EVERYWHERE.

Redis Data Structures for Non-Redis people

Alvin Richards
Chief Education Officer
Redis Labs

redisconf18

#Redis #RedisConf

presented by:

redislabs
home of redis

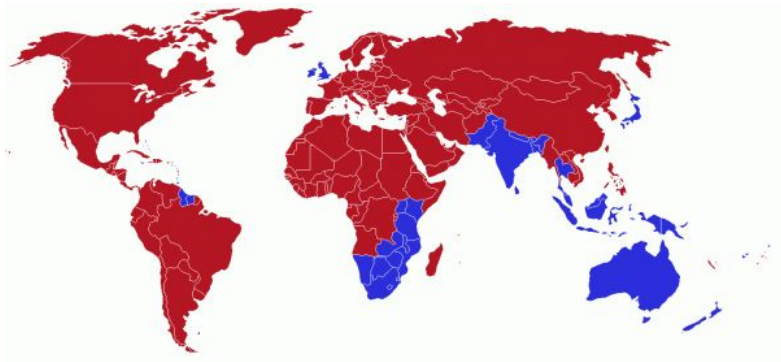
Today's ramble

- Data Structures in Redis
- Transactions in Redis
- How to think & approach Redis
- Conclusions

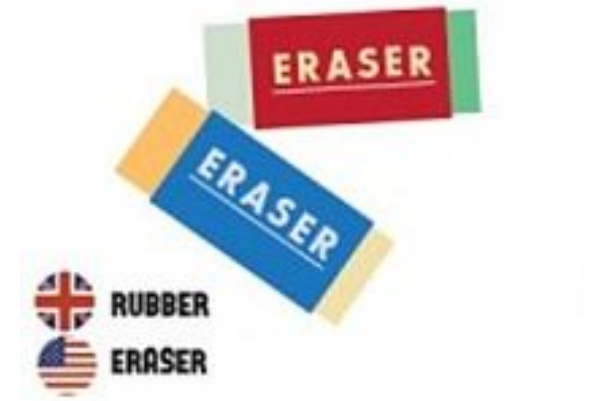
Everything is Different... in Britain

I really ate these things

I drove on this side of the road



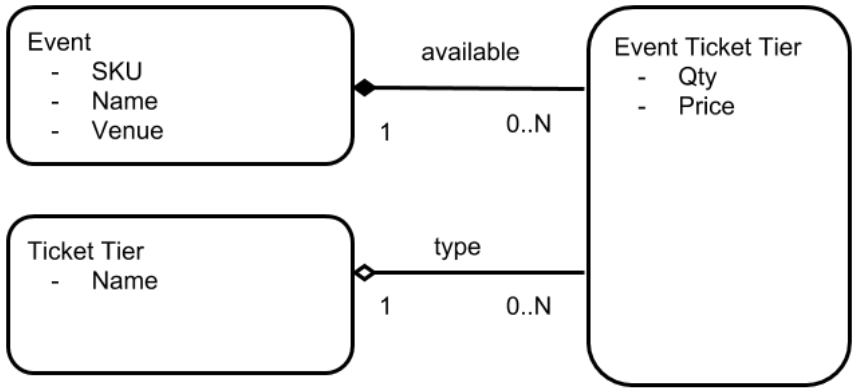
I learnt to "translate"



And you probably grew up with these...

Entity-Relation Diagrams

Tables/Foreign keys



XML

```
<?xml version="1.0"?>
<event_catalog>
  <event>
    <sku="123-ABC-723">
    <name>Men's 100m Final</name>
<disabled_access>true</disabled_access>
    <medal_event>true</medal_event>
    <venue>Olympic Stadium</event>
    <category>Track & Field</category>
    <capacity>60102</capacity>
    <available>
      <general>
        <qty>20000</qty>
        <price>25.00</price>
      </general>
   </available>
  </event>
</event_catalog>
```

