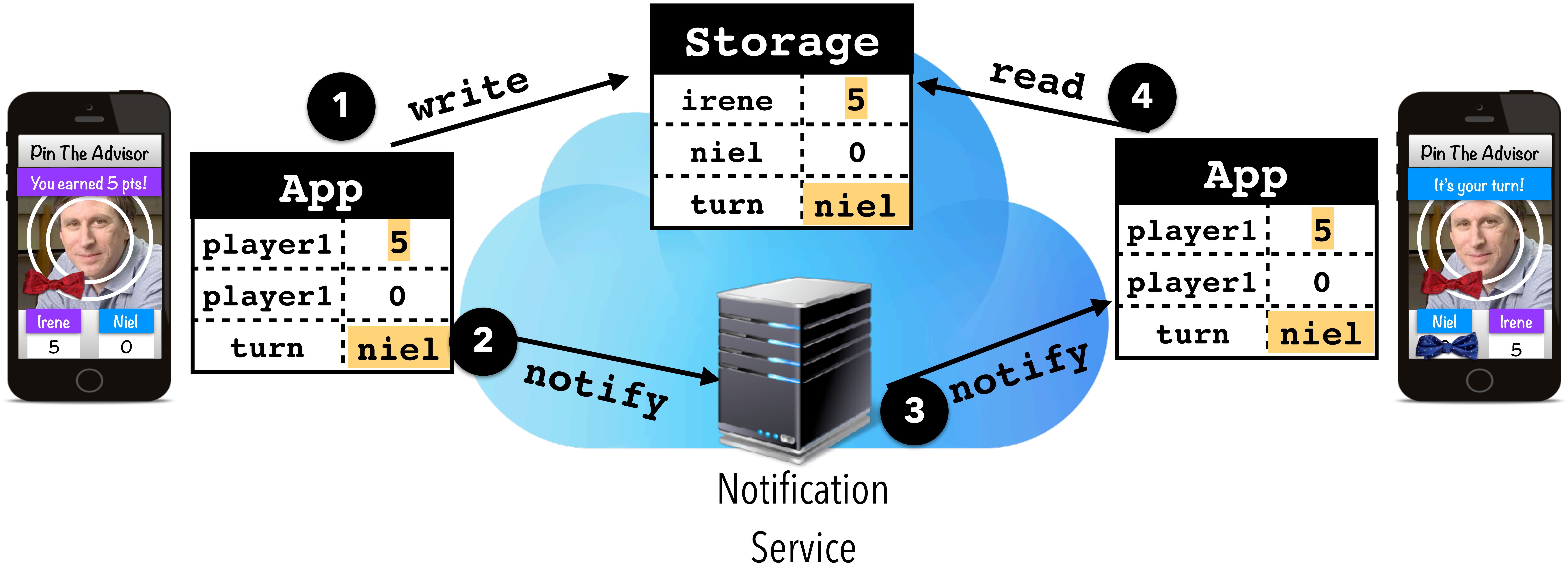
Which poses a challenge for app programmers.



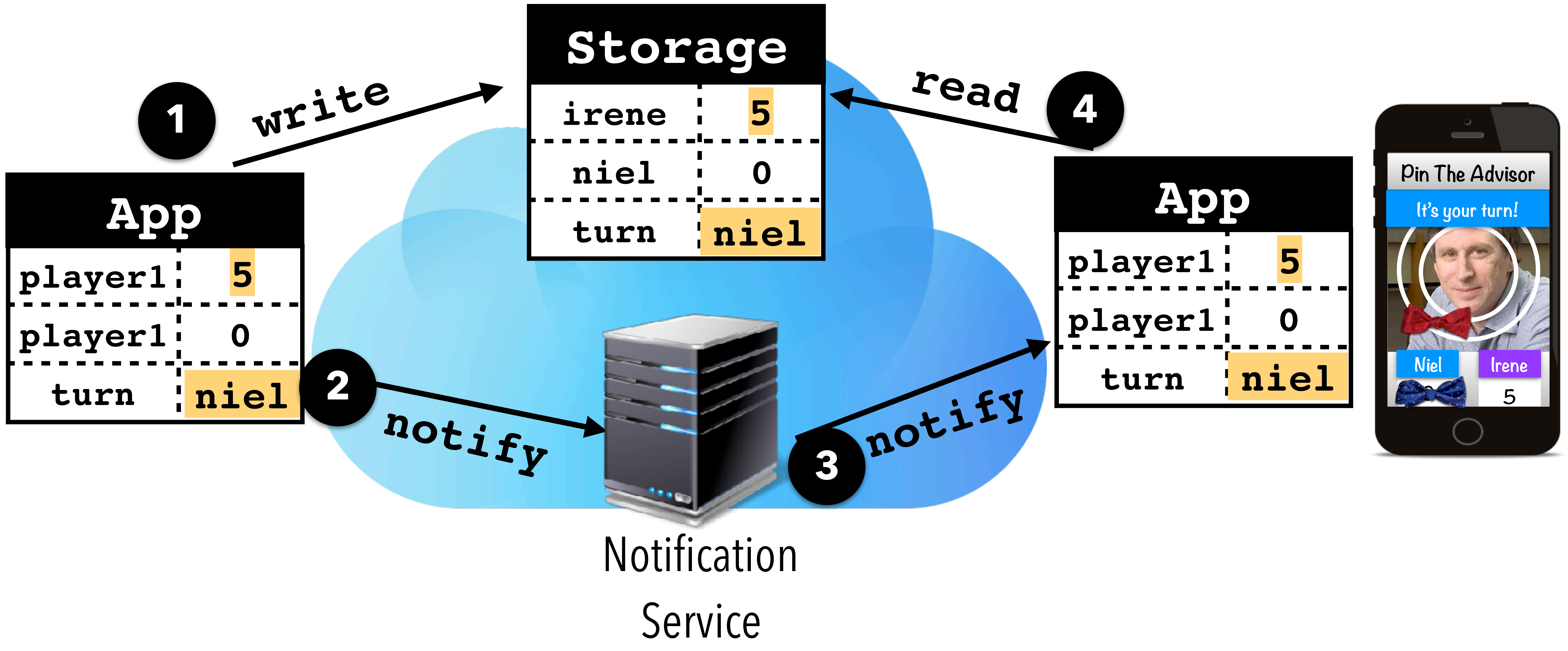
Which poses a challenge for app programmers.



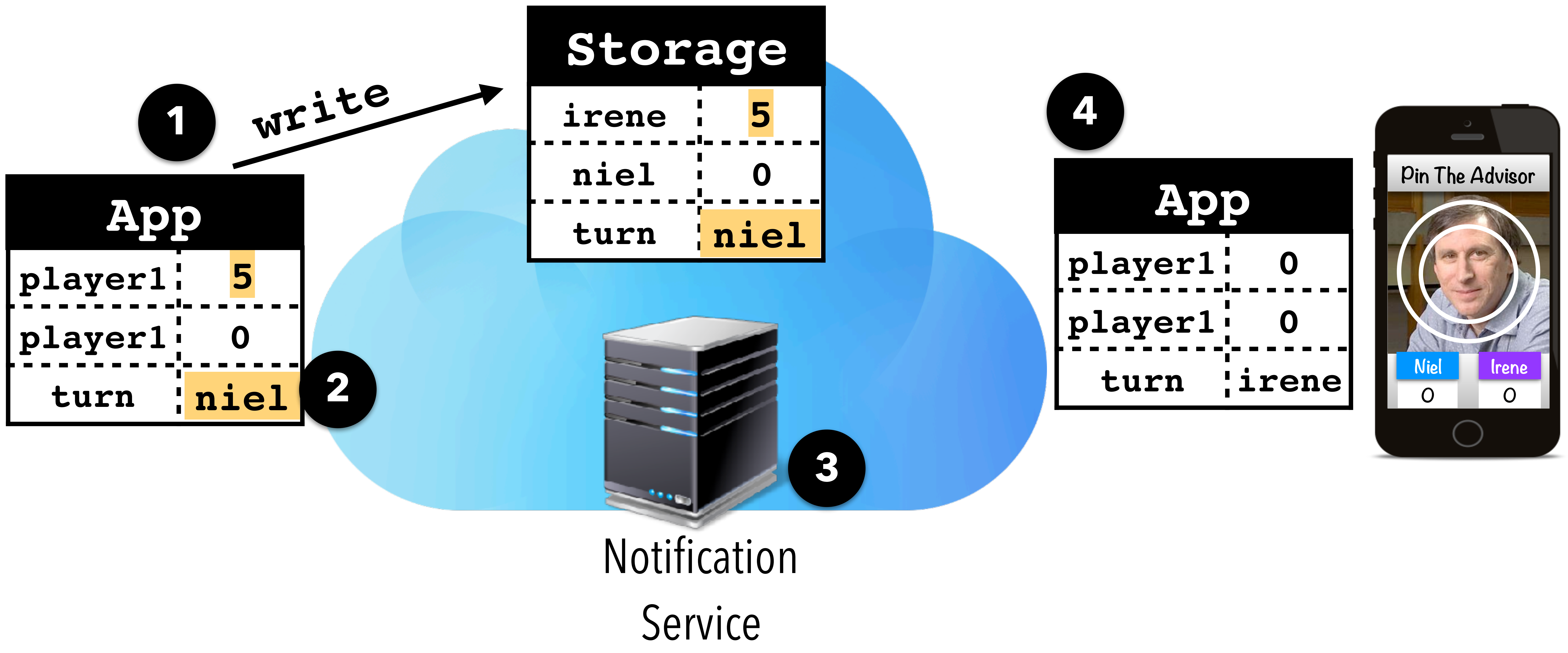
Which poses a challenge for app programmers.



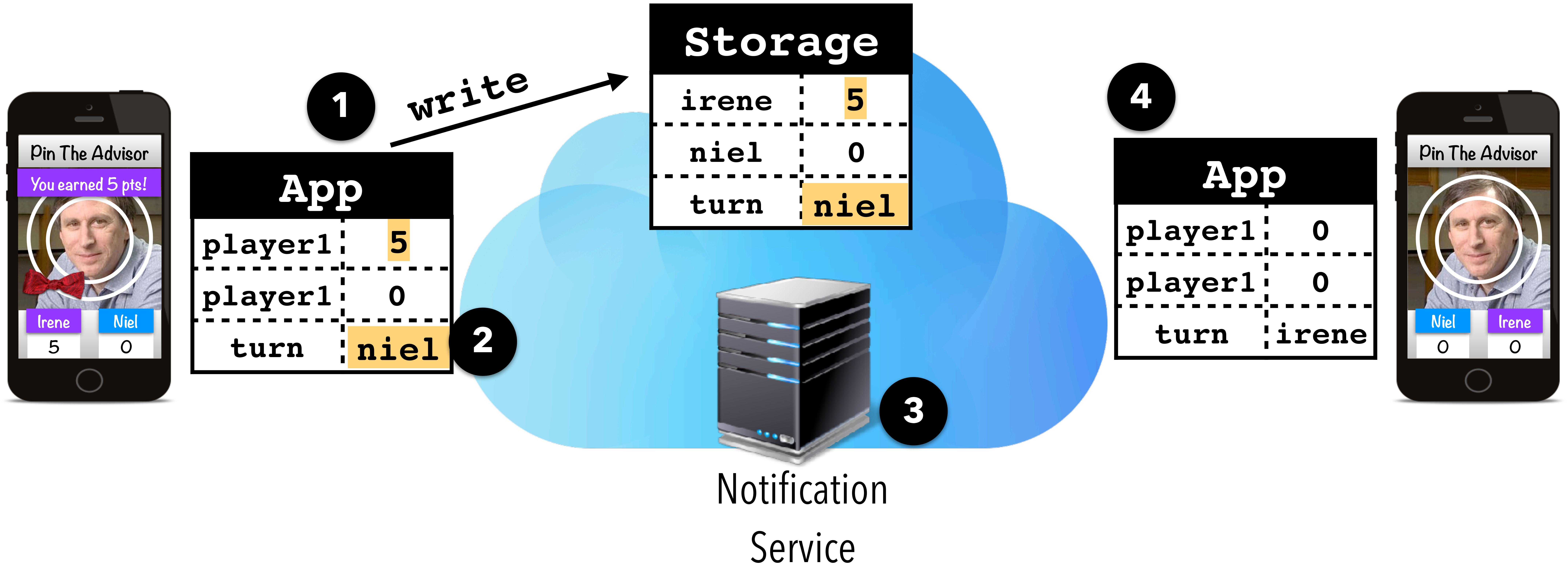
Which poses a challenge for app programmers.



Which poses a challenge for app programmers.

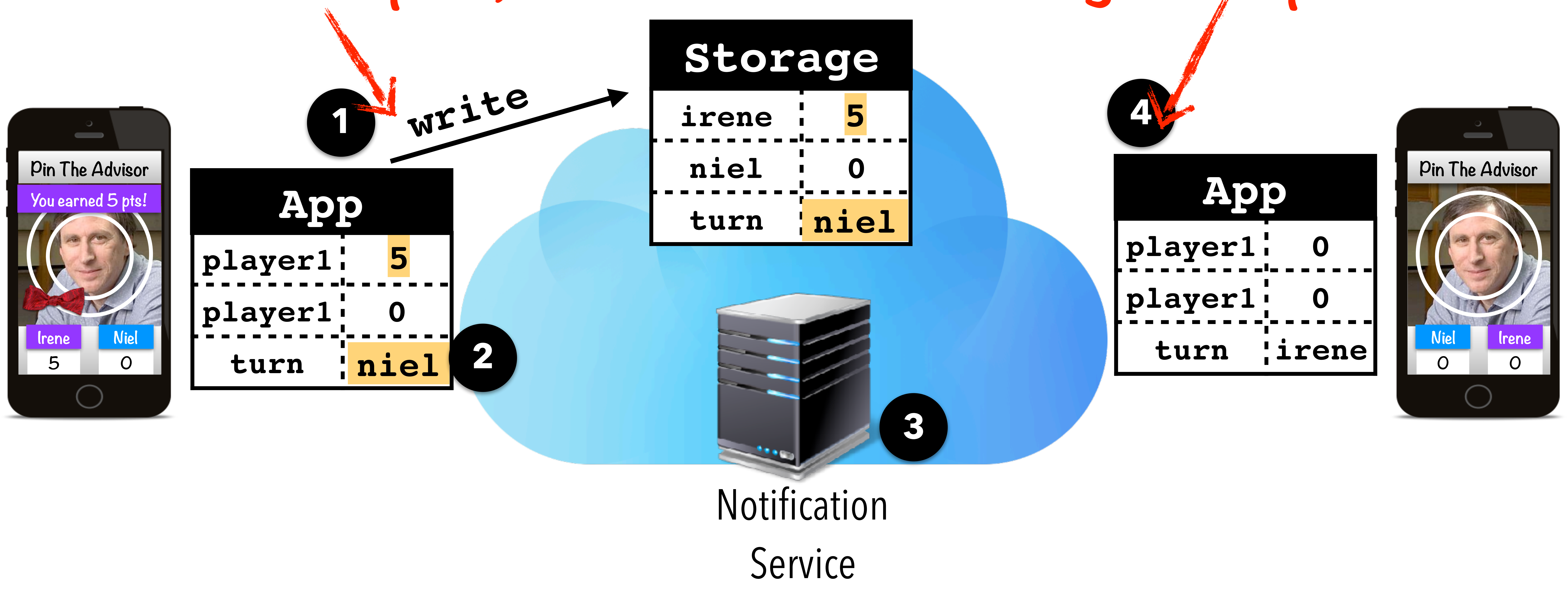


Which poses a challenge for app programmers.



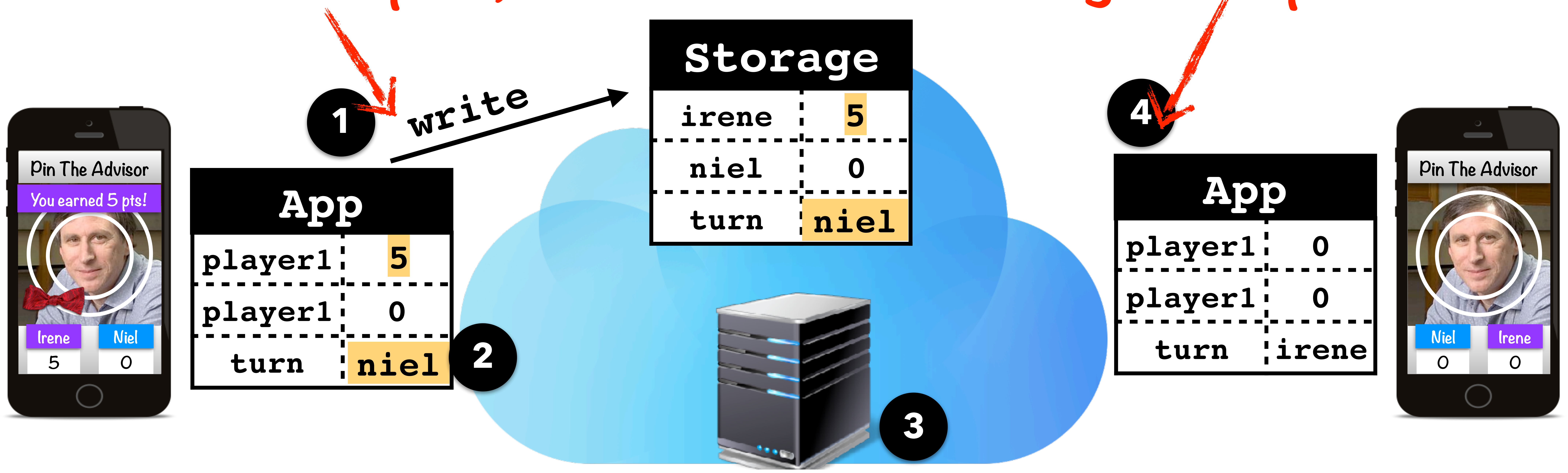
Which poses a challenge for app programmers.

This is a complex, distributed data management problem!



Which poses a challenge for app programmers.

This is a complex, distributed data management problem!