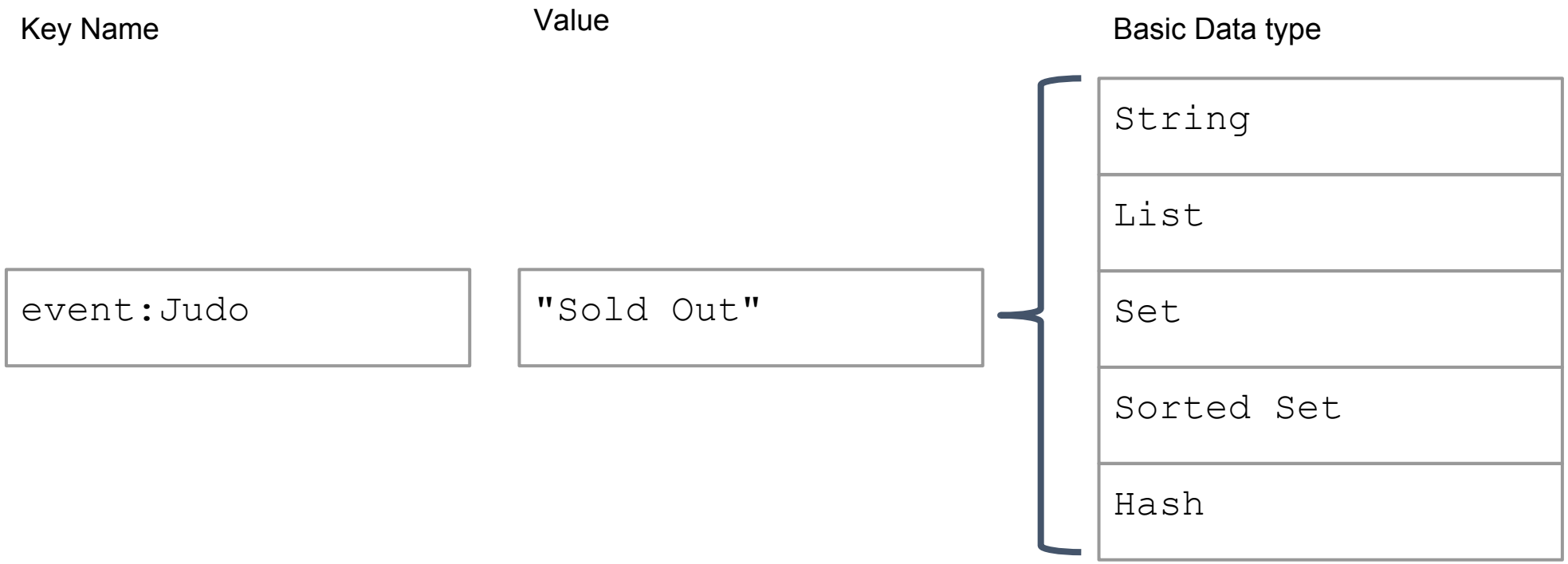
JSON

```
{
  'sku': "123-ABC-723",
  'name': "Men's 100m Final",
  'disabled_access': True,
  'medal_event': True,
  'venue': "Olympic Stadium",
  'category': "Track & Field",
  'capacity': 60102,
  'available':
    { 'general': { 'qty': 20000,
                   'price': 25.00
                 }
    }
}
```

Redis is different… and here's how

- Redis is a Multi Model Key store
- Commands operate on Keys
  - Commands work on specific data types
  - Commands work on specific encoding of values
- Data types of Keys can change overtime