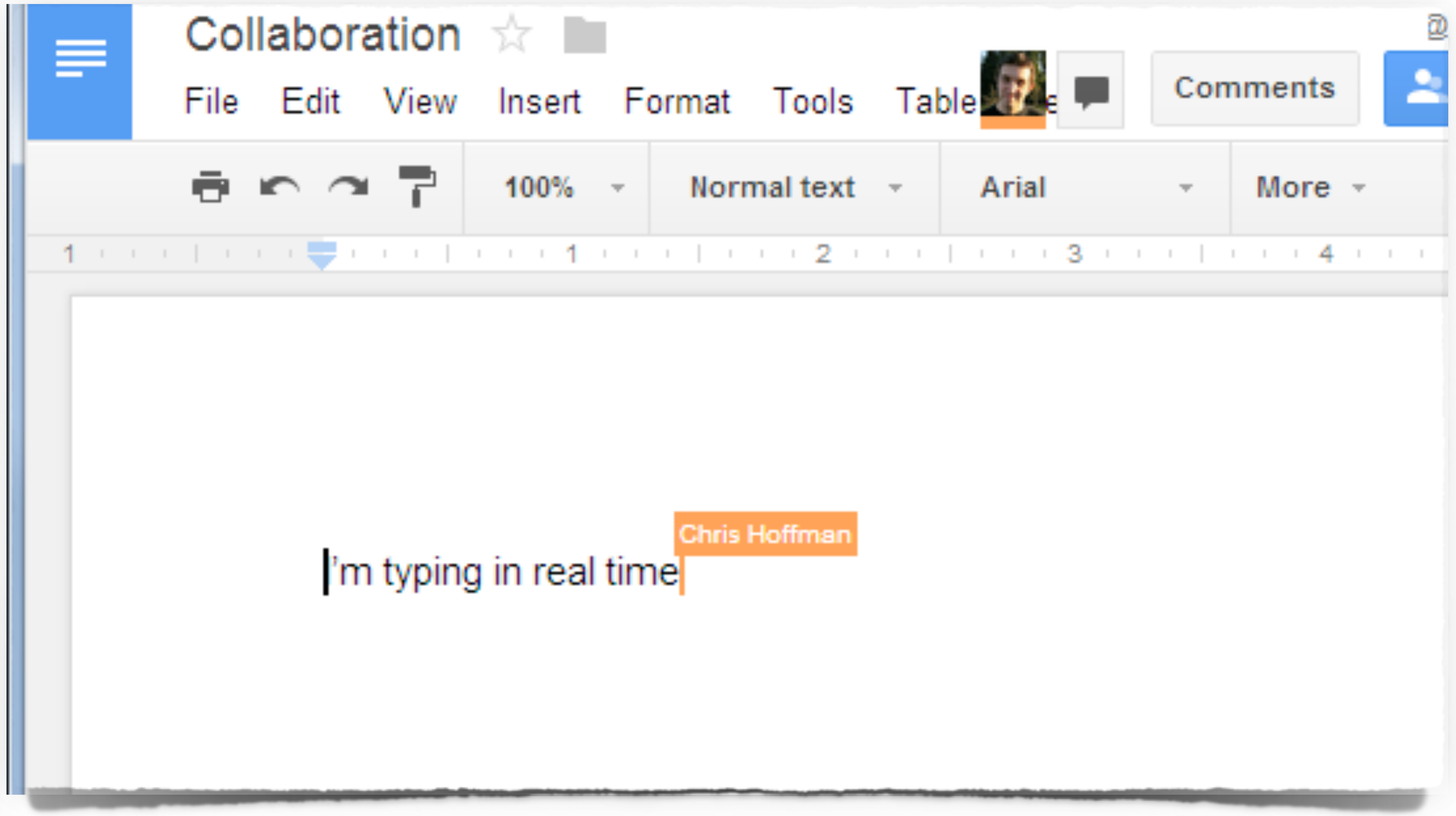


Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

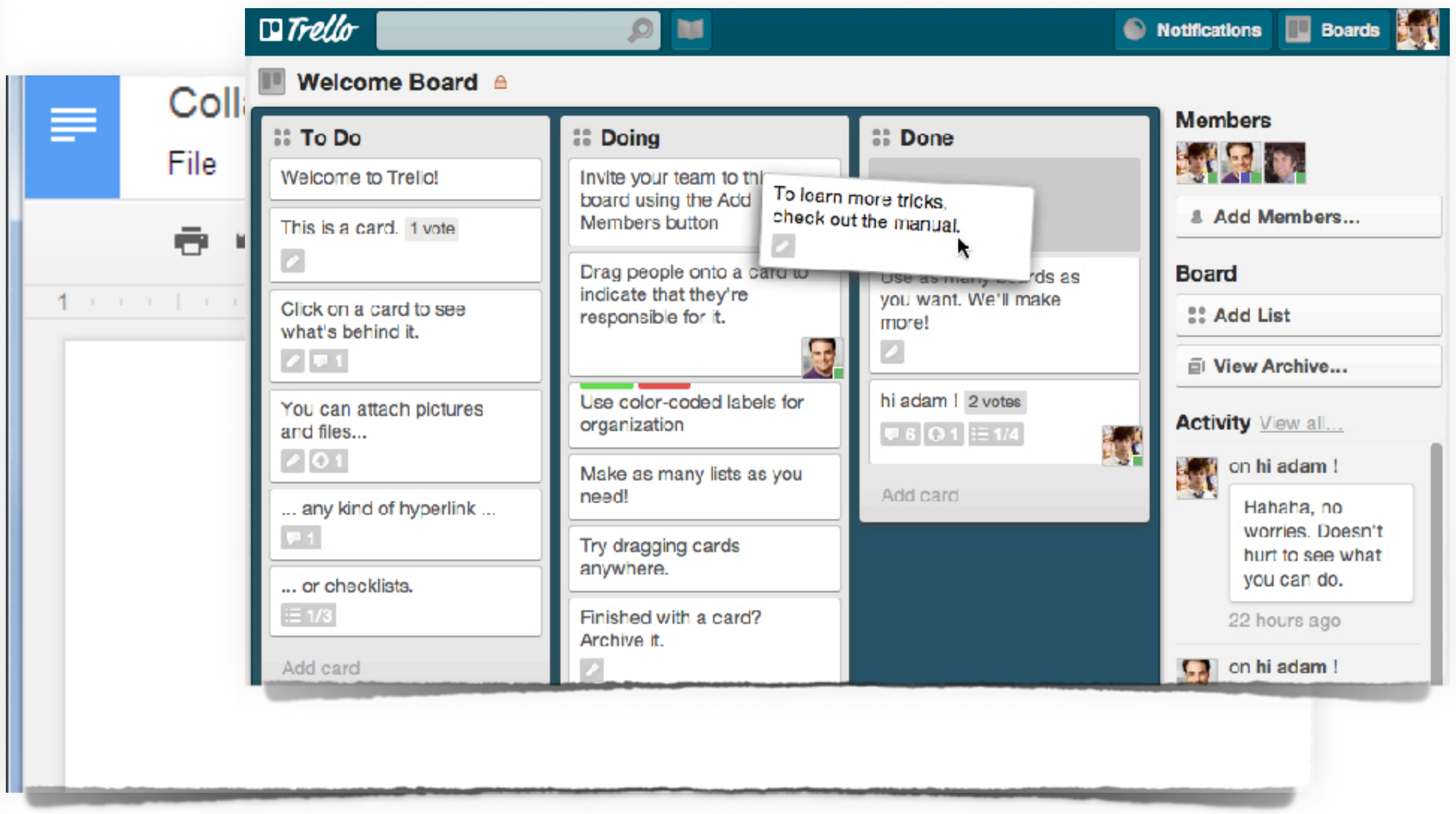
Which poses a challenge for app programmers.



I'm typing in real time

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Which poses a challenge for app programmers.



To Do

Welcome to Trello!

This is a card. 1 vote

Click on a card to see what's behind it.

You can attach pictures and files...

... any kind of hyperlink ...

... or checklists. 1/3

Add card

Doing

Invite your team to thi board using the Add Members button

Drag people onto a card to indicate that they're responsible for it.

Use color-coded labels for organization

Make as many lists as you need!

Try dragging cards anywhere.

Finished with a card? Archive it.

Done

To learn more tricks, check out the manual.

Use as many boards as you want. We'll make more!

hi adam ! 2 votes

Add card

Members

Add Members...

Board

Add List

View Archive...

Activity View all...

on hi adam !

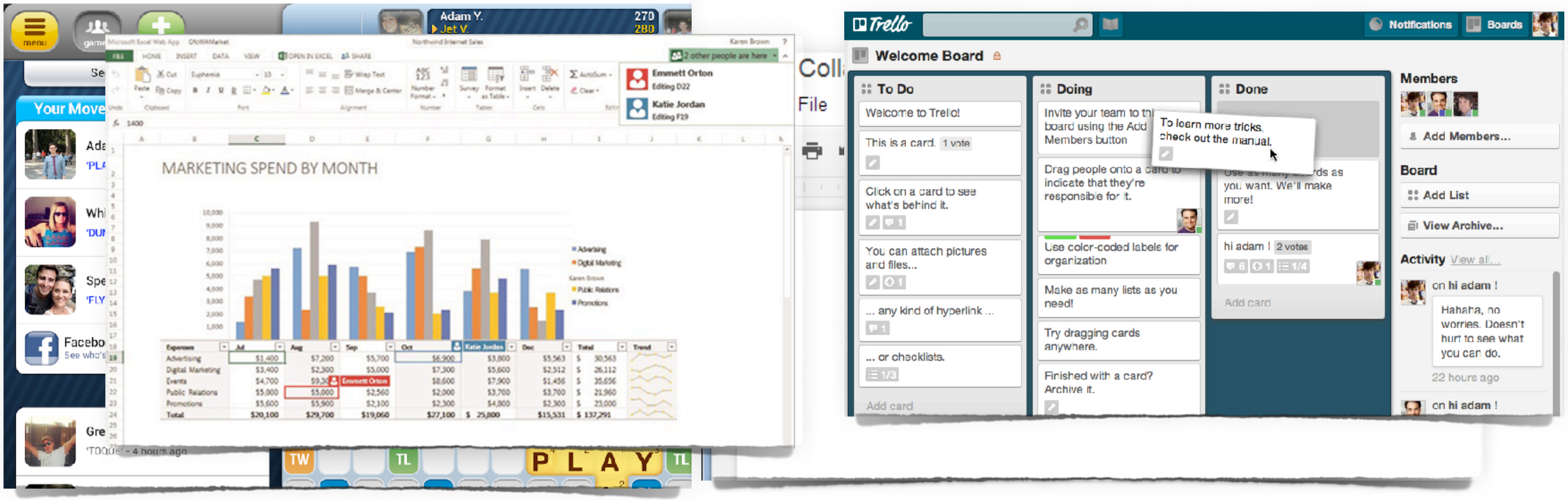
Hahaha, no worries. Doesn't hurt to see what you can do.

22 hours ago

on hi adam !

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Which poses a challenge for app programmers.



MARKETING SPEND BY MONTH

Expenses	Jul	Aug	Sep	Oct	Nov	Dec	Total
Advertising	$1,400	$7,200	$5,700	$6,900	$3,800	$5,563	$ 30,563
Digital Marketing	$3,400	$2,300	$5,000	$7,300	$5,600	$2,512	$ 26,112
Events	$4,700	$9,3[illegible]	[illegible]	$8,600	$7,900	$1,456	$ 35,656
Public Relations	$5,000	$5,000	$2,560	$2,000	$3,700	$3,700	$ 21,960
Promotions	$5,600	$5,900	$2,100	$2,300	$4,800	$2,300	$ 23,000
Total	$20,100	$29,700	$19,060	$27,100	$ 25,800	$15,531	$ 137,291

To Do
- Welcome to Trello!
- This is a card. 1 vote
- Click on a card to see what's behind it.
- You can attach pictures and files...
- ... any kind of hyperlink ...
- ... or checklists. 1/3
- Add card

Doing
- Invite your team to this board using the Add Members button
- Drag people onto a card to indicate that they're responsible for it.
- Use color-coded labels for organization
- Make as many lists as you need!
- Try dragging cards anywhere.
- Finished with a card? Archive it.

Done
- To learn more tricks, check out the manual.
- Use as many boards as you want. We'll make more!
- hi adam ! 2 votes 8 1 1/4
- Add card

on hi adam !
Hahaha, no worries. Doesn't hurt to see what you can do.
22 hours ago
on hi adam !

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.

Diamond

Diamond is the first <u>reactive data management service</u>, which provides the following guarantees:

- Ensures updates to shared data are consistent and durable
- Coordinates and synchronizes updates reliably across mobile clients and cloud storage
- Automatically triggers application code in response to updates to shared data

Talk Outline

Diamond System & Programming Model

What does Diamond provide for reactive apps?

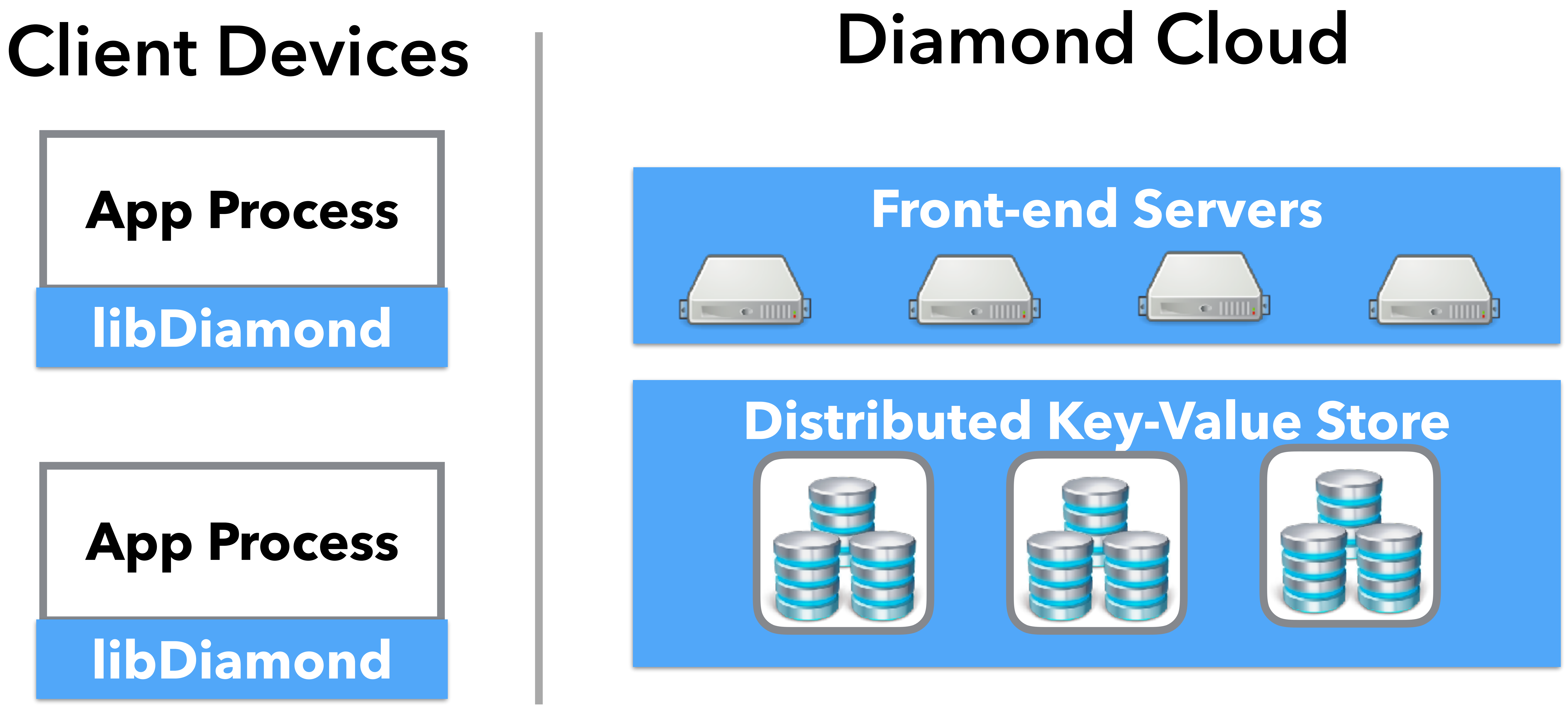
Diamond Guarantees & Implementation

What does Diamond guarantee for reactive apps?

Evaluation

How does Diamond impact app complexity and performance?

Diamond System Model



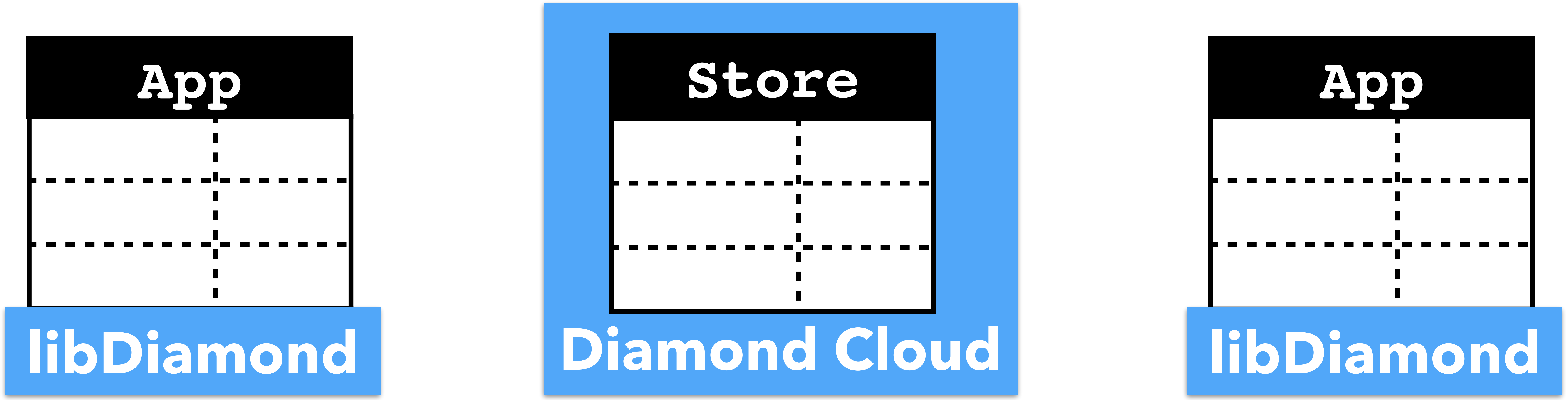
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

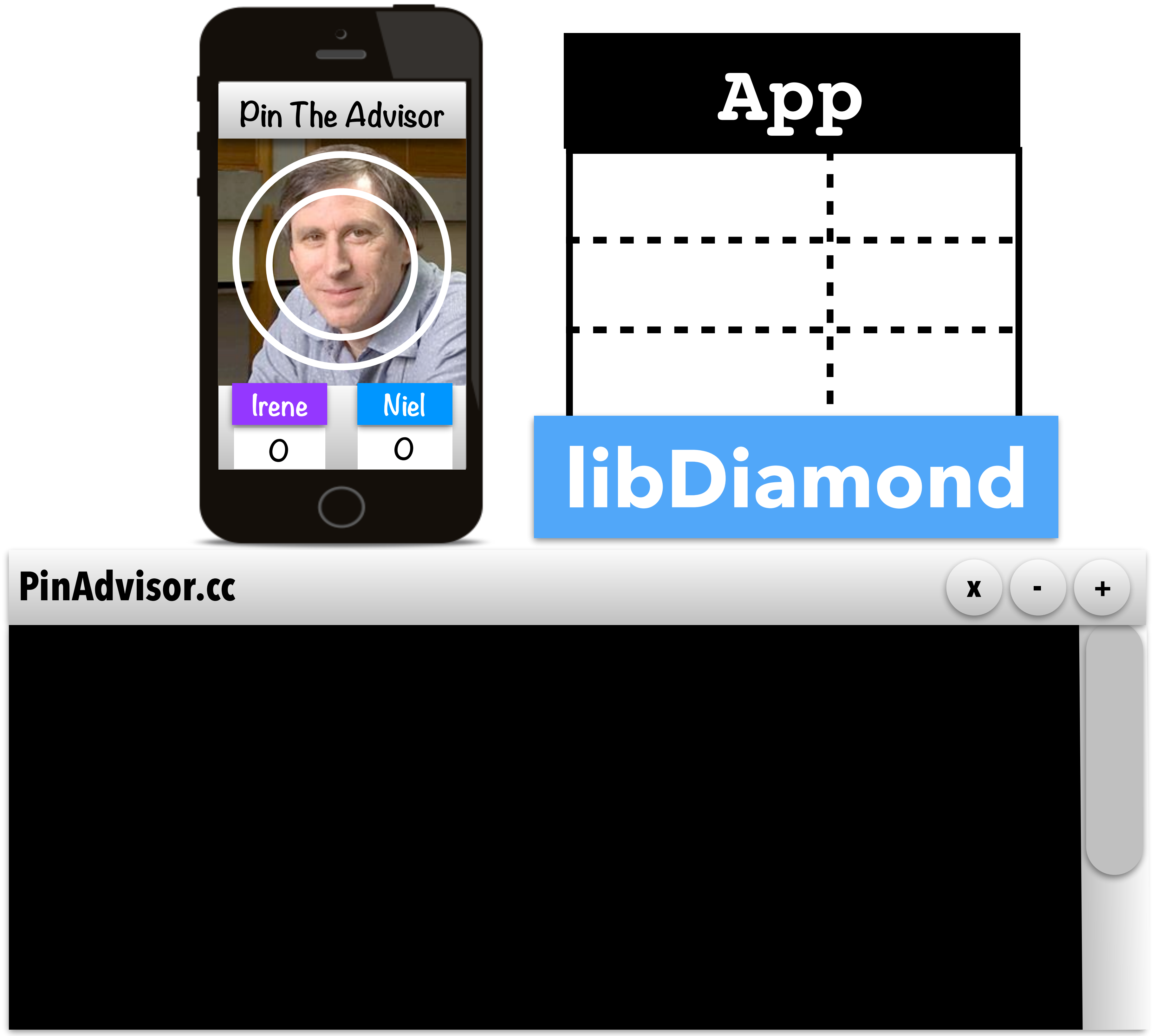
Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



Reactive Data Types (RDTs)