Keys, Values & Data types



Keys

Key Name

```
event:Judo
```

Keys

- Unique
- Binary Safe
- Up to 512MB in size

Key Naming

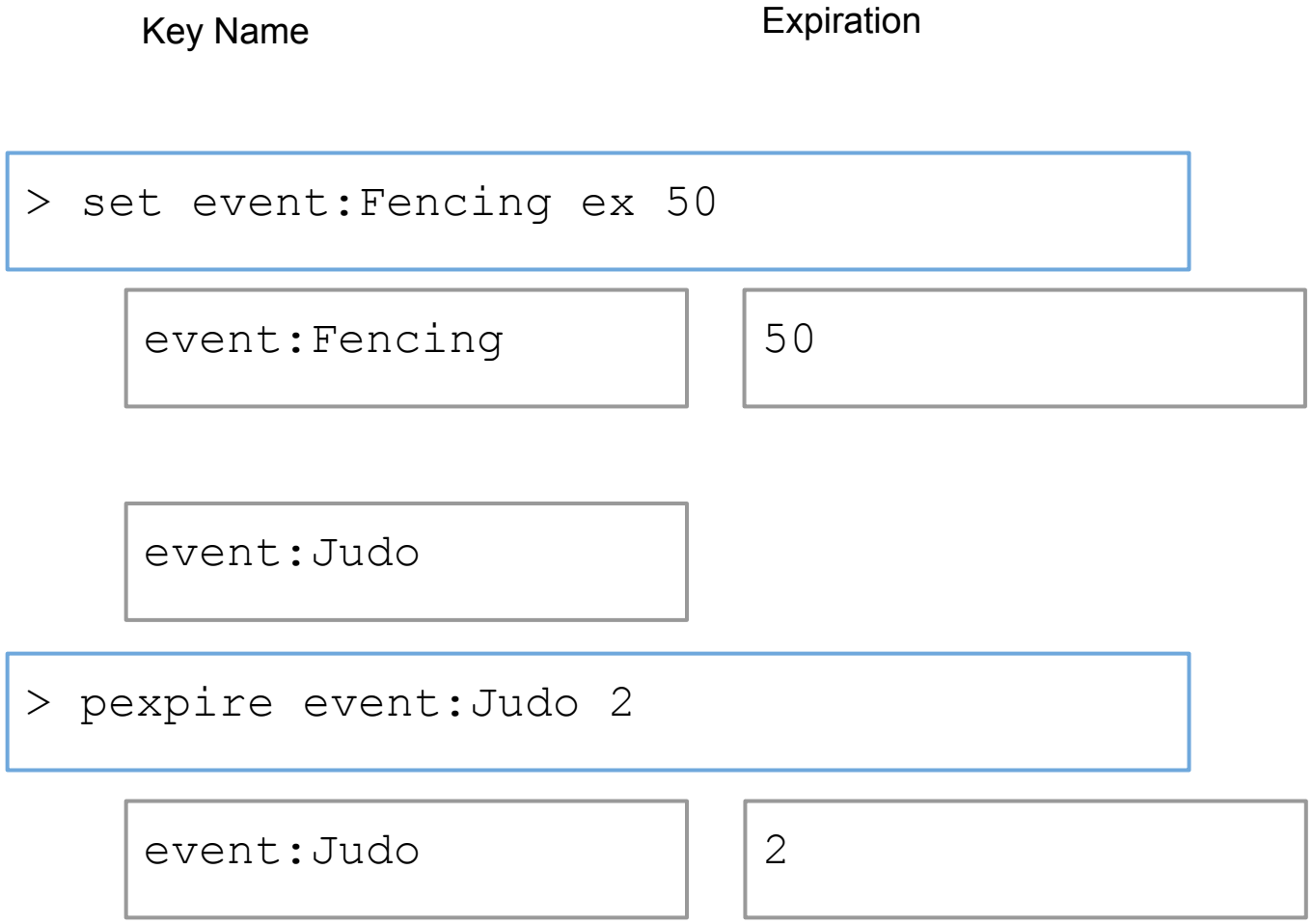
Key Name
`event:Judo`
`event:venues`

`event:Fencing`
`event:fencing`
`Event:fencing`

Naming

- Up to you
- Be consistent
- Colon is a typical community convention
  - Domain Object Name
  - Colon
  - Unique ID
- Plurals are nice for Lists, Sets
- Case-sensitive

Key Expiration



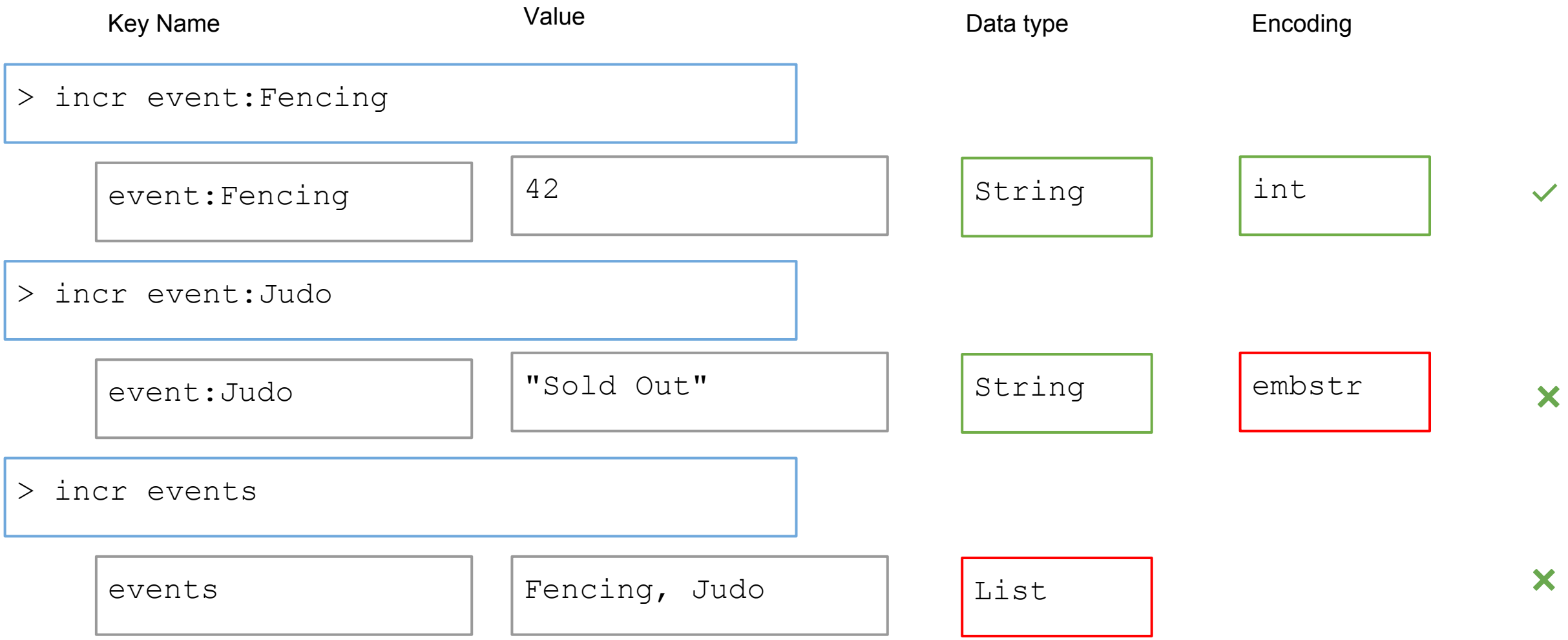
Expiration or TTL

- Default - keys are retained
- Specified in
  - Seconds
  - Milliseconds
  - Unix Epoch
- Added to Key
- Removed from Key

String Datatype & Encoding

Key Name	Value	Encoding
`> set event:Judo "Sold Out"`		
`event:Judo`	`"Sold Out"`	`embstr`
`> set event:Judo 42`		
`event:Judo`	`"42"`	`int`
`> set event:Judo 3.14159`		
`event:Judo`	`"3.14159"`	`embstr`