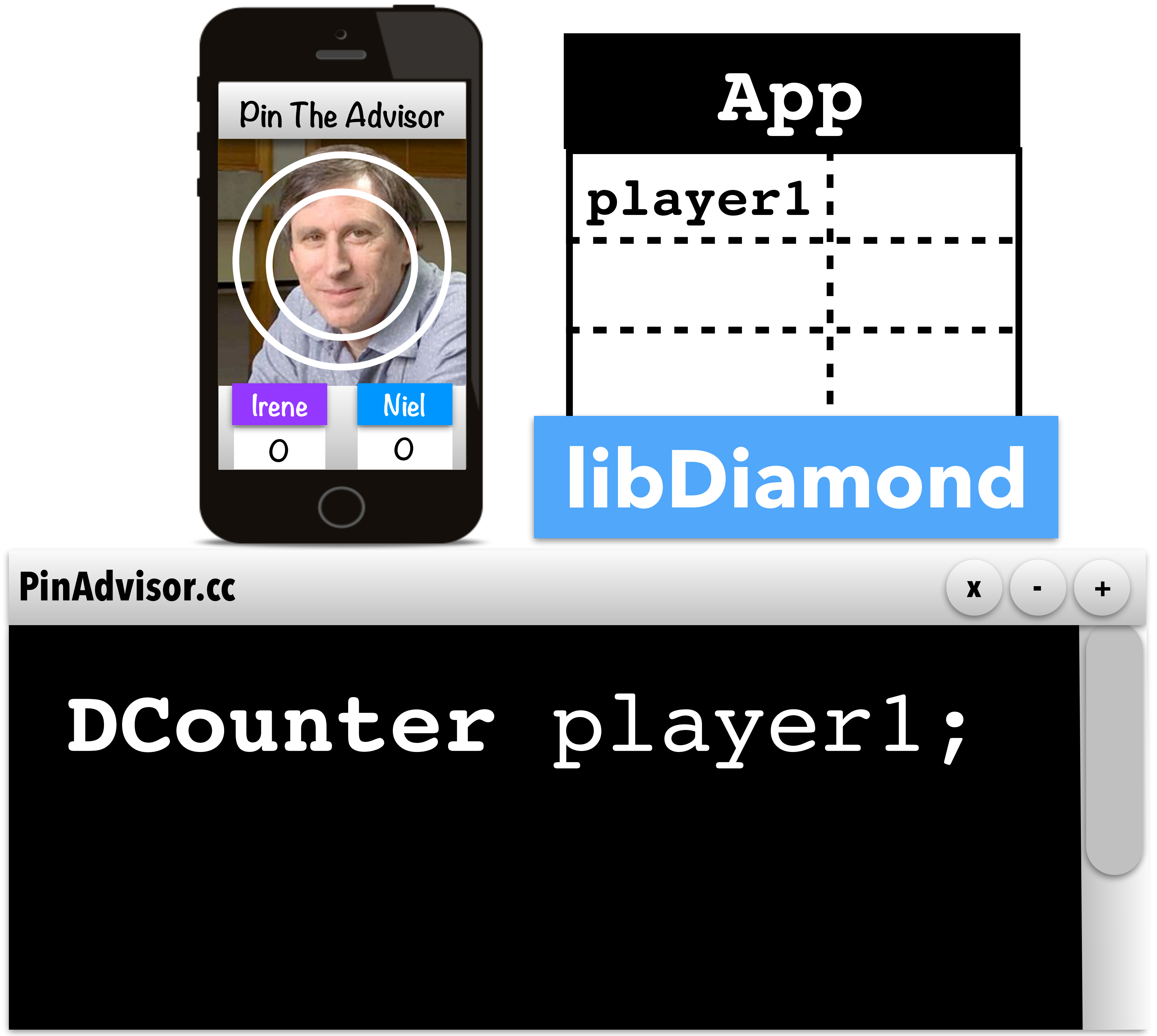
Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings



Reactive Data Types (RDTs)
Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings

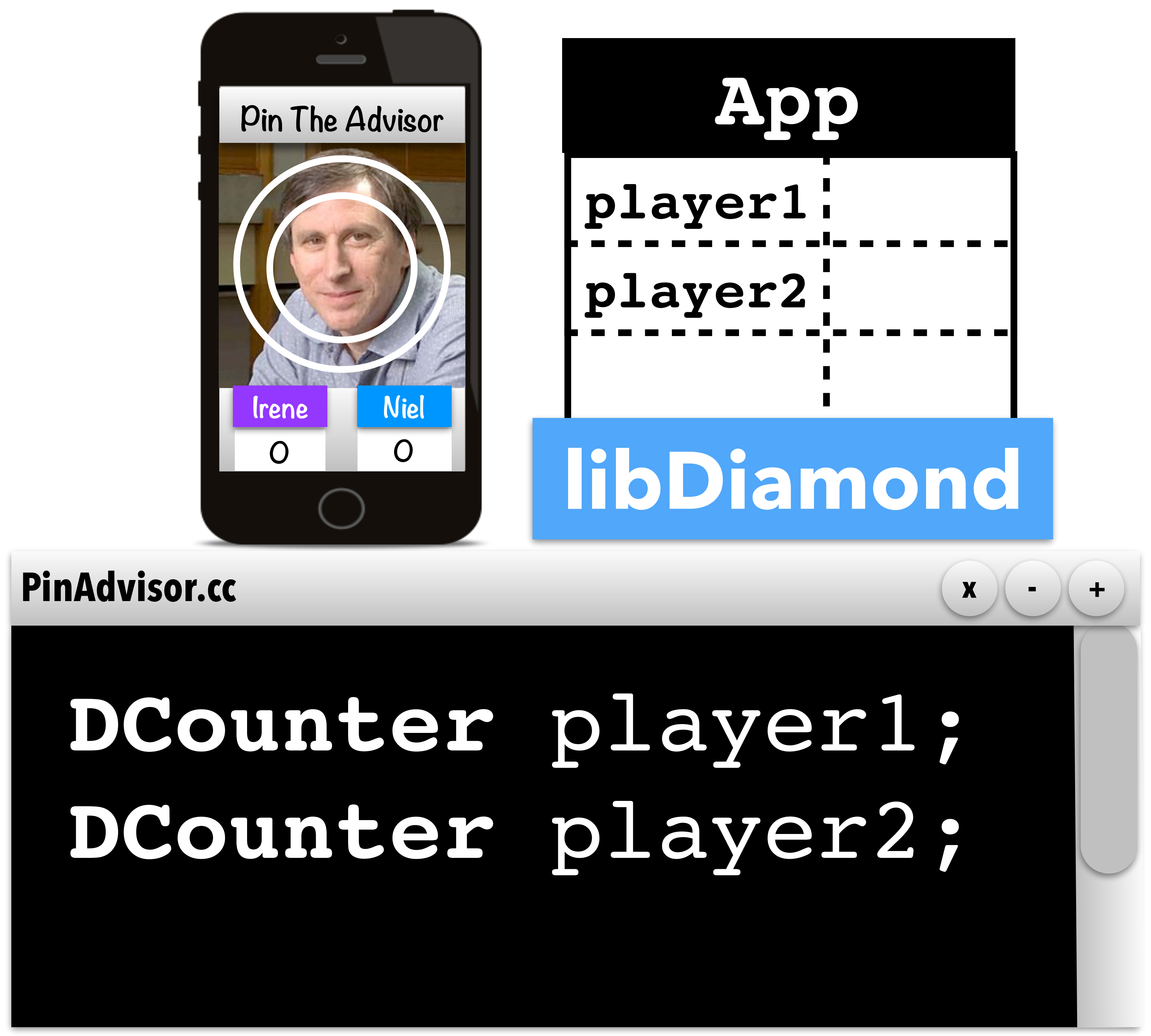


```
DCounter player1;
```

Reactive Data Types (RDTs)

Shared, persistent data structures

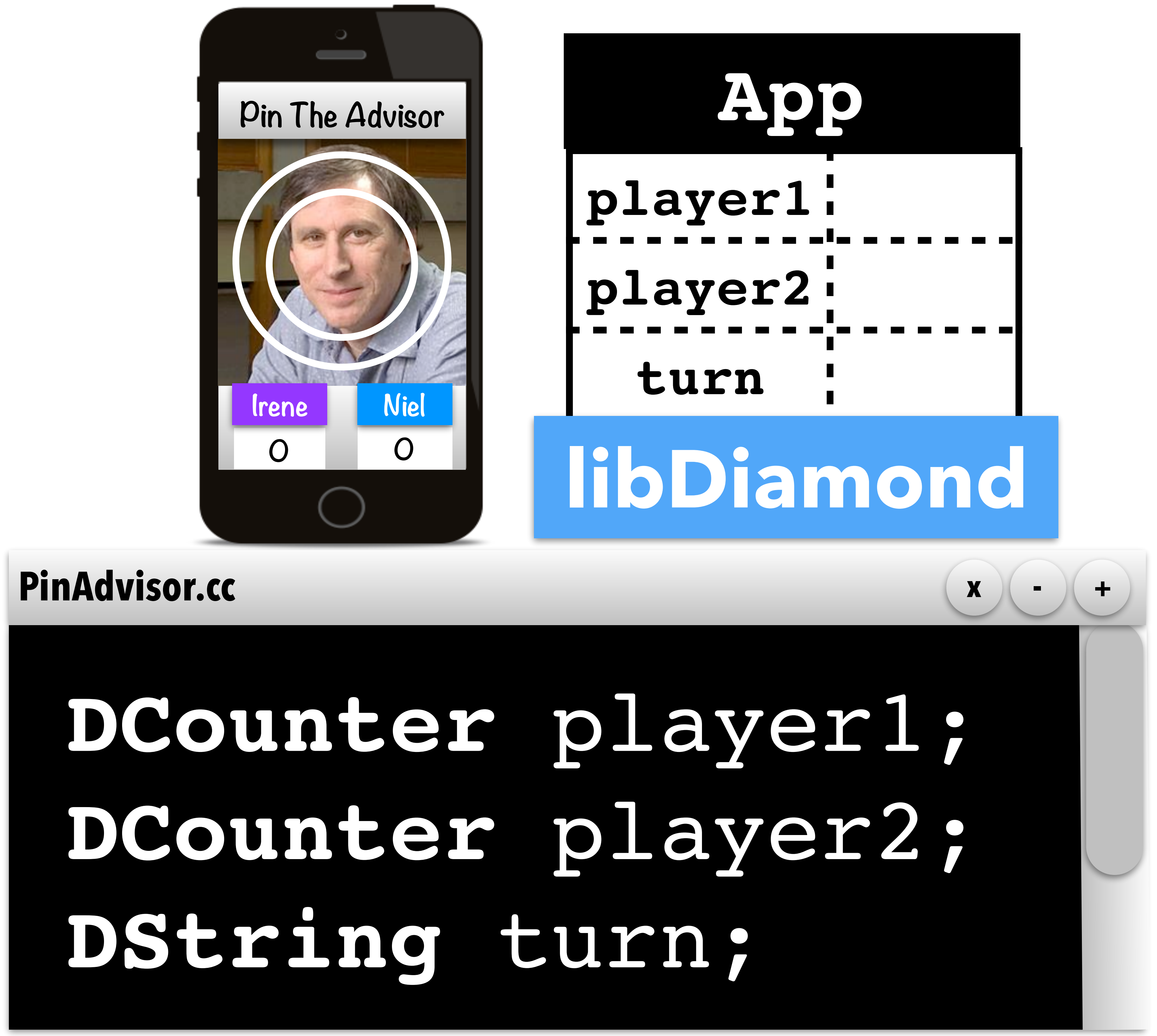
- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings



```
DCounter player1;
DCounter player2;
```

Reactive Data Types (RDTs)
Shared, persistent data structures

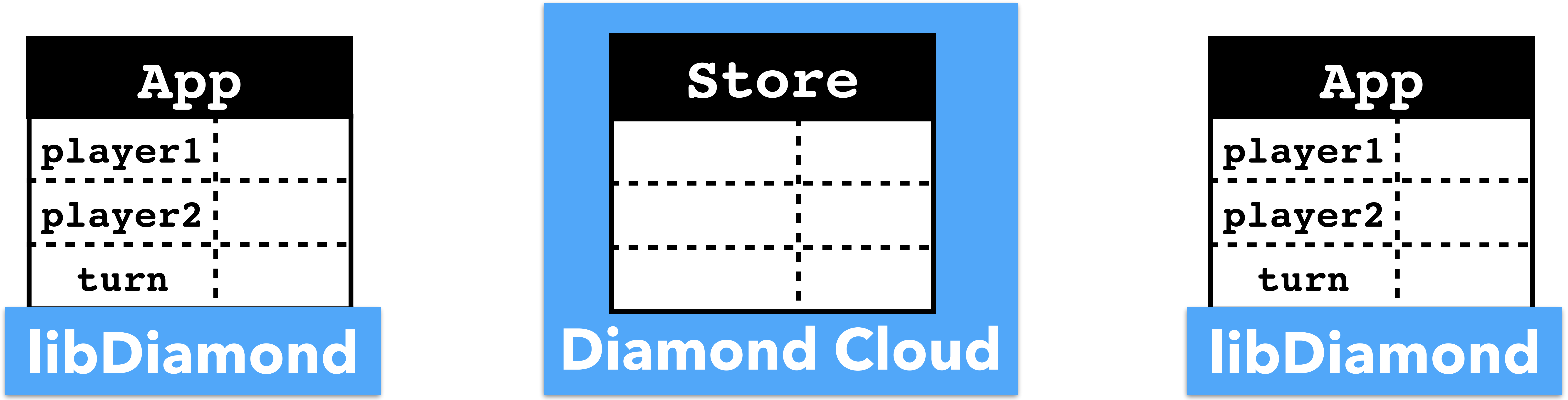
- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings



```
DCounter player1;
DCounter player2;
DString turn;
```

Diamond Programming Model

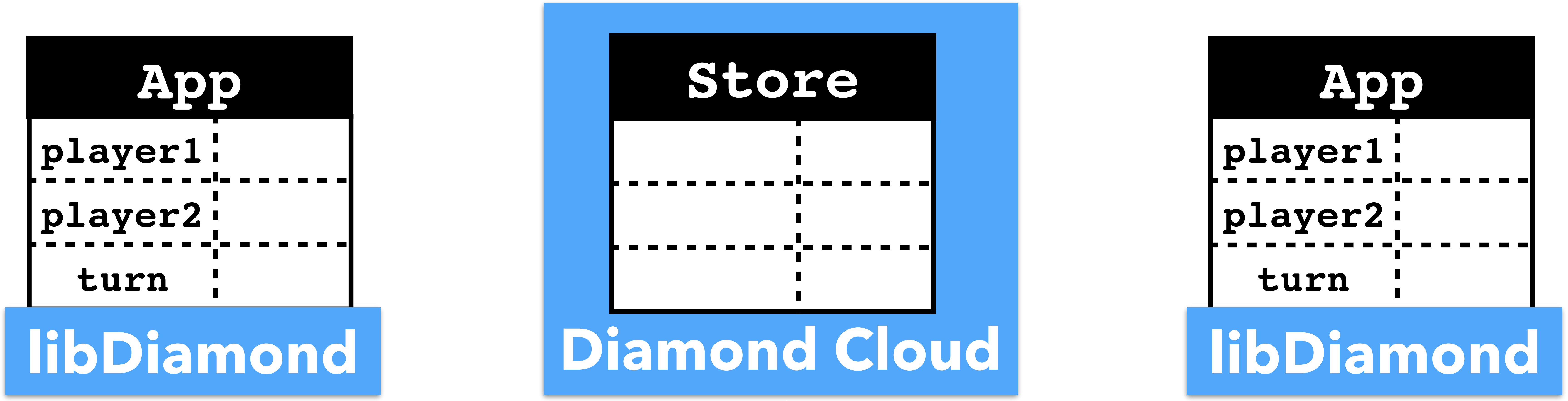
Reactive Data Types (RDTs)
Shared, persistent data structures



Diamond Programming Model

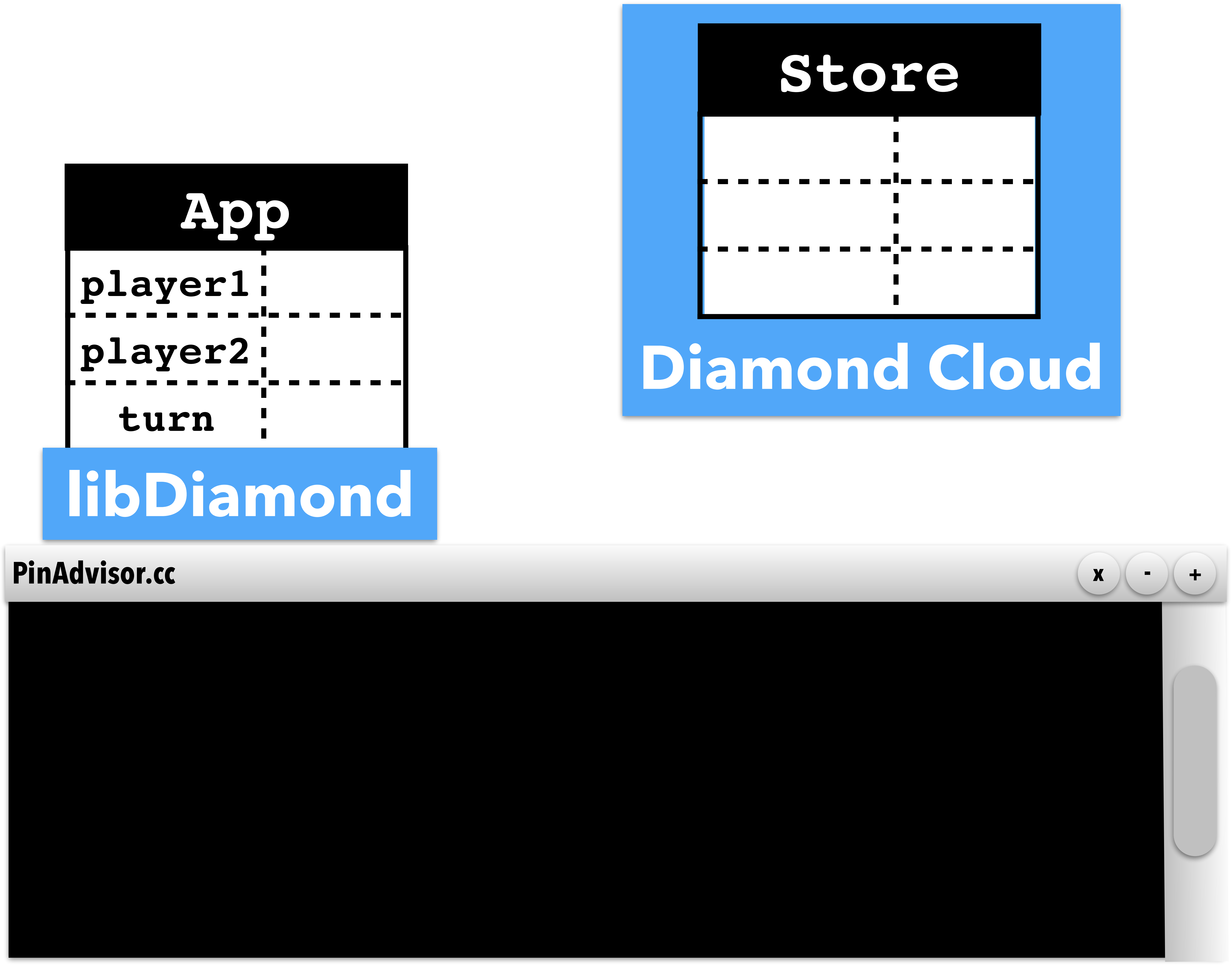
Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store



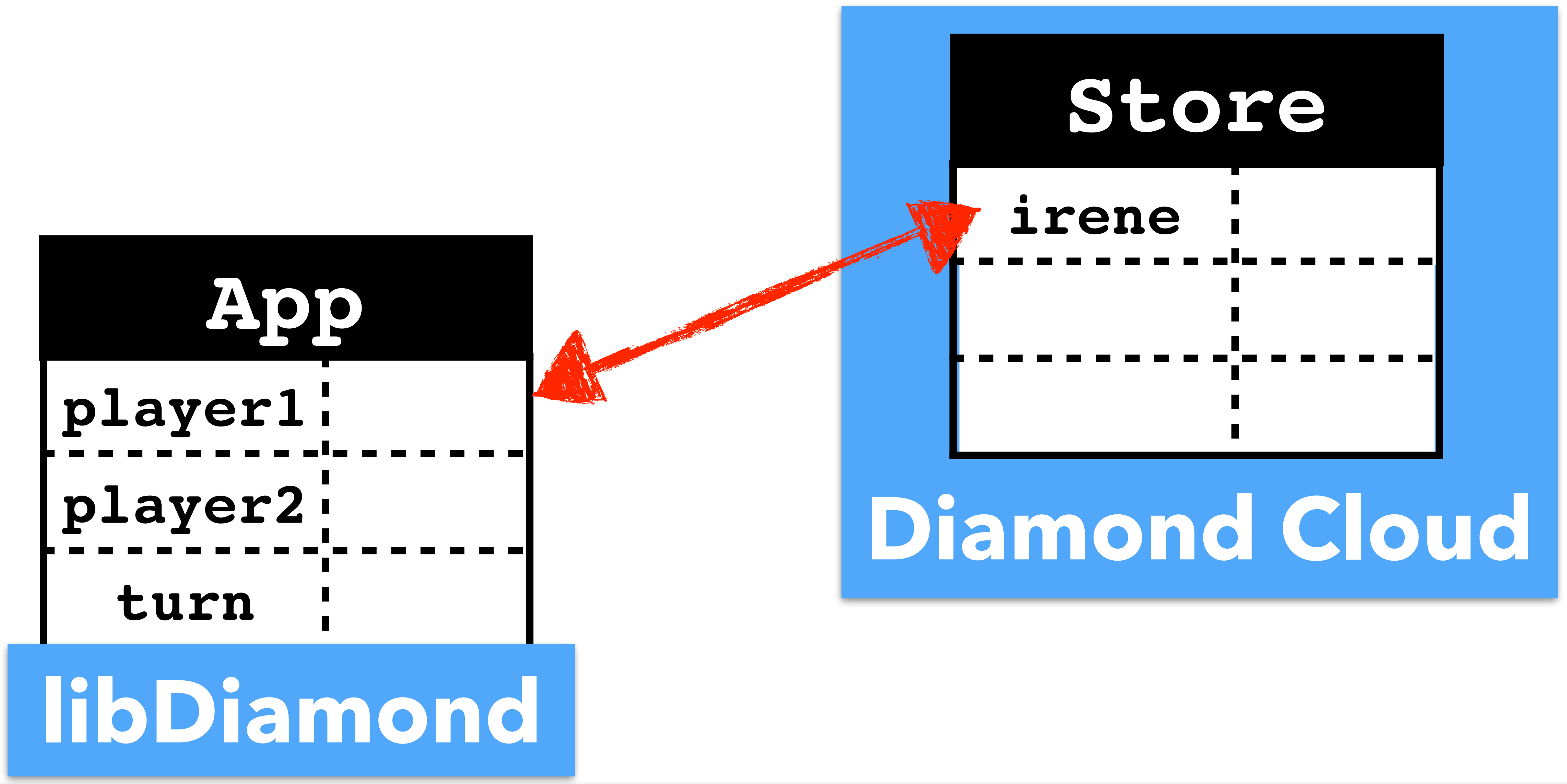
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

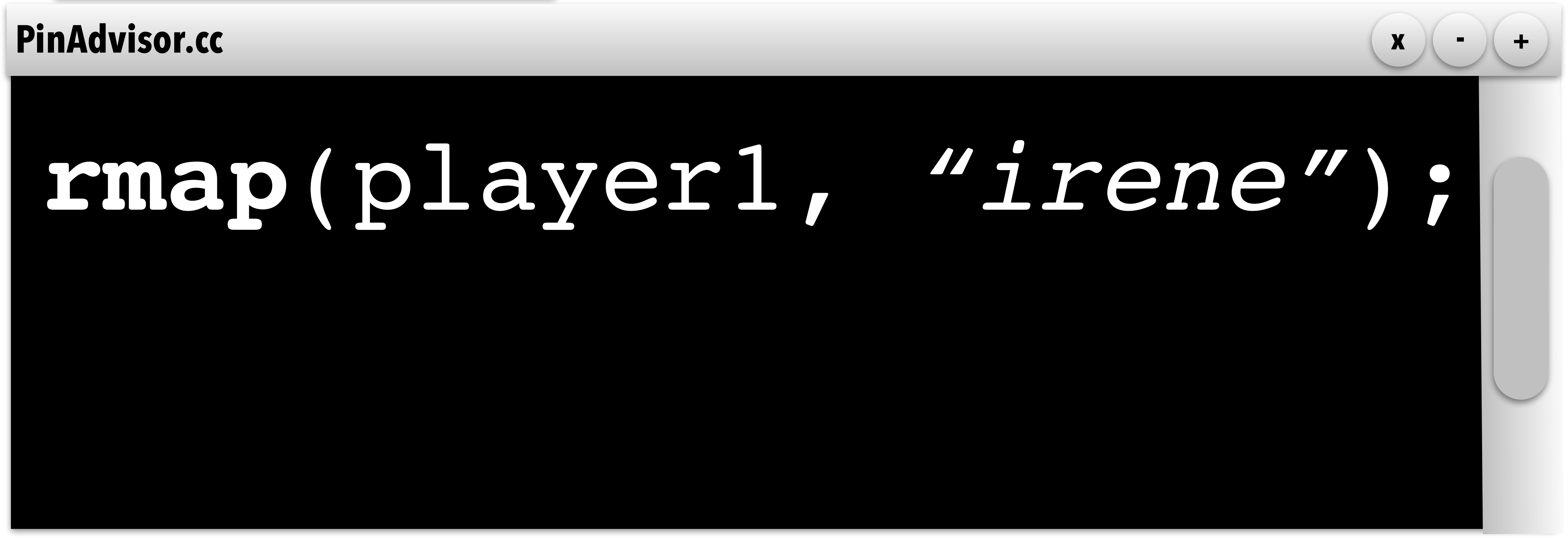
- Key abstraction for providing <u>flexible, shared memory</u>
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to <u>automatically</u> provide availability, fault-tolerance and consistency to RDTs



Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

- Key abstraction for providing flexible, shared memory
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to automatically provide availability, fault-tolerance and consistency to RDTs

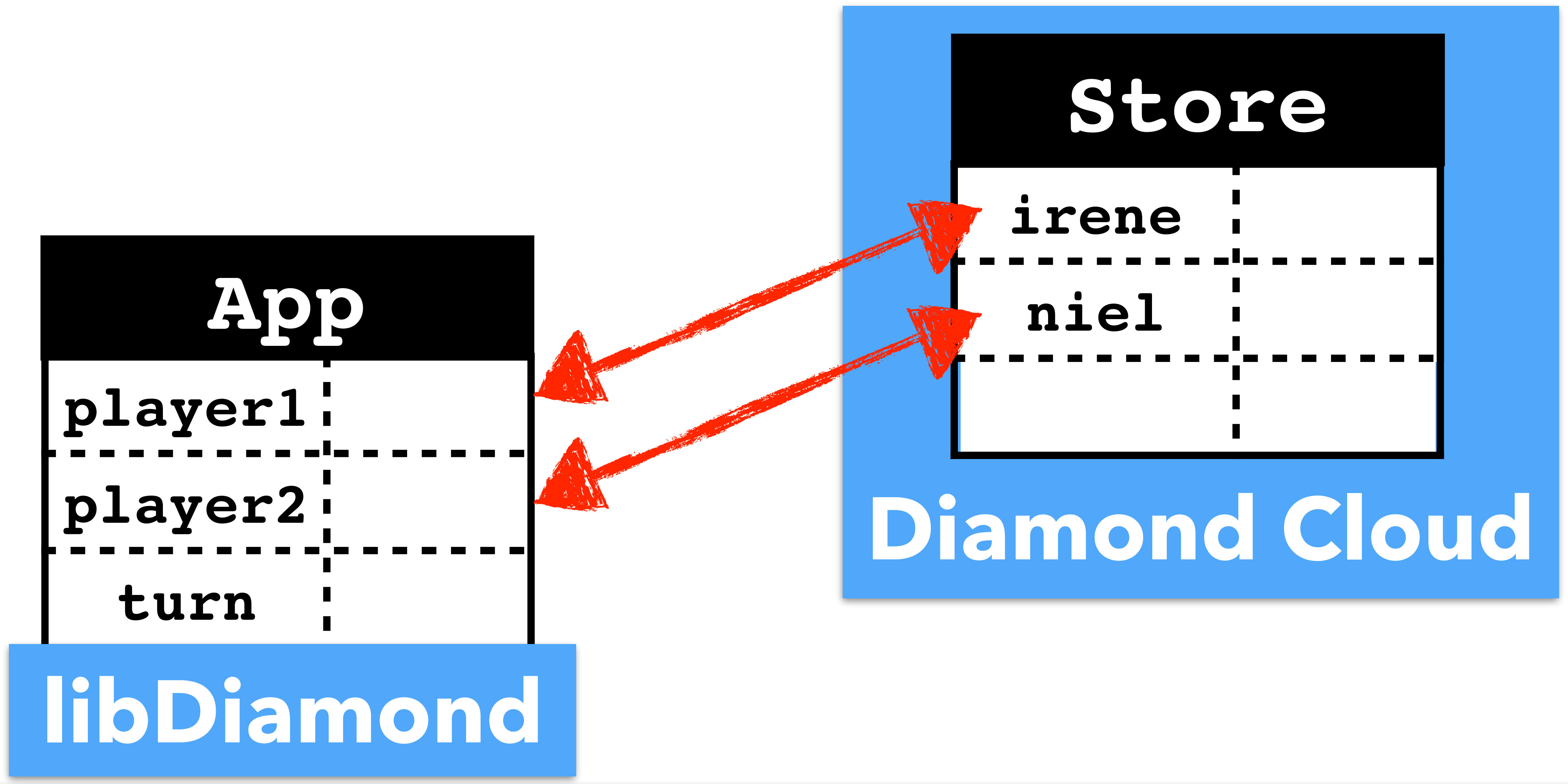


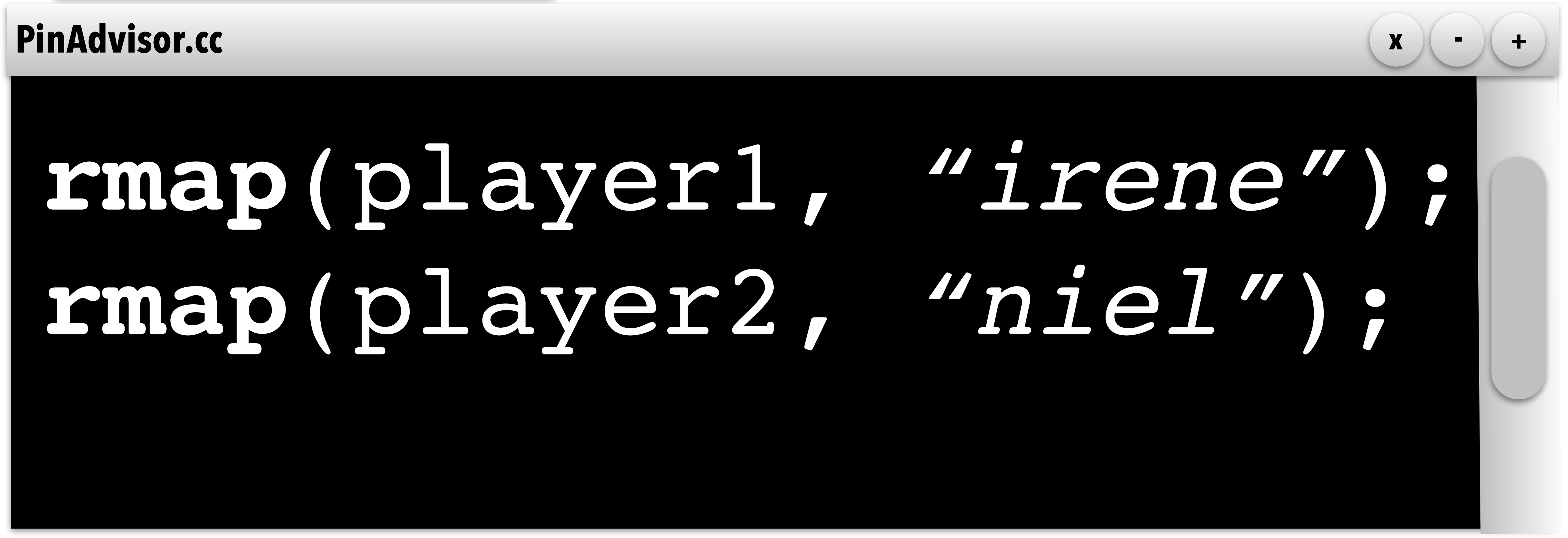


```
rmap(player1, "irene");
```

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

- Key abstraction for providing <u>flexible, shared memory</u>
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to <u>automatically</u> provide availability, fault-tolerance and consistency to RDTs

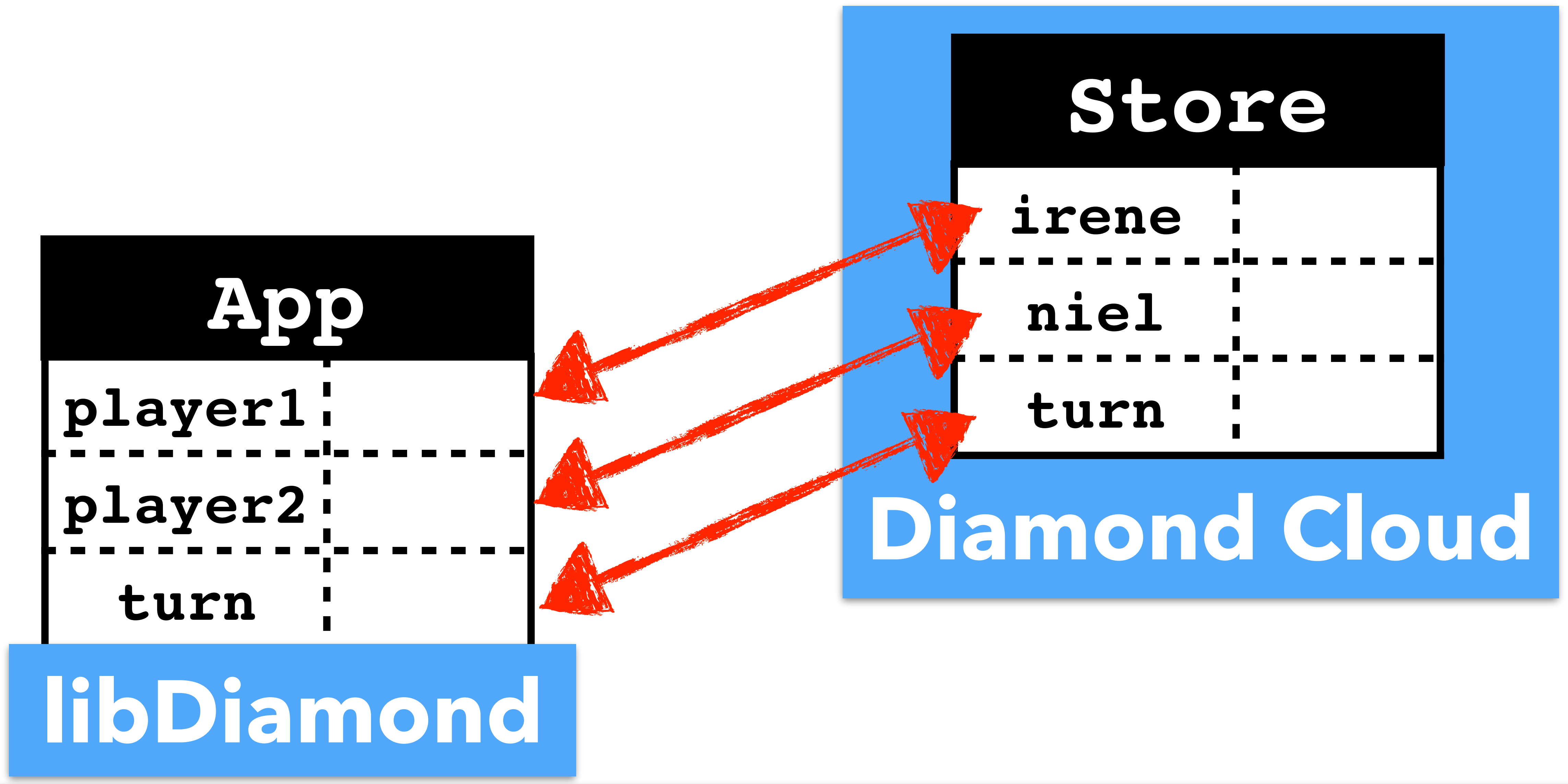


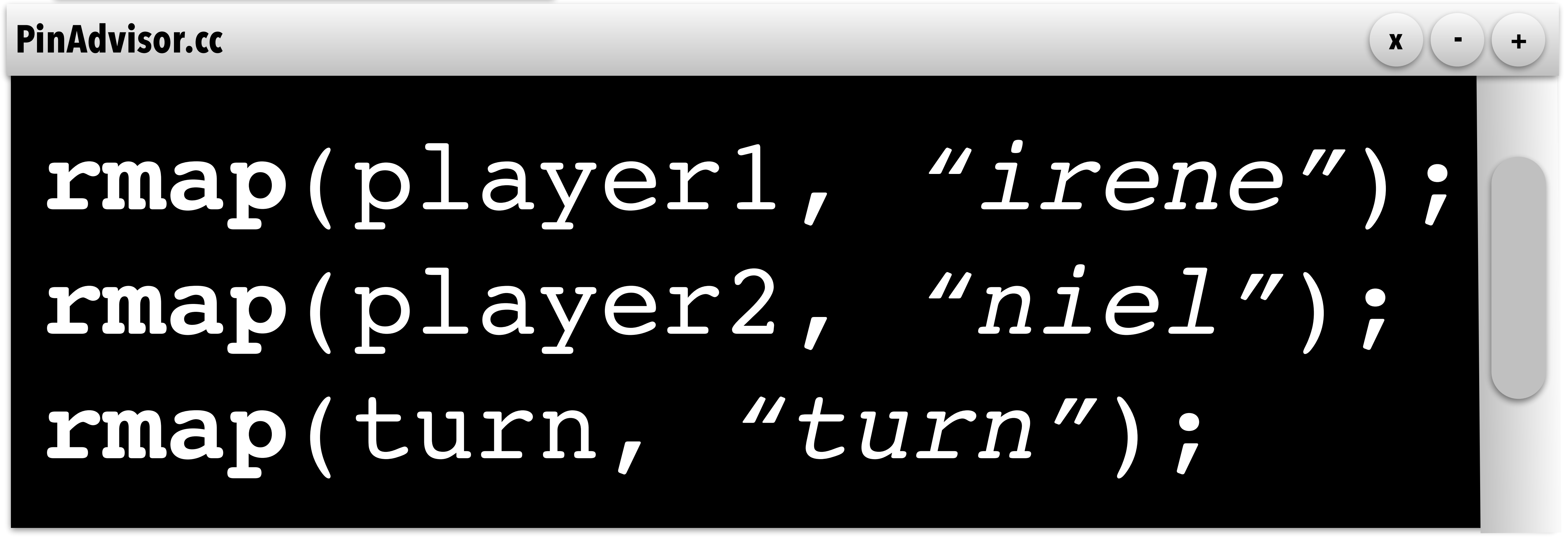


```
rmap(player1, "irene");
rmap(player2, "niel");
```

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

- Key abstraction for providing <u>flexible, shared memory</u>
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to <u>automatically</u> provide availability, fault-tolerance and consistency to RDTs



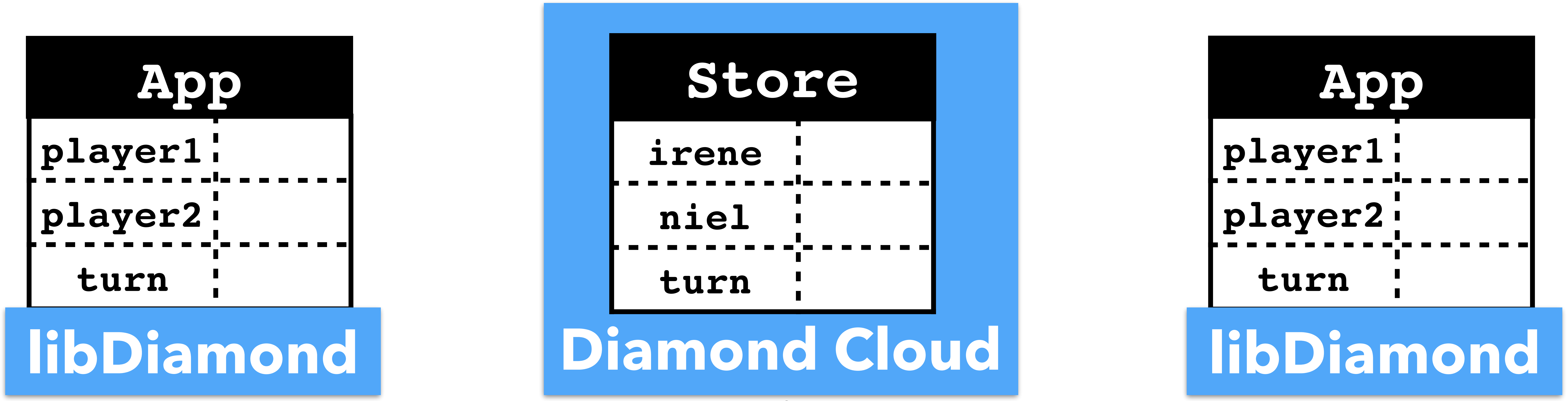


```
rmap(player1, “irene”);
rmap(player2, “niel”);
rmap(turn, “turn”);
```

Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

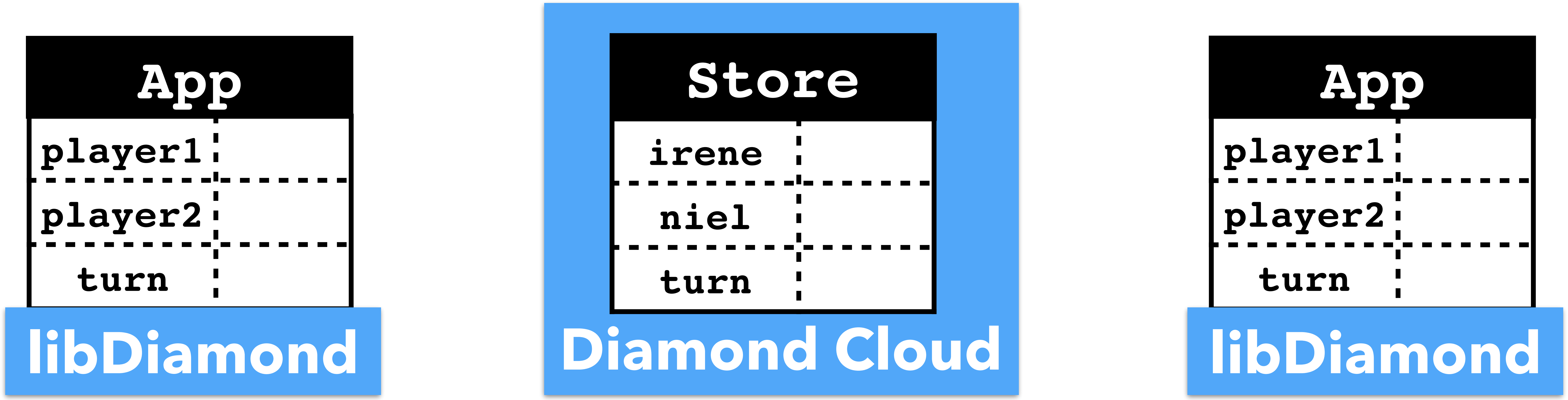


Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

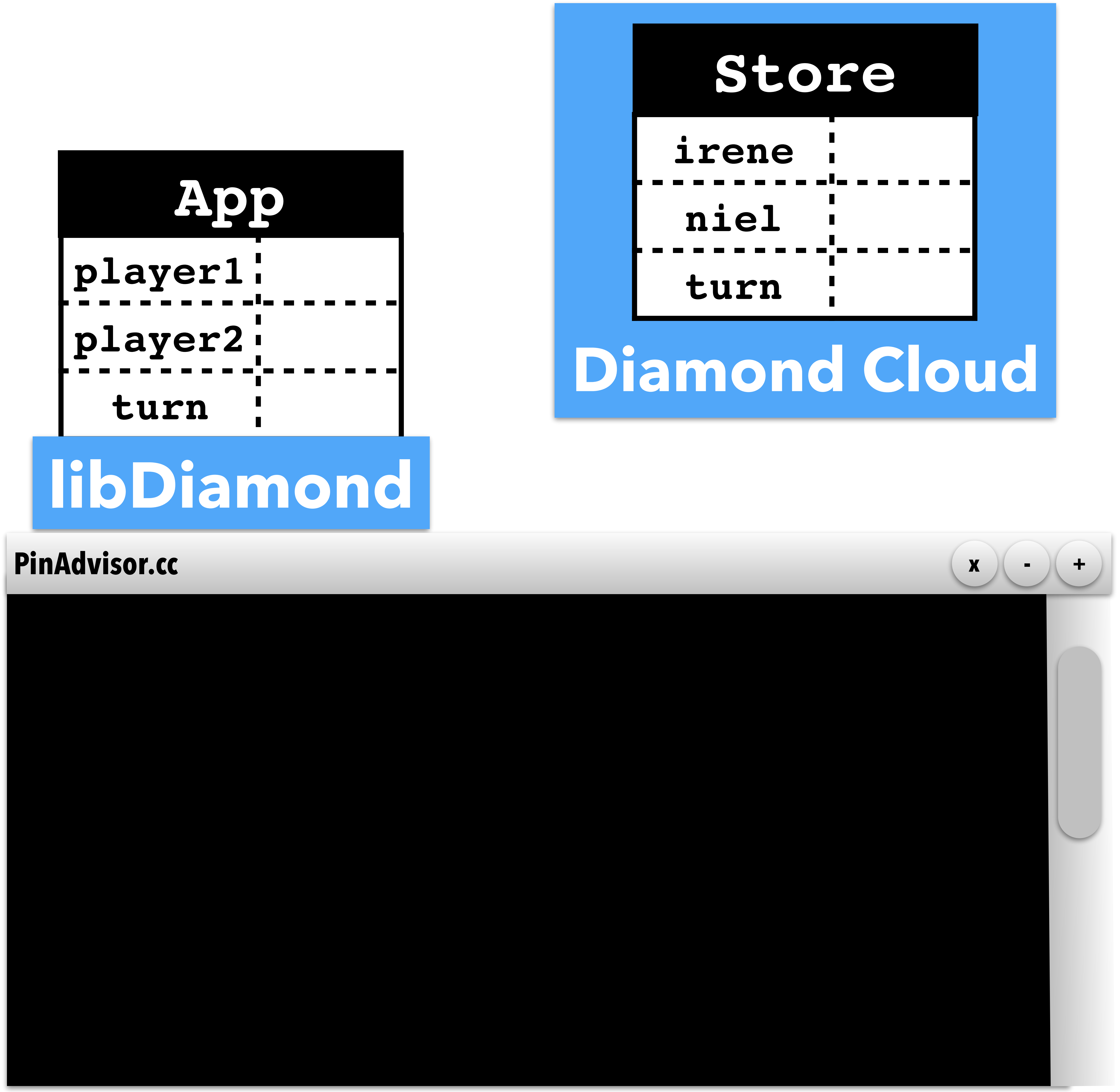
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.



Read-write Transactions
Read-write transactions to update shared RDTs.

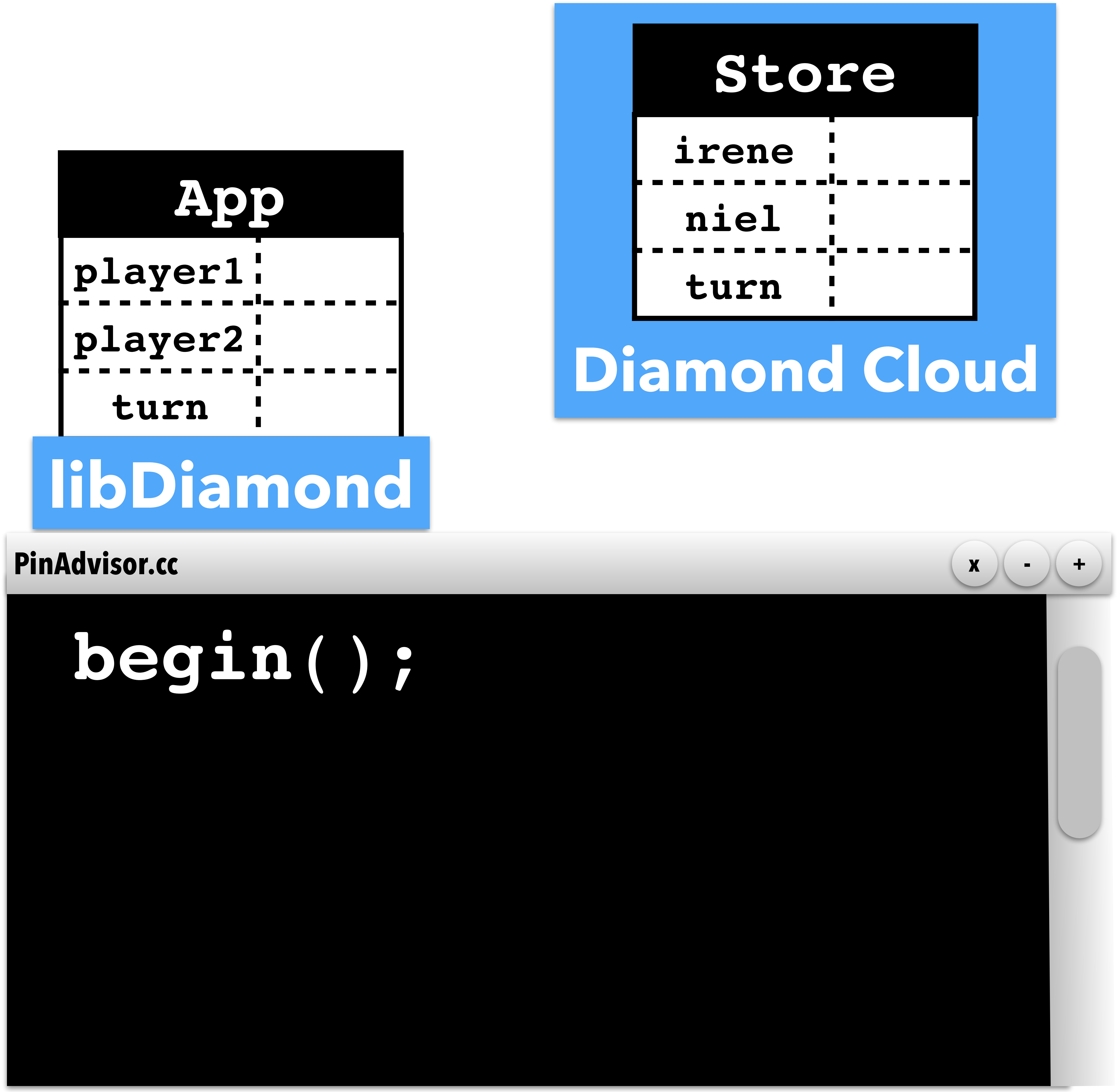
- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data



Read-write Transactions

Read-write transactions to update shared RDTs.

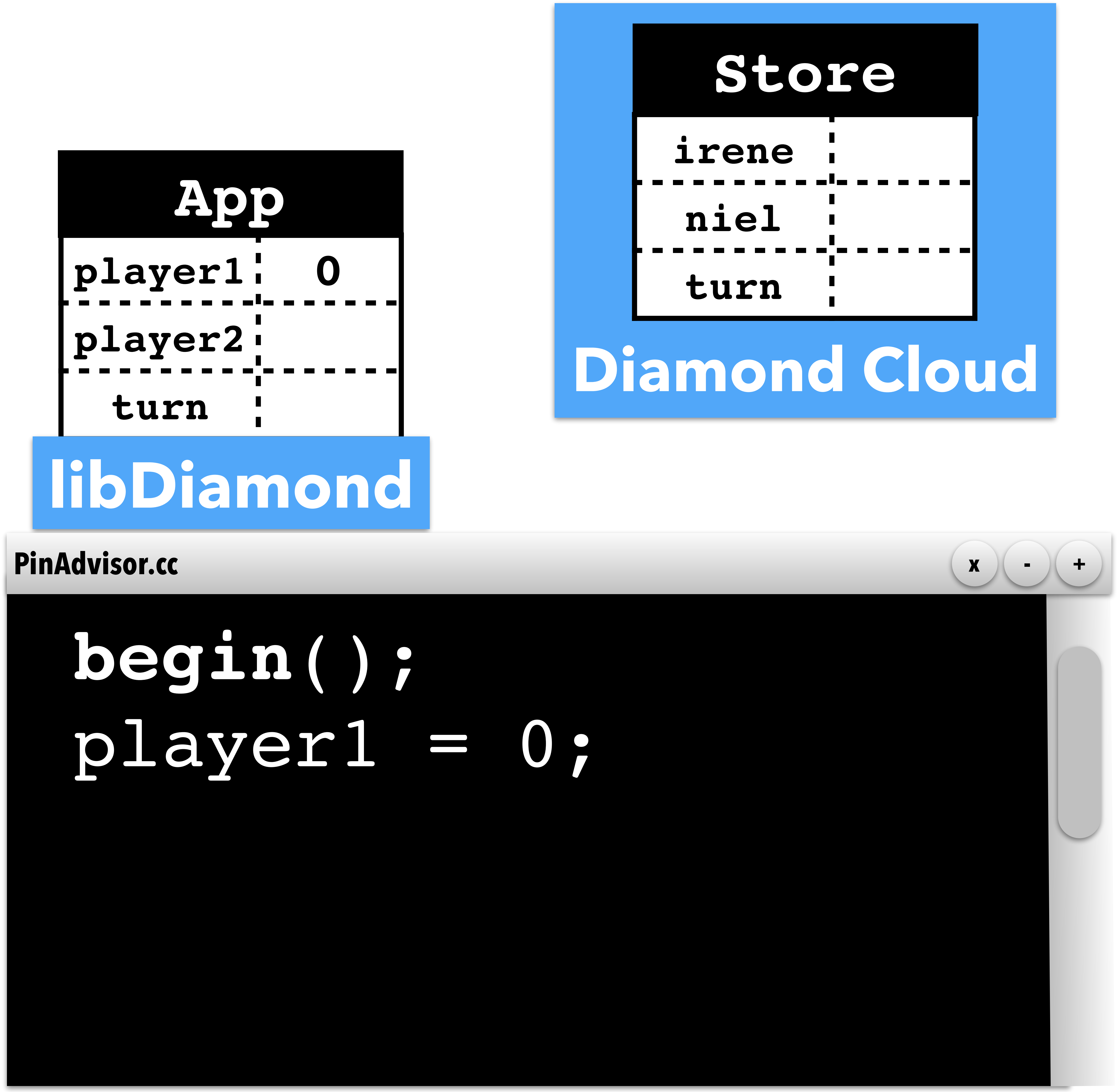
- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data



```
begin();
```

Read-write Transactions
Read-write transactions to update shared RDTs.

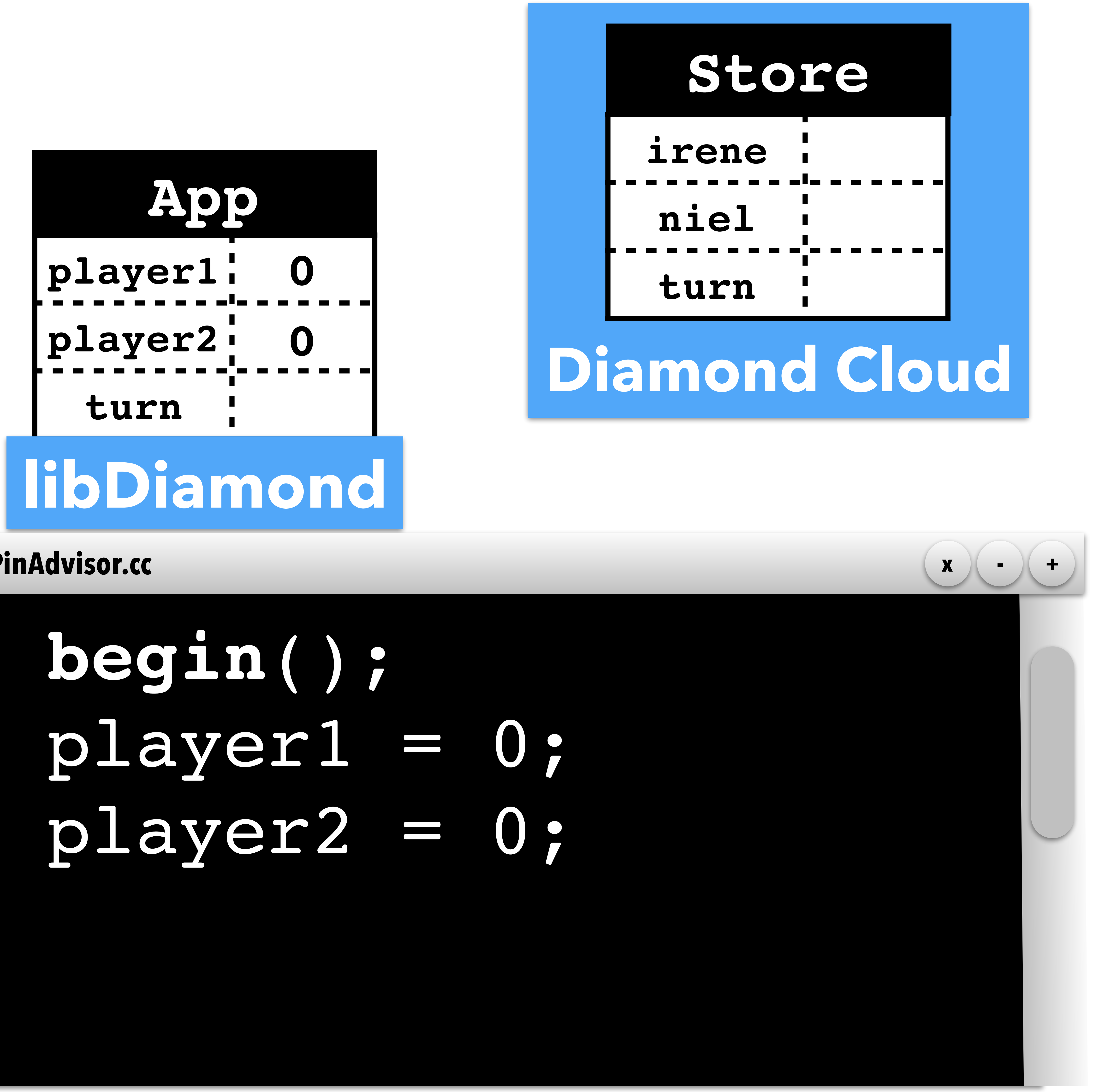
- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data



```
begin();
player1 = 0;
```

Read-write Transactions
Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data

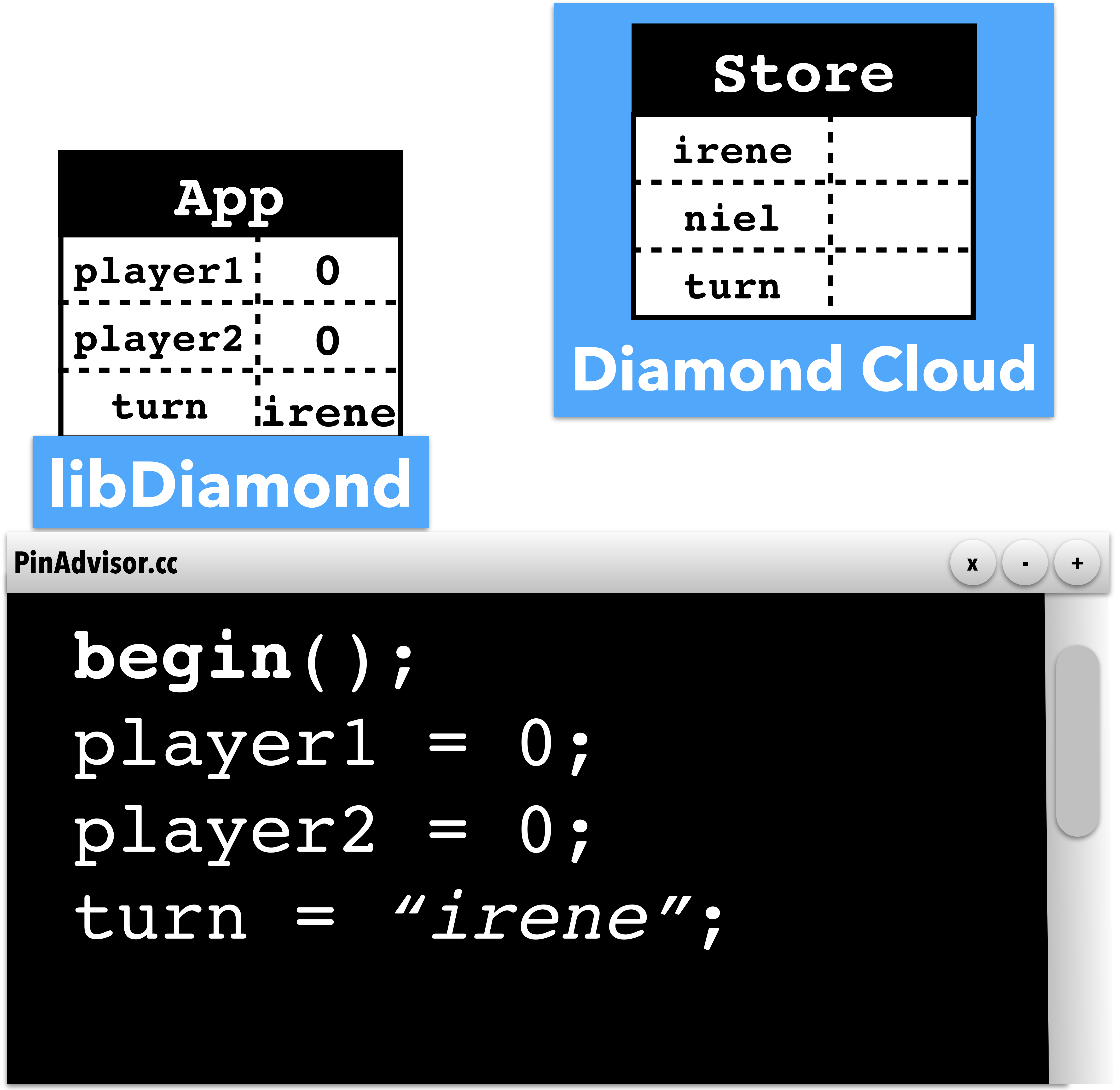


```
begin();
player1 = 0;
player2 = 0;
```

Read-write Transactions

Read-write transactions to update shared RDTs.