Command Execution, Data types and Encoding



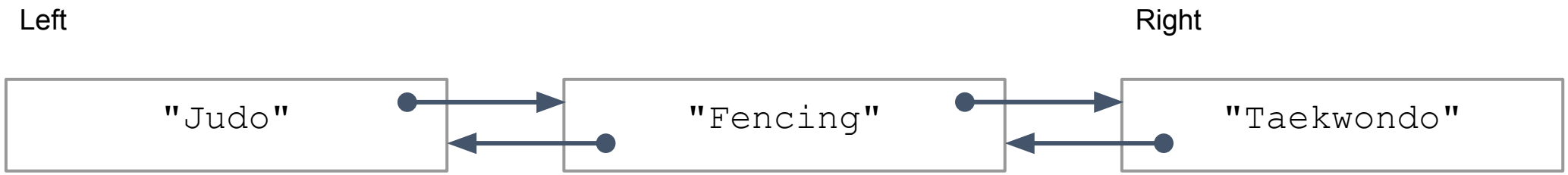
Lists

Key Name

```
events
```

Lists

- Ordered collection of Strings
- Duplicates allowed
- Elements added
  - Left, Right, by position




```
> lpop events
"Judo"
```

```
> rpop events
"Taekwondo"
```

Why are Redis Lists cool?

Useful to create

- Queues, Stacks
- Capped Lists

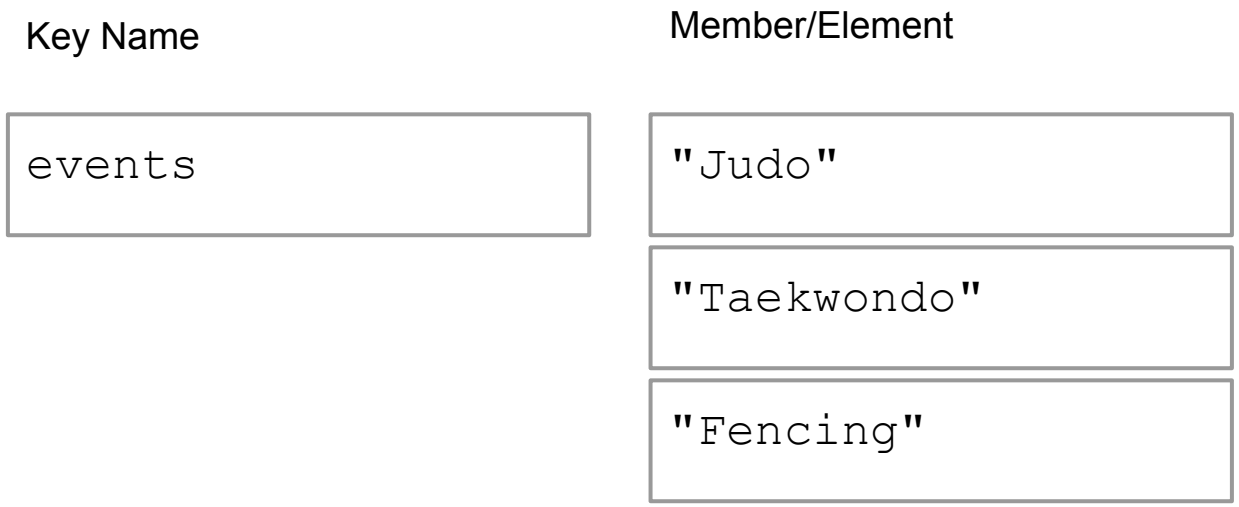
Typical use cases

- Interprocess communications

Many operations are O(1)

- Same time complexity regardless of List size
- LPOP, LPUSH, LPUSHX
- RPOP, RPUSH, RUSHPX
- BLPOP, BRPOP
- RPOPLPUSH, BRPOPLPUSH
- LLEN

Sets




```
> smembers events
1)  "Judo"
2)  "Taekwondo"
3)  "Fencing"
```

Sets

- Unordered collection of unique Strings
- Set operations
  - Difference
  - Intersect
  - Union

Why are Redis Sets cool?

Set Operations

- Difference, Intersect, Union

Typical use cases

- Tag Cloud
- Unique Visitors

Many operations are O(1)

- Same time complexity regardless of Set size
- SADD
- SCARD
- SPOP, SRANDMEMBER

Sorted Sets

Key Name	Score	Value
events	2	"Judo"
	3	"Taekwondo"
	1	"Fencing"

```
> zrange events 0 1
1)  "Fencing"
2)  "Judo"
```

Sets

- Ordered collection of unique Strings
- Manipulation by value, score, position or lexicography
- Set operations
  - Intersect
  - Union

Why are Redis Sorted Sets cool?