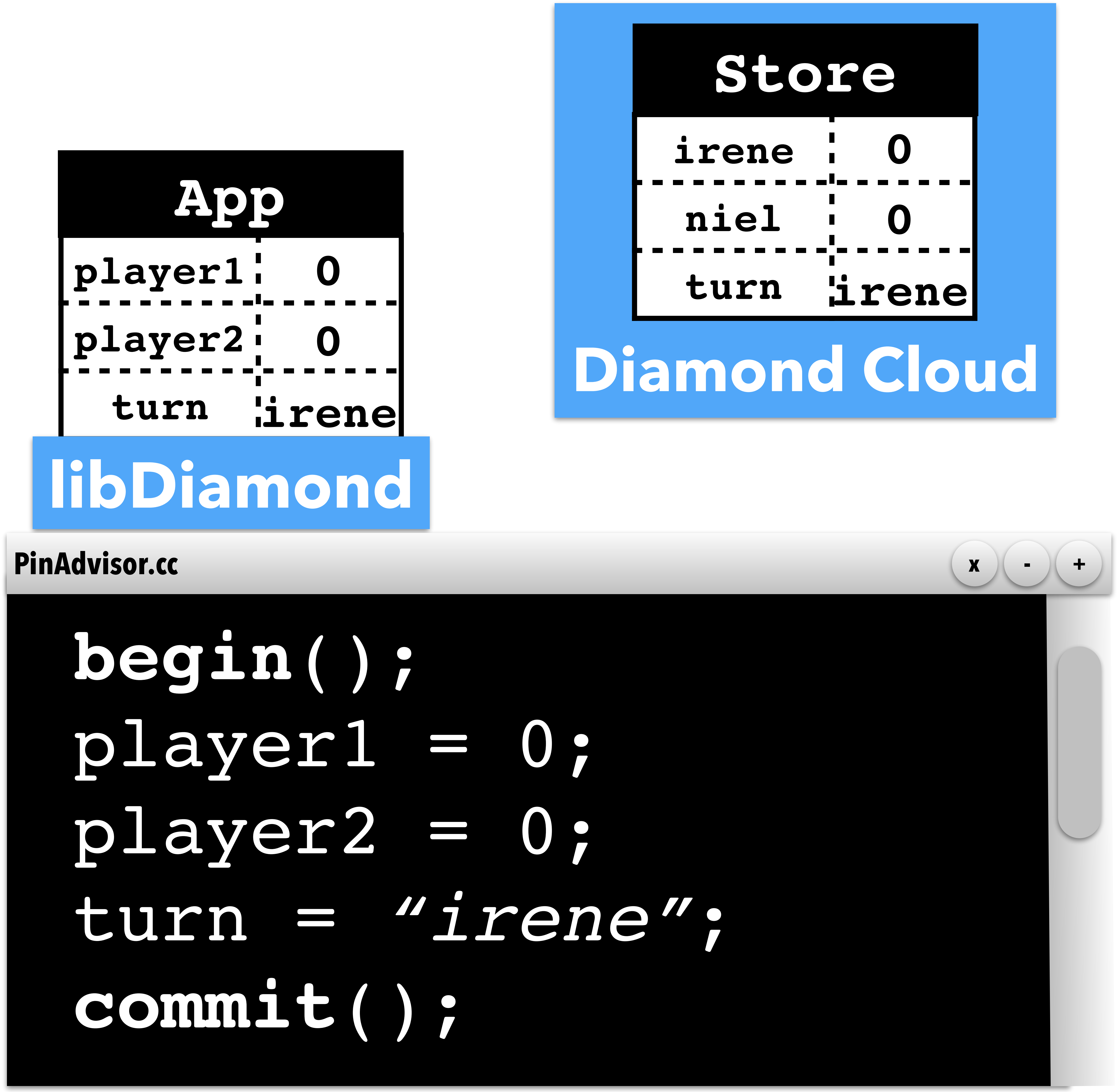
- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data



```
begin();
player1 = 0;
player2 = 0;
turn = “irene”;
```

Read-write Transactions
Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data



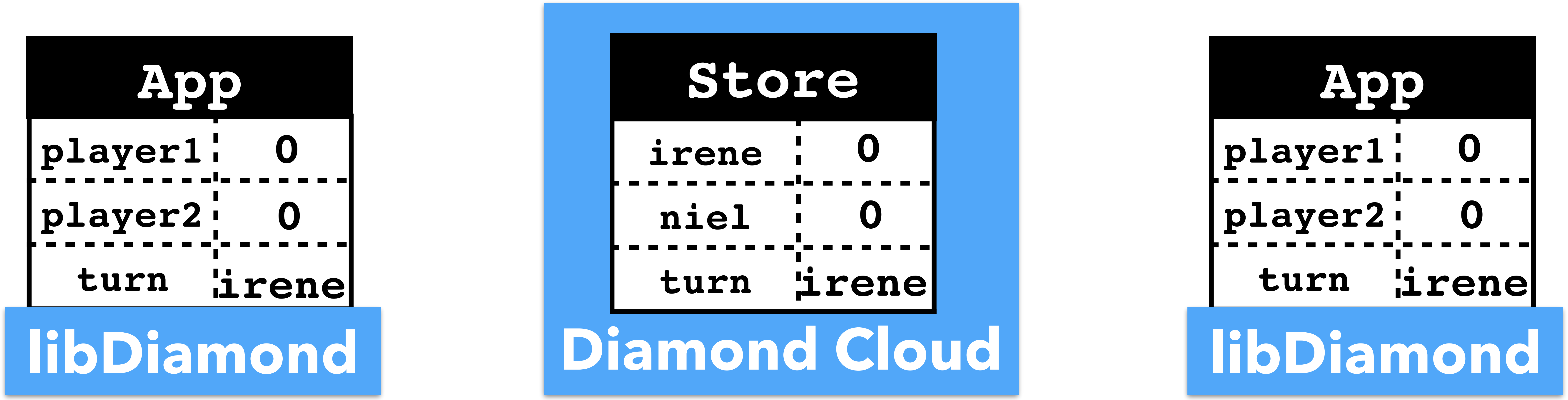
```
begin();
player1 = 0;
player2 = 0;
turn = "irene";
commit();
```

Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store



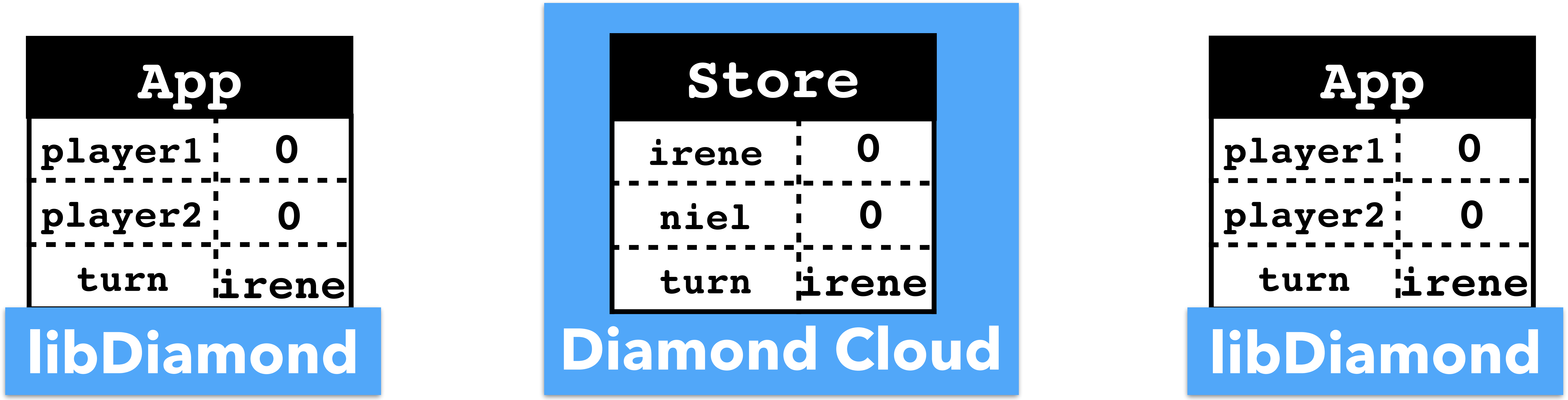
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

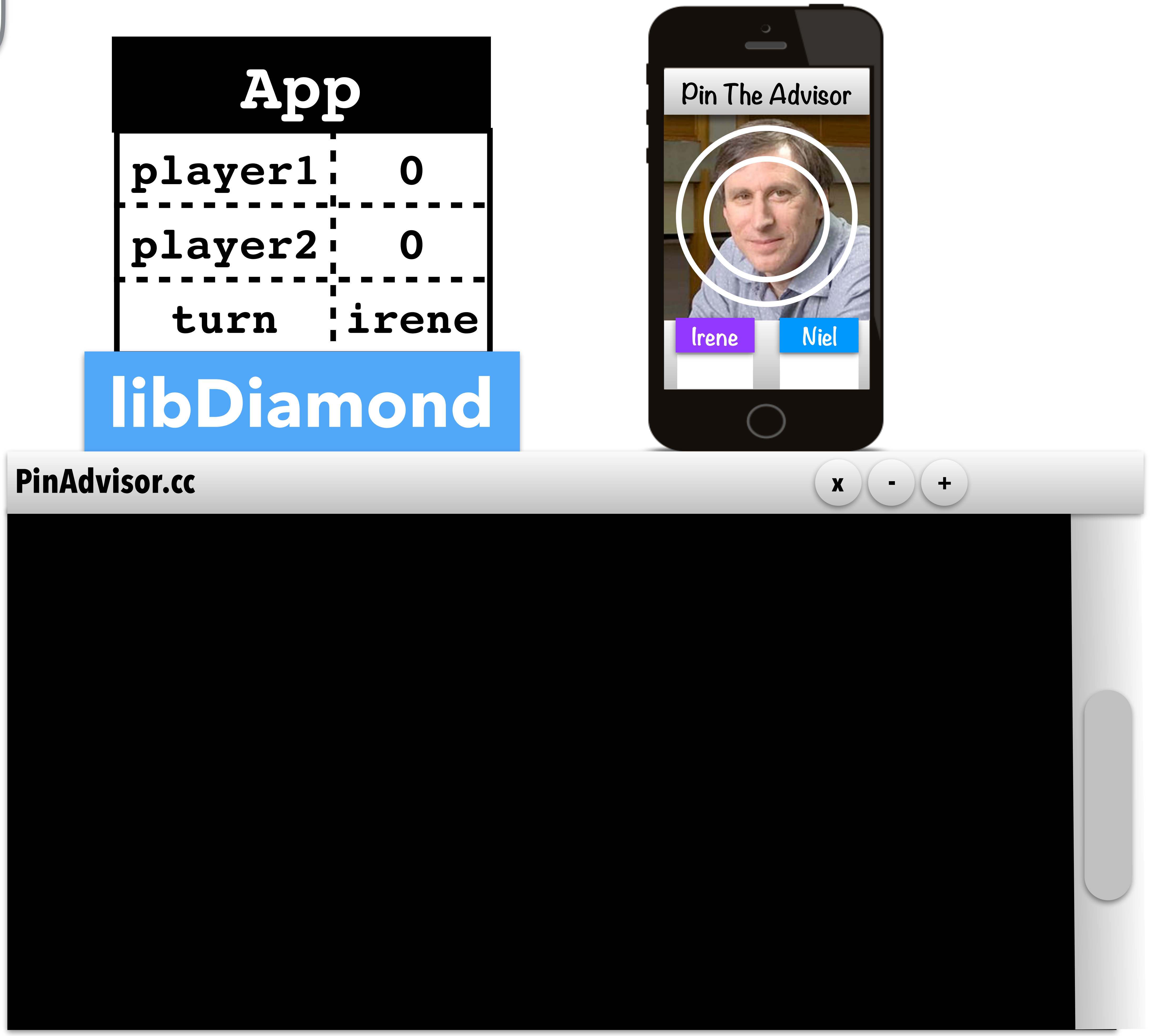
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

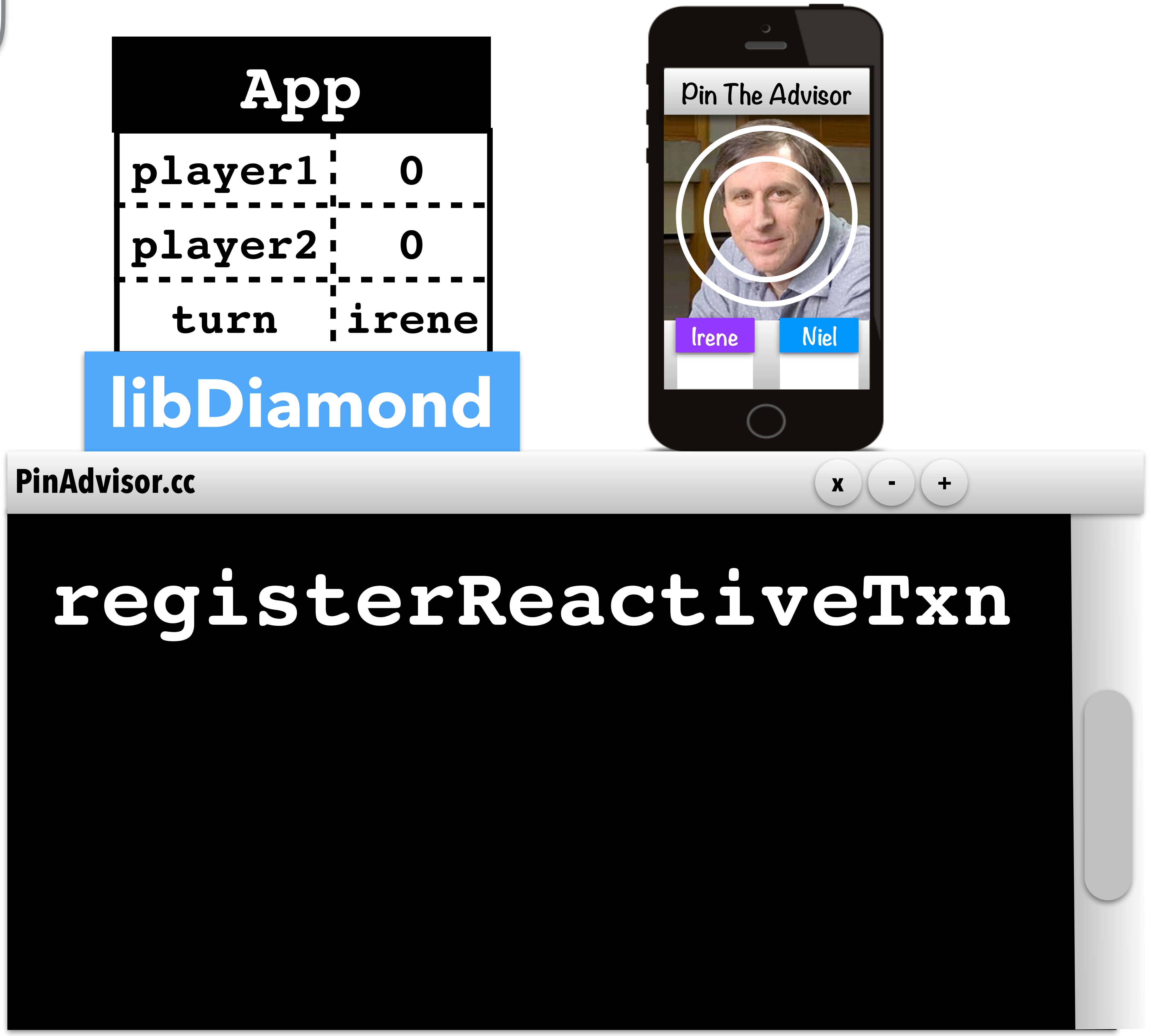
- Key abstraction for automatically propagating updates to local data
- Gives apps a consistent view of shared data and control over what to sync
- Automatically triggers app code in response to updates from read-write transactions to shared RDTs



Reactive Transactions

Read-only transactions that re-execute app code when the read set updates.

- Key abstraction for automatically propagating updates to local data
- Gives apps a consistent view of shared data and control over what to sync
- Automatically triggers app code in response to updates from read-write transactions to shared RDTs

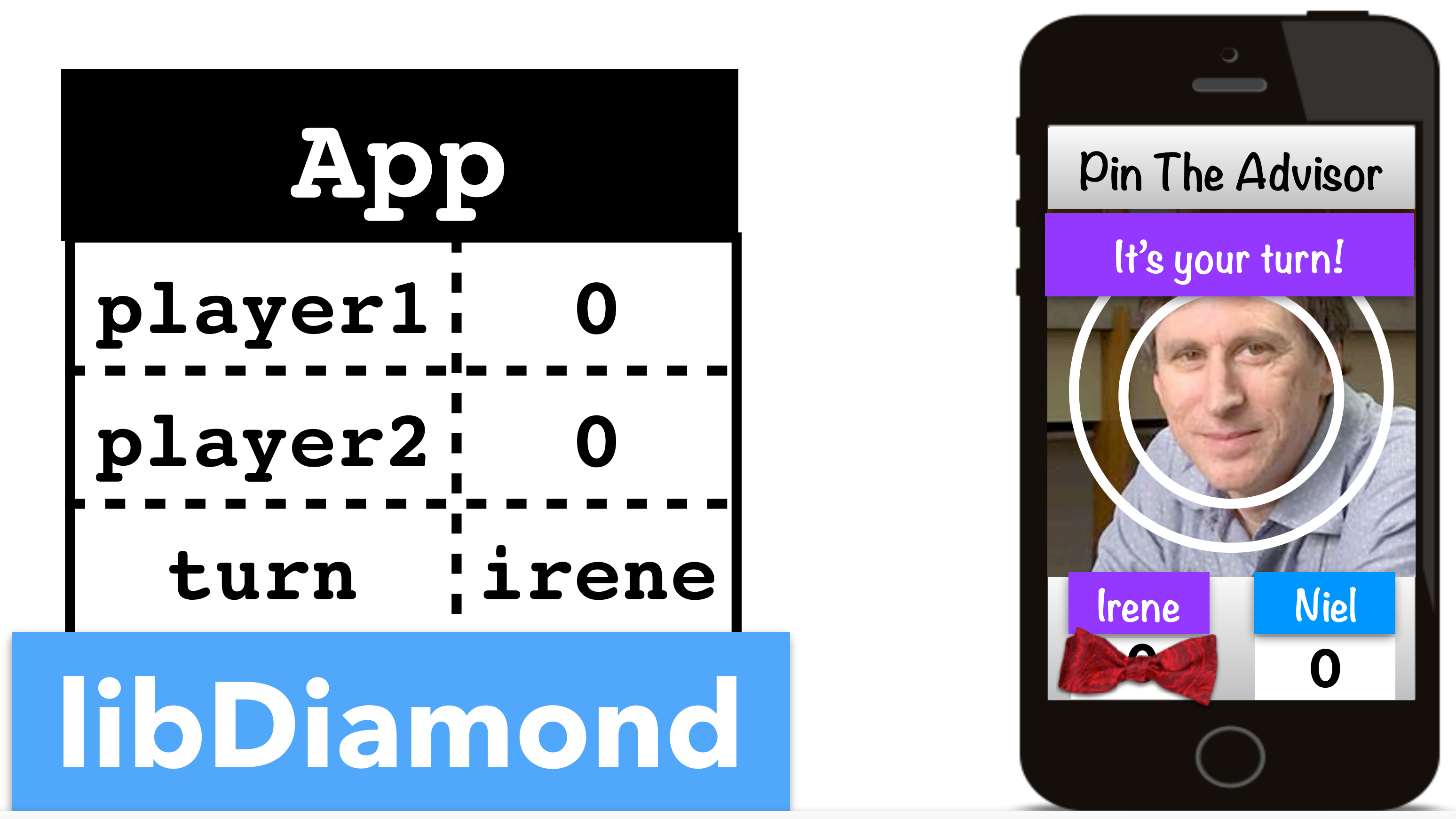


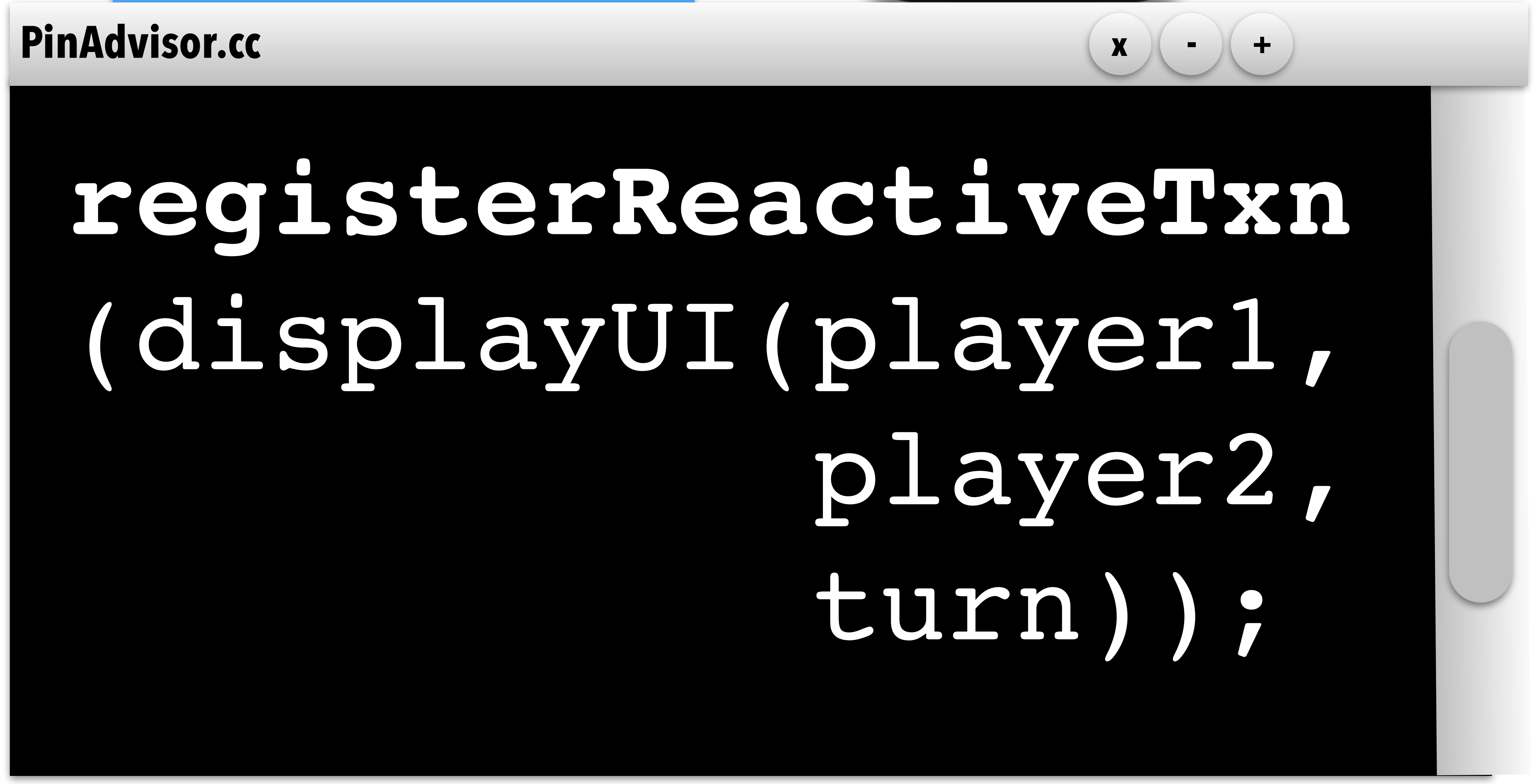
```
registerReactiveTxn
```

Reactive Transactions

Read-only transactions that re-execute app code when the read set updates.