Set Operations

- Intersect & Union

Typical use cases

- Leaderboard
- Priority Queue

Many operations are O(1) or O(log N)

- ZADD,
- ZCOUNT, ZINCRBY
- ZSCORE, ZRANK

Hashes

Key Name	Score	Value
event:Judo	venue	Super Dome
	capacity	32000
	subway	True

```
> hget event:Judo capacity
"32000"
```

Hashes

- Field & Value pairs
- Single level
- Dynamically add remove fields

Why are Redis Hashes cool?

Set Operations

- Intersect & Union

Typical use cases

- Rate Limiting
- Session Cache

Many operations are O(1)

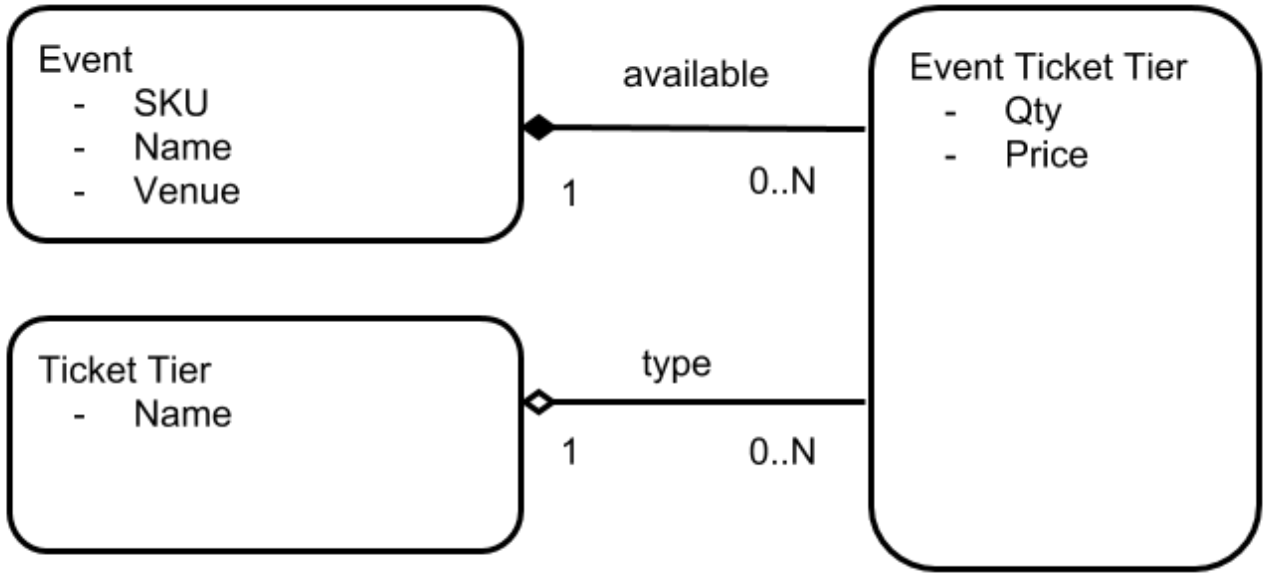
- HGET, HSET, HSETNX
- HINCRBY, HINCRBYFLOAT
- HLEN, HSTRLEN
- HEXISTS

Like… Wow… dude

Putting that into practice

Modeling an Event & Ticket Tiers

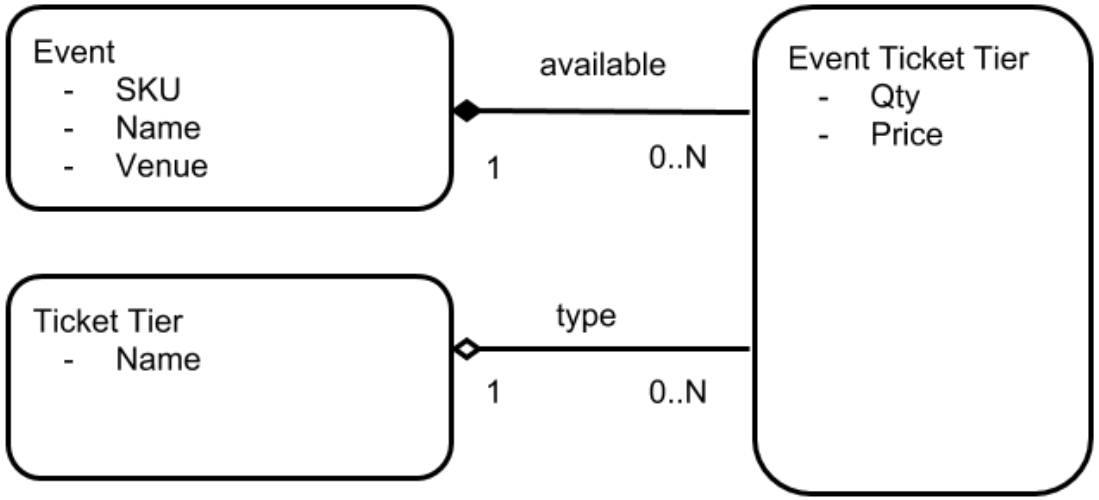
```
{
    'sku': "123-ABC-723",
    'name': "Men's 100m Final",
    'disabled_access': True,
    'medal_event': True,
    'venue': "Olympic Stadium",
    'category': "Track & Field",
    'capacity': 60102,
    'available':
      { 'general': { 'qty': 20000,
                     'price': 25.00
                   }
      }
}
```



Embedding

Relationships

Associations, Cardinality & Containment



- Association & Cardinality
  - An Event Ticket Tier is associated with One and Only One Event
  - An Event may have zero or more Event Ticket Tiers
  - An Event Ticket Tier is of One and Only One Ticket Tier
  - A Ticket Tier may have zero or more Event Ticket Tiers
- Containment
  - When the Event is removed, Event Ticket Tiers are also removed

Solution #1: Strings
The most common (and worst)

```
> set event:event:123-ABC-723

"{\"sku\": \"123-ABC-723\",
\"category\": \"Track & Field\",
\"available\": {\"general\":
{\"price\": 25.0, \"qty\": 20000}},