- Key abstraction for automatically propagating updates to local data
- Gives apps a consistent view of shared data and control over what to sync
- Automatically triggers app code in response to updates from read-write transactions to shared RDTs





```
registerReactiveTxn
(displayUI(player1,
           player2,
           turn));
```

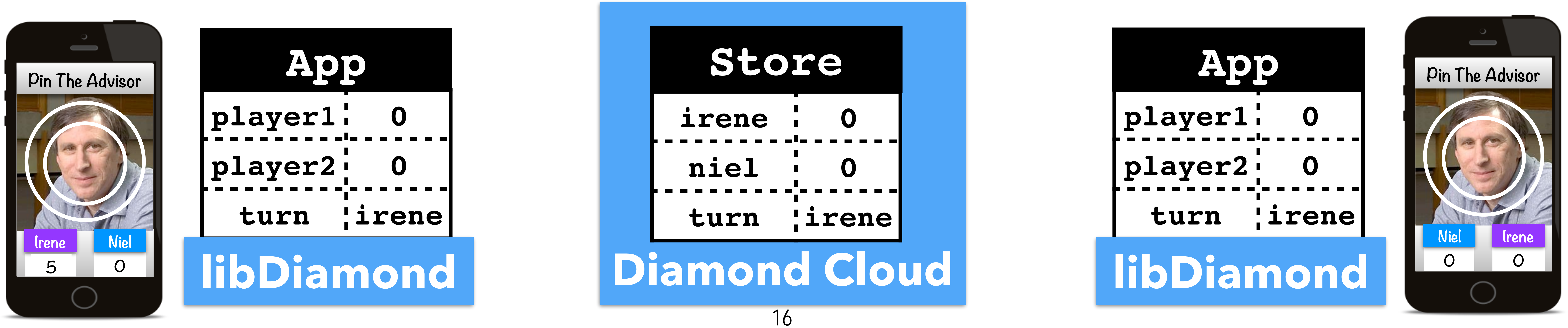
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



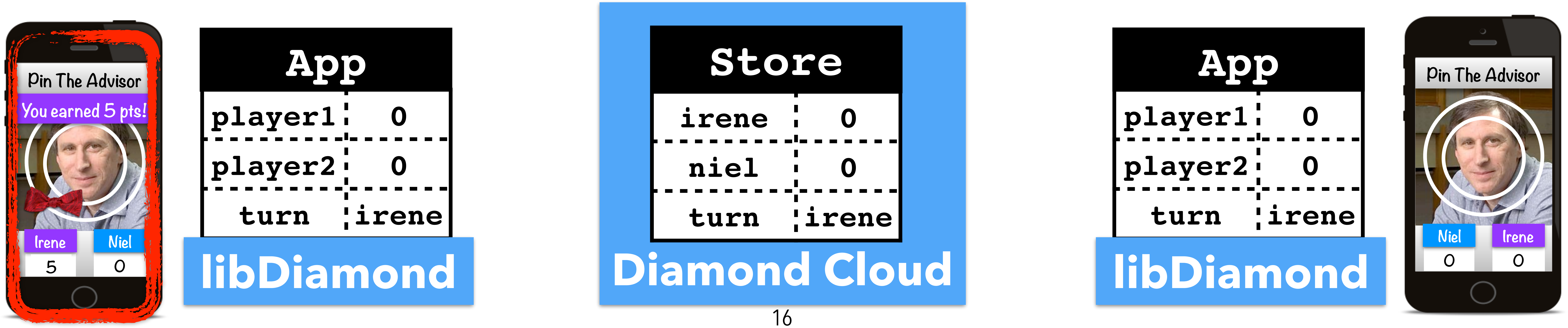
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



App	
player1	0
player2	0
turn	irene

Store	
irene	0
niel	0
turn	irene

App	
player1	0
player2	0
turn	irene

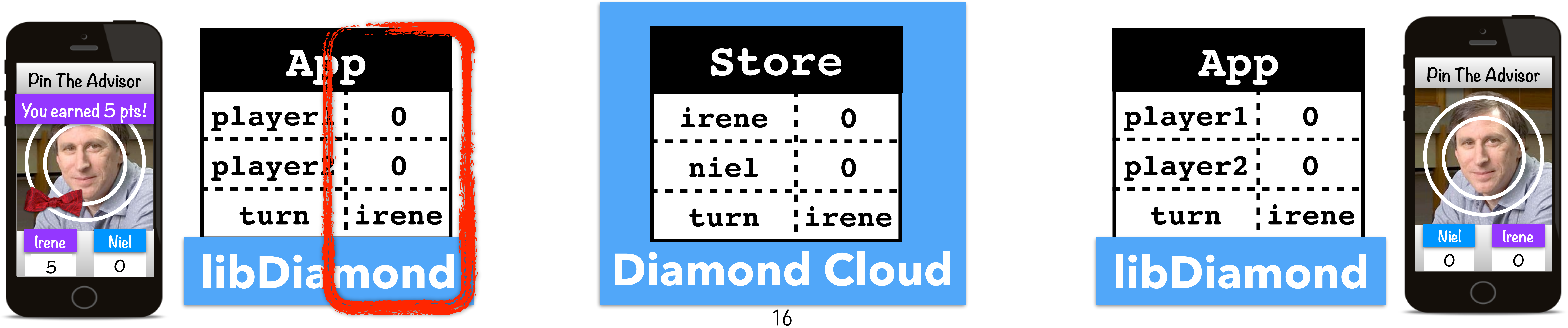
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



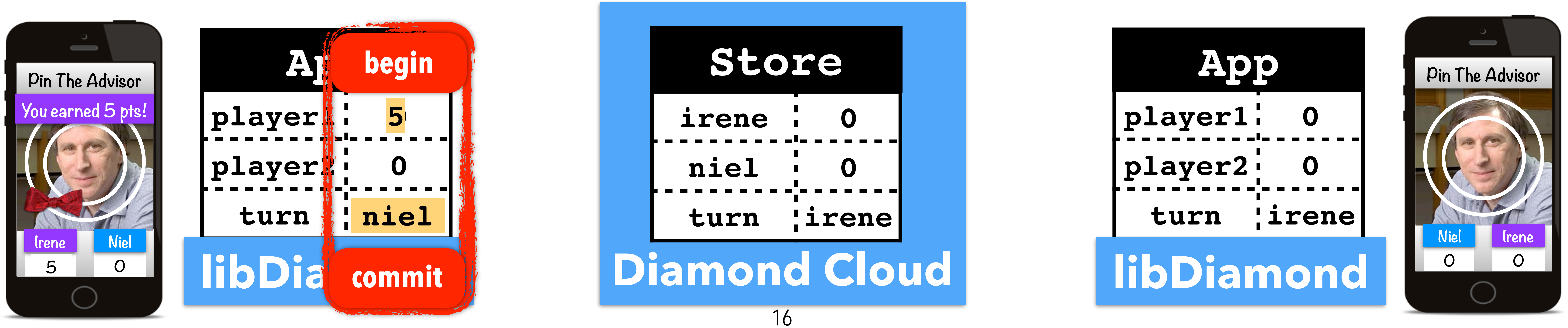
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



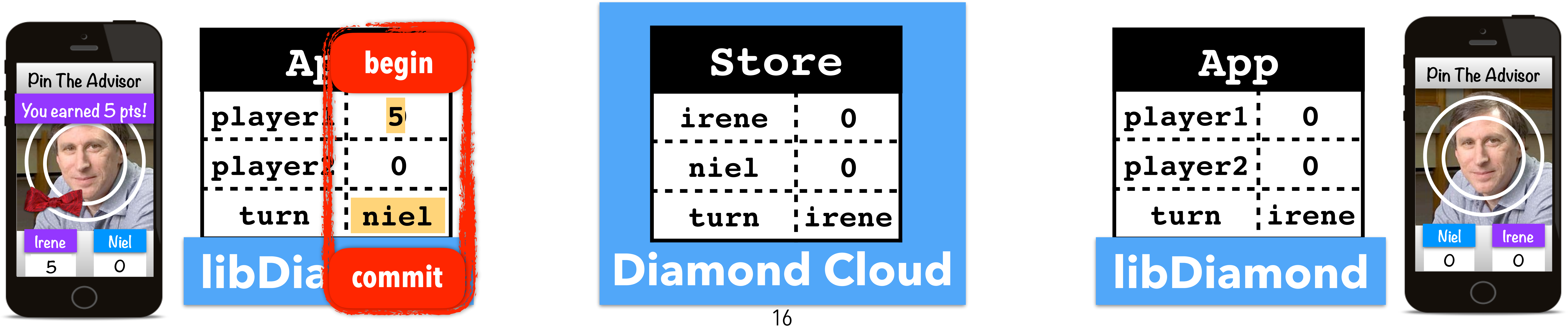
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



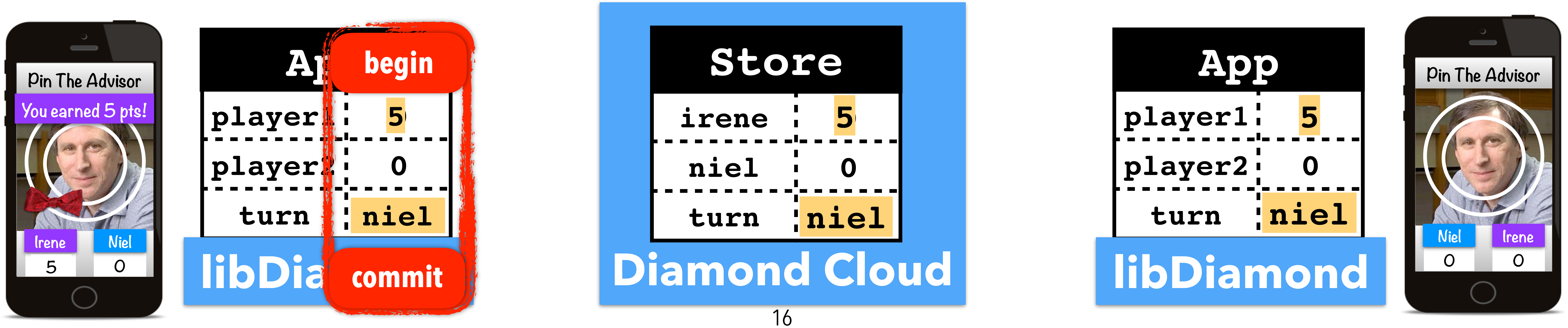
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



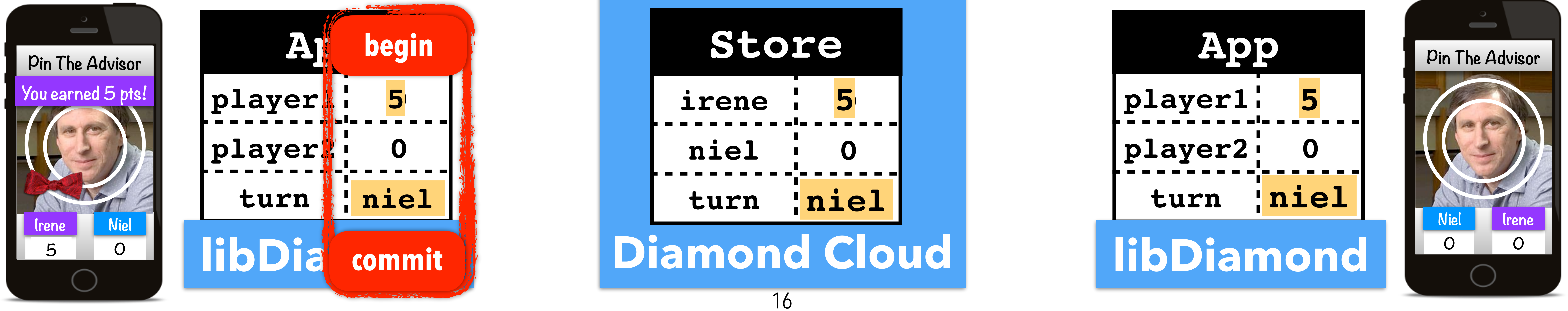
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



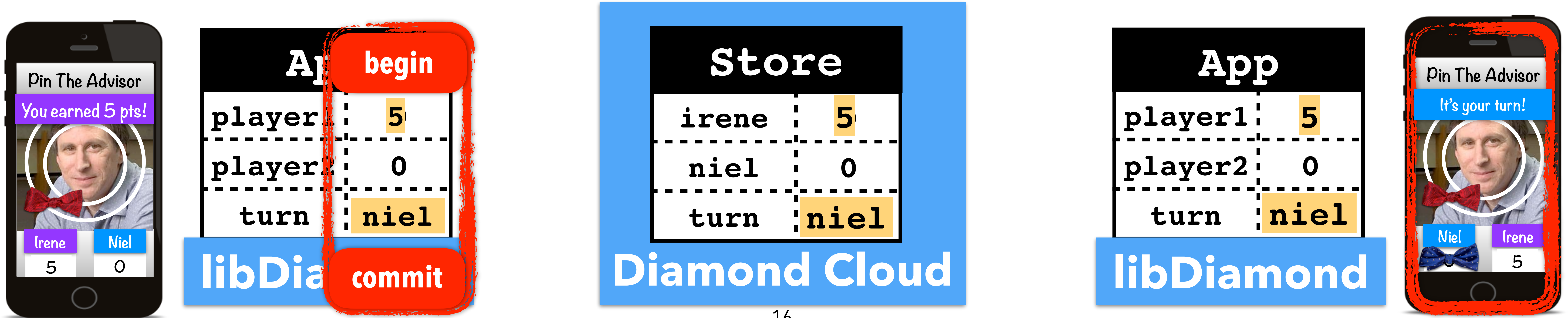
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



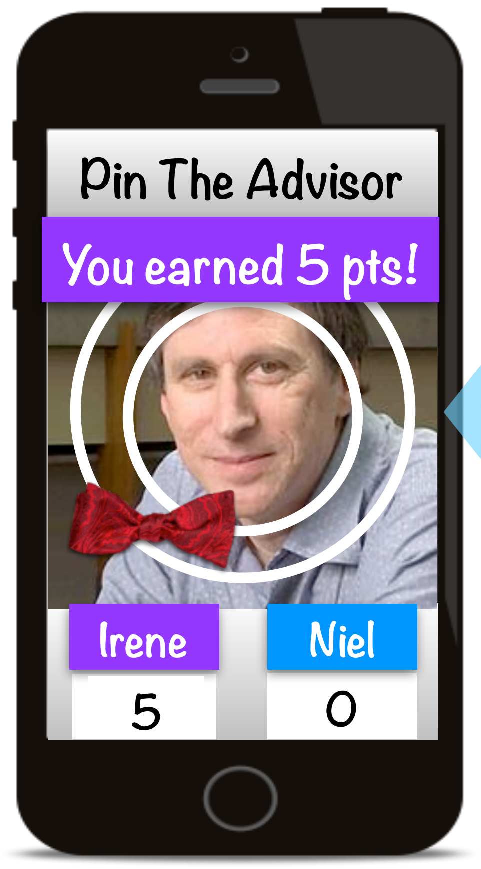
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.



Automated end-to-end data management and storage with fault-tolerance, availability and consistency

Talk Outline

Diamond System & Programming Model

What does Diamond provide for reactive apps?

Automated end-to-end data management and storage.

Diamond Guarantees & Implementation

What does Diamond guarantee for reactive apps?

Evaluation

How does Diamond impact app complexity and performance?

Diamond ACID+R Guarantees

Atomicity - All or no updates to shared data in a read-write transaction complete.

Consistency - All accesses in a transaction (read-write or reactive) reflect a single, point-in-time view of shared data.

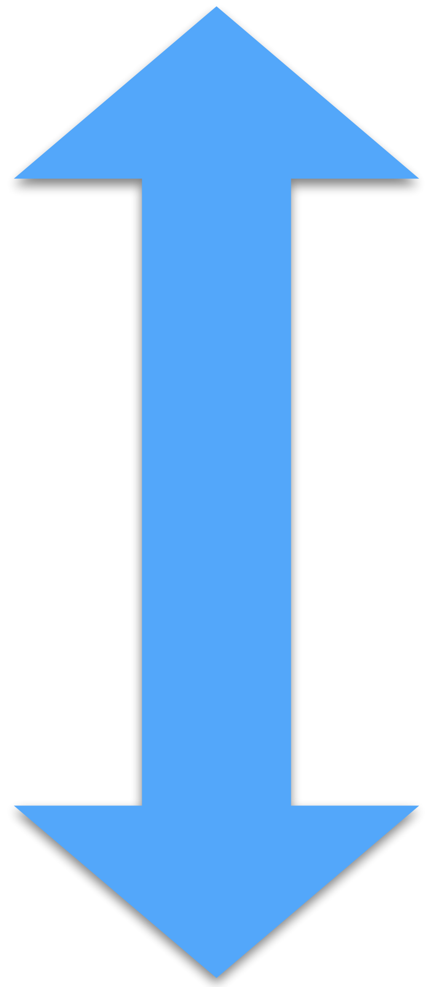
Isolation - All transactions reflect a serial execution order over shared data.

Durability - All updates in committed transactions are never lost.

Reactivity - All accesses in reactive transactions will eventually reflect the latest updates.

Diamond Isolation Levels

Stronger Guarantees

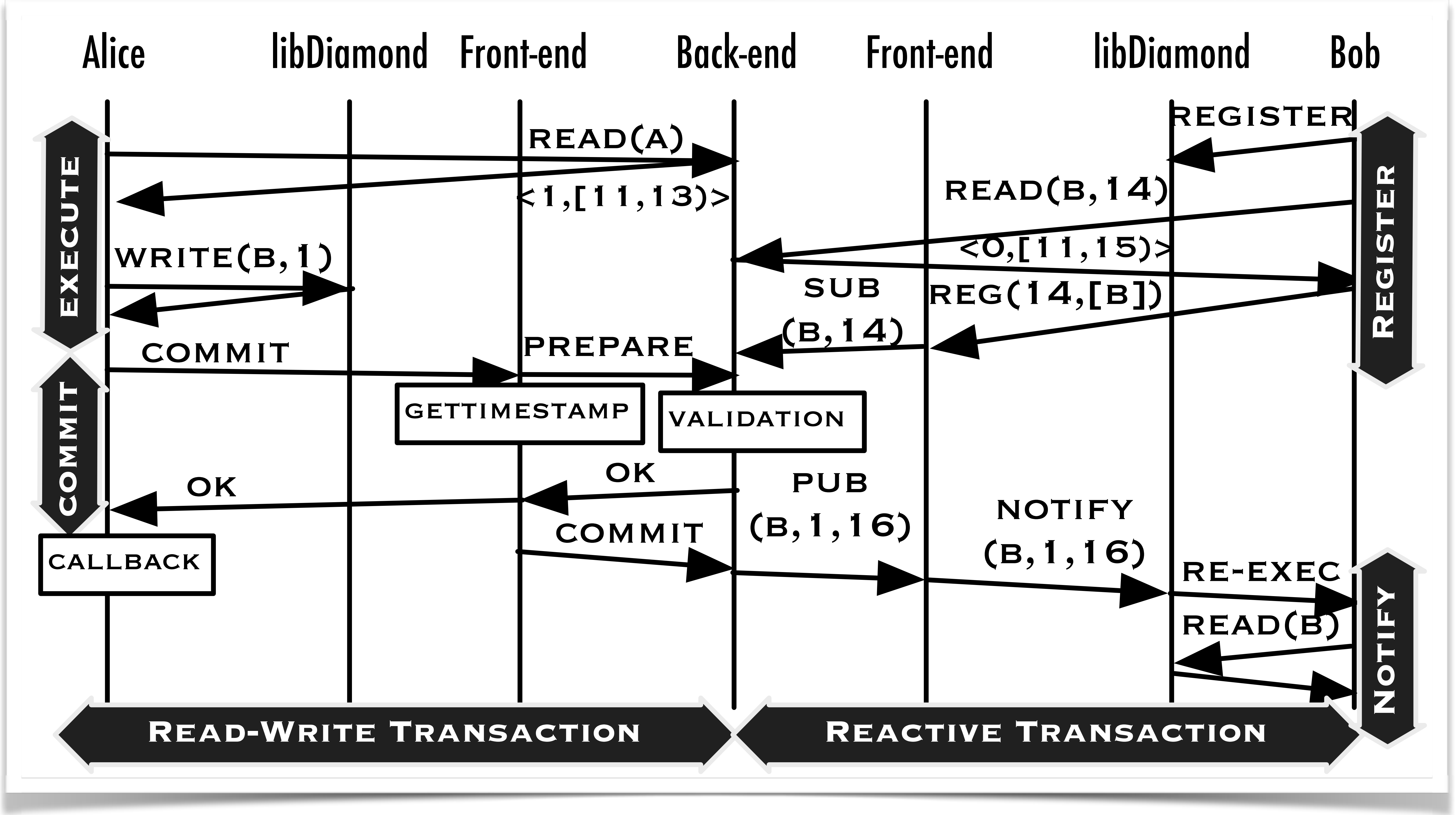


Better Performance

Read-write Isolation Level	Reactive Isolation Level
Strict Serializability	Serializable Snapshot
Snapshot Isolation	Serializable Snapshot
Read Committed	Read Committed

Diamond Implementation

Diamond Transaction Protocol



Wide-area Optimizations

- Data-type Optimistic Concurrency Control
- Multi-versioned caching
- Data Push Notifications

Take a look at the paper!

Talk Outline

Diamond System & Programming Model

What does Diamond provide for reactive apps?

Automated end-to-end data management and storage.

Diamond Guarantees & Implementation

What does Diamond guarantee for reactive apps?

Strong ACID+R transactional guarantees

Evaluation

How does Diamond impact app complexity and performance?

Evaluation Overview

- Does Diamond simplify reactive applications?
- How does Diamond perform compare to a hand coded implementation?
- Testbed: Google Compute Engine VMs (5 shards x 3 replicas)
- Workload: Retwis-based Twitter benchmark

Diamond reduces the complexity and improves the guarantees of reactive apps.

Application	Original LoC	Diamond LoC	% Saved
Multi-player Game	46	34	26%
Chat Room	335	225	33%
Scrabble clone	8729	7603	13%
Twitter clone	14,278	12,554	13%

Diamond reduces the complexity and improves the guarantees of reactive apps.

No UI. Mostly sync code.

Application	Original LoC	Diamond LoC	% Saved
Multi-player Game	46	34	26%
Chat Room	335	225	33%
Scrabble clone	8729	7603	13%
Twitter clone	14,278	12,554	13%

Diamond reduces the complexity and improves the guarantees of reactive apps.

Application	Original LoC	Diamond LoC	% Saved
Multi-player Game	46	34	26%
Chat Room	335	225	33%
Scrabble clone	8729	7603	13%
Twitter clone	14,278	12,554	13%

No UI. Mostly sync code.

Full UI. Complex app logic.

Diamond reduces the complexity and improves the guarantees of reactive apps.

Application	Original LoC	Diamond LoC	% Saved
Multi-player Game	46	34	26%
Chat Room	335	225	33%
Scrabble clone	8729	7603	13%
Twitter clone	14,278	12,554	13%

No UI. Mostly sync code.

Full UI. Complex app logic.

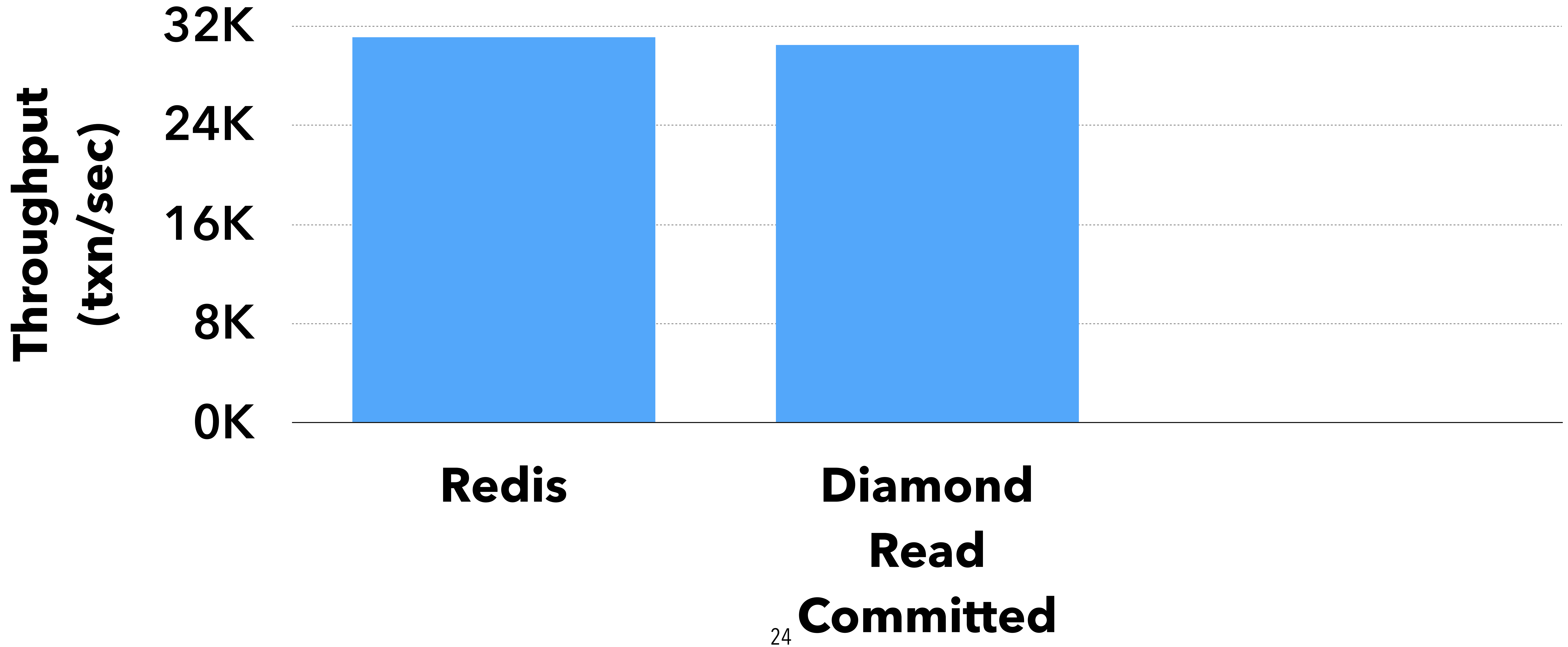
+durability
+reactivity

+durability

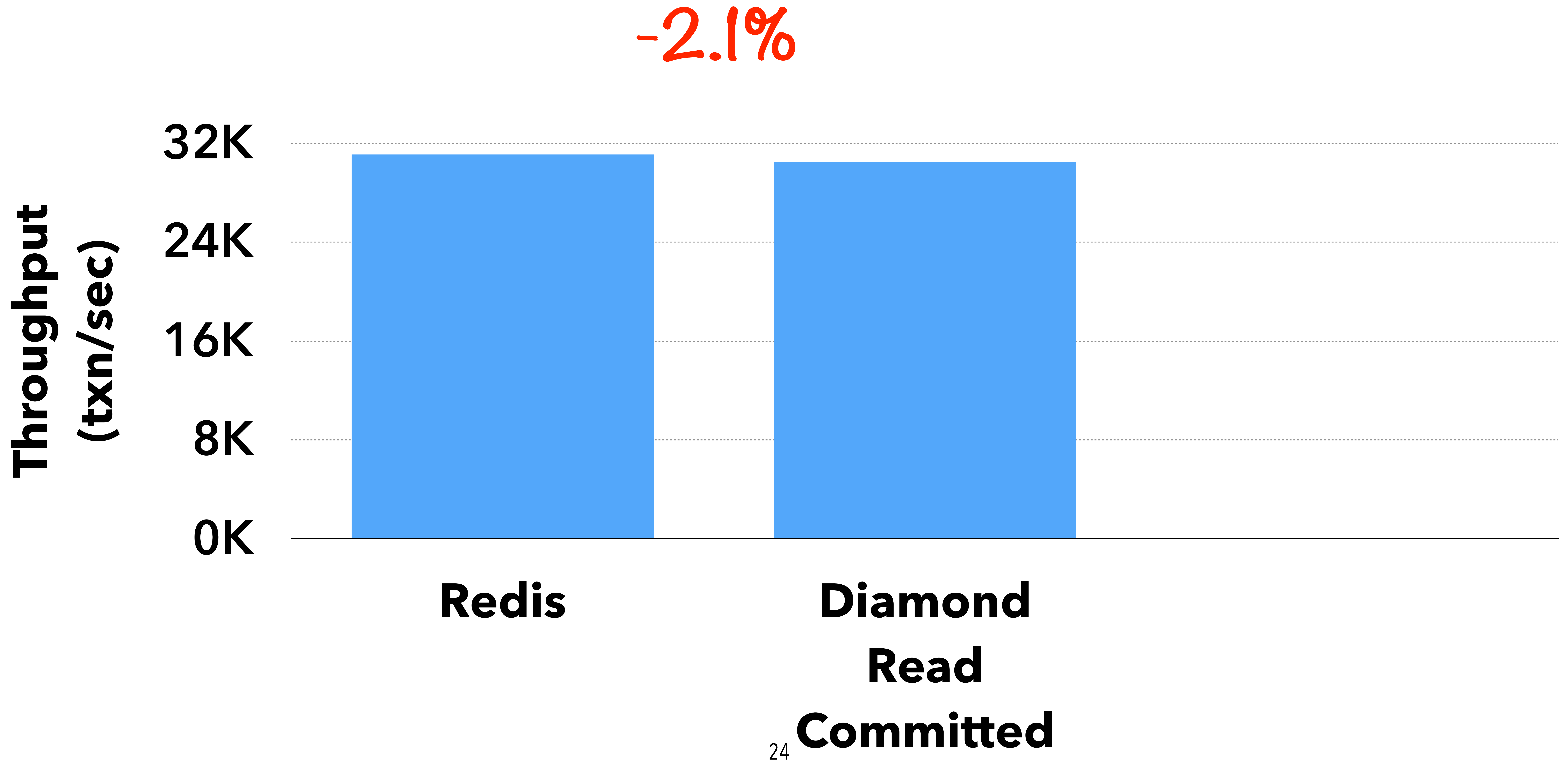
+consistency
+isolation
+reactivity

+durability

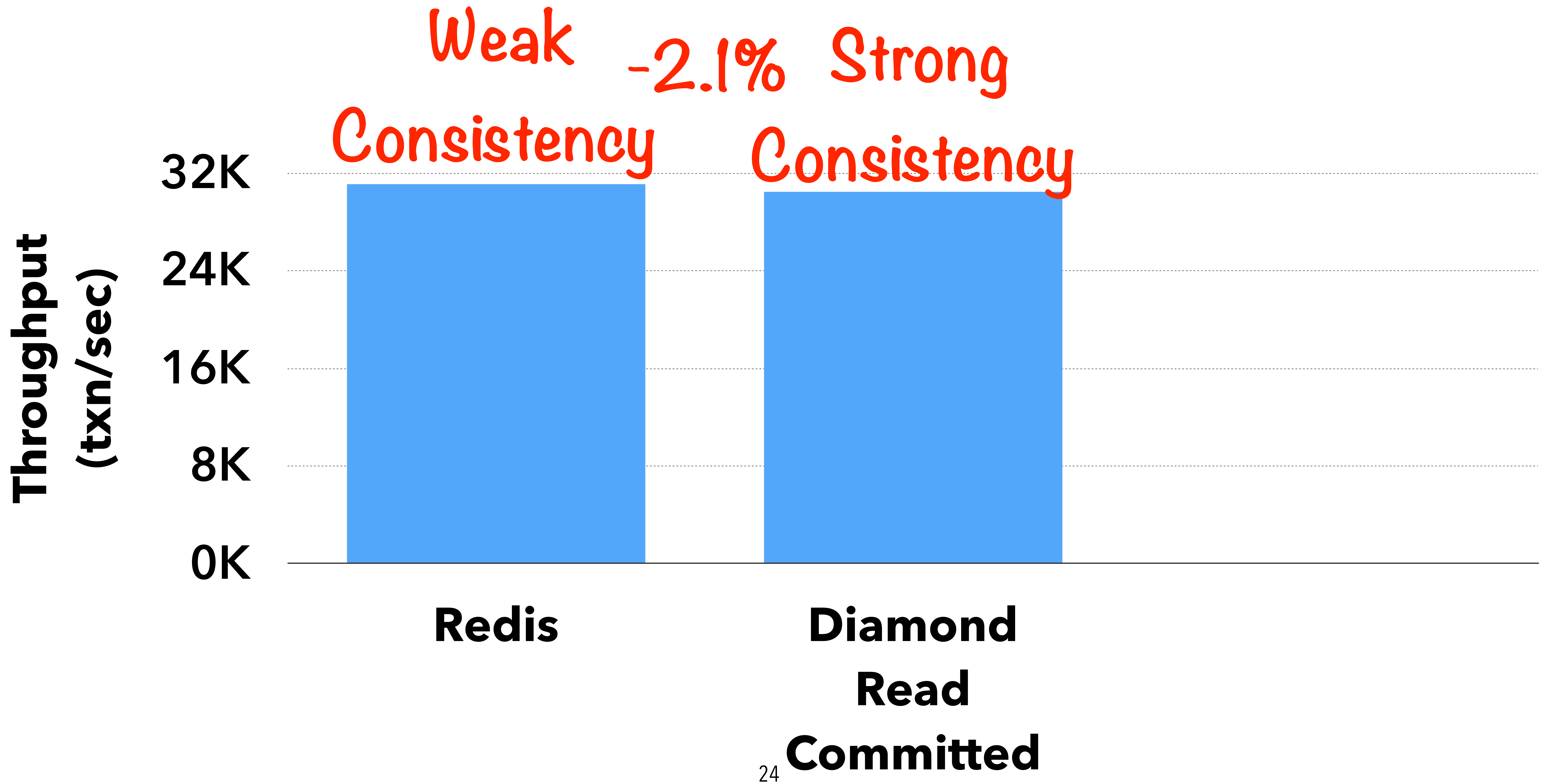
Diamond’s data management has low overhead.



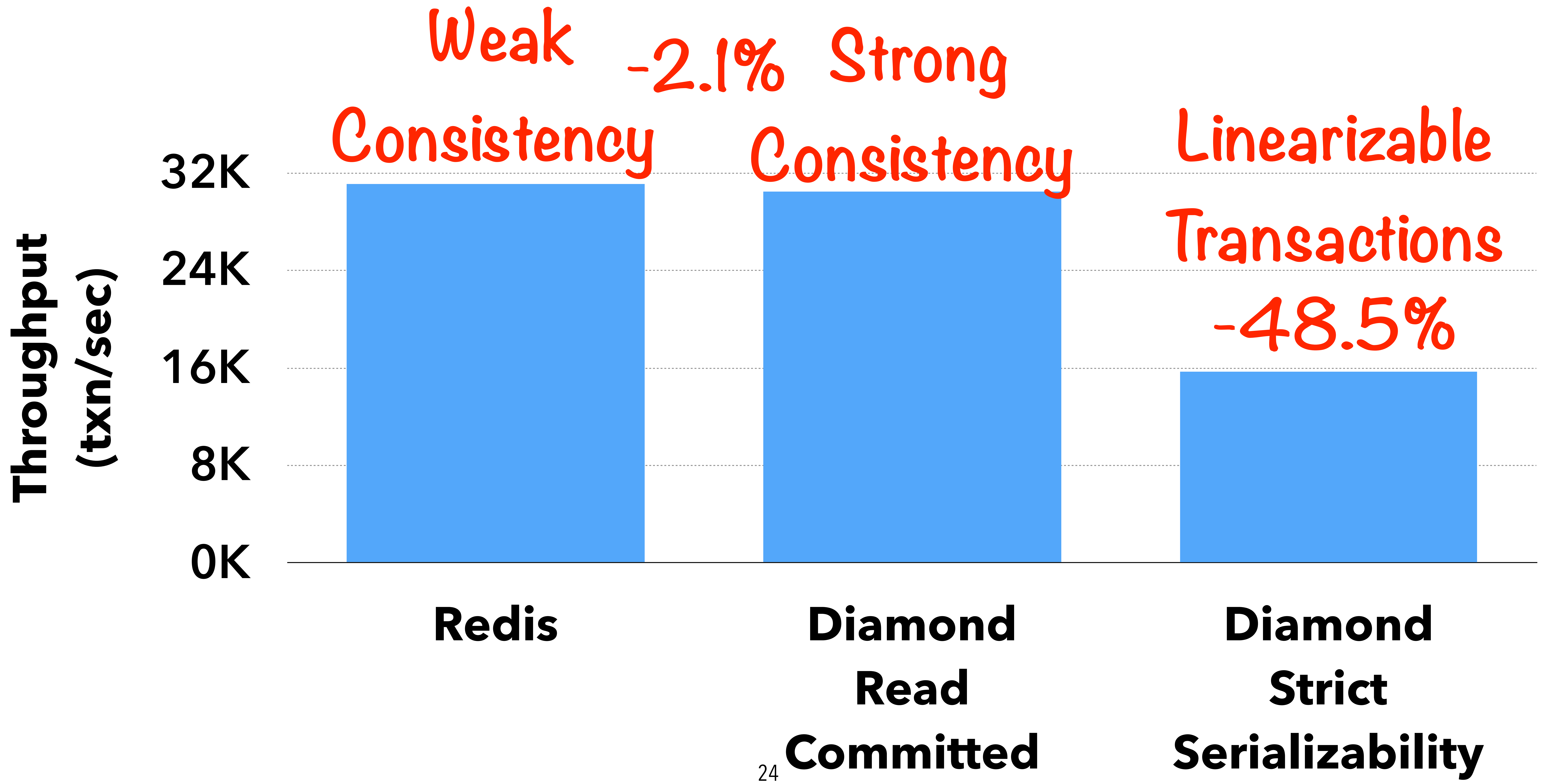
Diamond’s data management has low overhead.



Diamond’s data management has low overhead.



Diamond’s data management has low overhead.



Summary

What does Diamond provide for reactive apps?

Automated end-to-end data management and storage.

What does Diamond guarantee for reactive apps?

Strong ACID+R transactional guarantees.

How does Diamond impact app complexity and performance?

Simplifies reactive apps with low overhead.

https://github.com/UWSysLab/diamond

Related Work

- Distributed Programming Frameworks
  Meteor, Parse, Firebase, Mjolnir, Mapjax, RethinkDB
- Client-side Programming Frameworks
  React, Angular, Blaze, ReactiveX
- Distributed Storage Systems
  Redis, MongoDB, Dropbox
- Notification/Pub-Sub/Streaming Services
  Thialfi, Apache Kafka, Amazon Kinesis