\"capacity\": 60102, \"name\":
\"Men's 100m Final\",
\"disabled_access\": true,
\"medal_event\": true, \"venue\":
\"Olympic Stadium\"}"
```

- Good
  - Simple to get and set the whole object
  - Many frameworks do this by default
- Bad
  - Many frameworks do this by default
  - Resource usage
    - CPU, Memory, Network
    - Serialization / Deserialization
  - Whole object must be Get/Set
    - e.g. Cannot just increment a single field
  - Painful to deal with

Solution #2: Hash
Typical recommendation

```
> hmset event:123-ABC-723

sku "123-ABC-723"
name "Men's 100m Final"
disabled_access True
medal_event True
venue "Olympic Stadium"
category "Track & Field"
capacity 60102
available:general:qty 20000
available:general:price 22.50
```

- Hashes are arbitrarily sized
- Field names up to 512MB
  - But really, should be short!
- Flatten hierarchy / Associations
  - Large number of price/qty pairs can be created
  - Ticket Tiers can be removed by deleting pairs of fields
- Containment
  - When the Key is removed, all fields are also removed

Solution #2: Hash

```
> hmset event:123-ABC-723

sku "123-ABC-723"
name "Men's 100m Final"
disabled_access True
medal_event True
venue "Olympic Stadium"
category "Track & Field"
capacity 60102
available:general:qty 20000
available:general:price 22.50
```

- Good
  - Atomic updates
  - Atomic deletes
- Bad
  - Large objects

Solution #3: Multiple Hashes plus Set Uncommon

```
> hmset event:123-ABC-723

sku "123-ABC-723"
name "Men's 100m Final"
disabled_access True
medal_event True
venue "Olympic Stadium"
category "Track & Field"
capacity 60102

> hset event:123-ABC-723:general

qty 20000
price 22.50

> sadd event:123-ABC-723:available

event:123-ABC-723:general
```

- Hash for each Object
  - Event
  - Tier
- (Optional) Set to hold Relationship
  - Redundant
  - Useful if you need to only detect presence of Relationship end

Solution #3: Multiple Hashes plus Set

```
> hmset event:123-ABC-723
sku "123-ABC-723"
name "Men's 100m Final"
disabled_access True
medal_event True
venue "Olympic Stadium"
category "Track & Field"
capacity 60102
> hset event:123-ABC-723:general
qty 20000
price 22.50
> sadd event:123-ABC-723:available
event:123-ABC-723:general
```

- Good
  - Extensible structures
  - Independently stored
- Bad
  - Relationship has to be manually maintained
  - All Keys need to be on same Shard

We're cool, right?

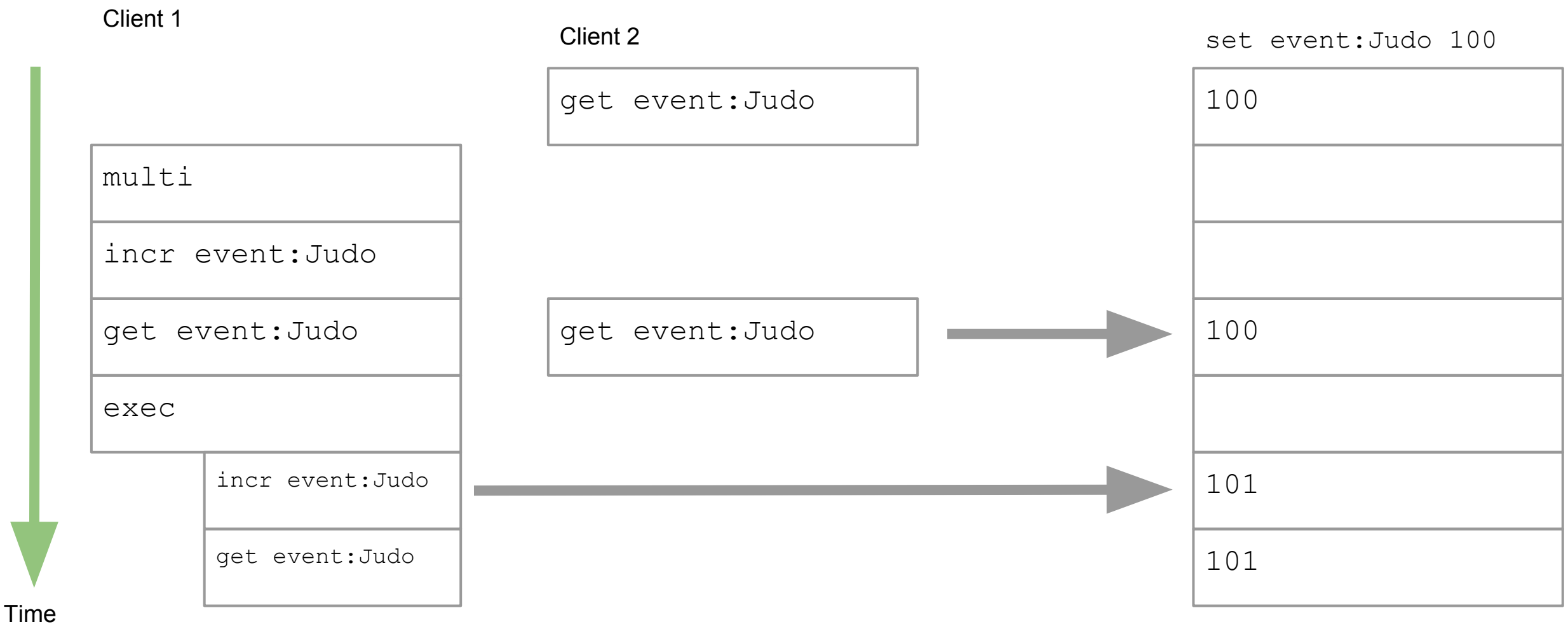
Transactions

- Redis has Transactions!
  - Atomic
  - Isolated
- Commands queued
- Queued commands executed sequentially as a single unit

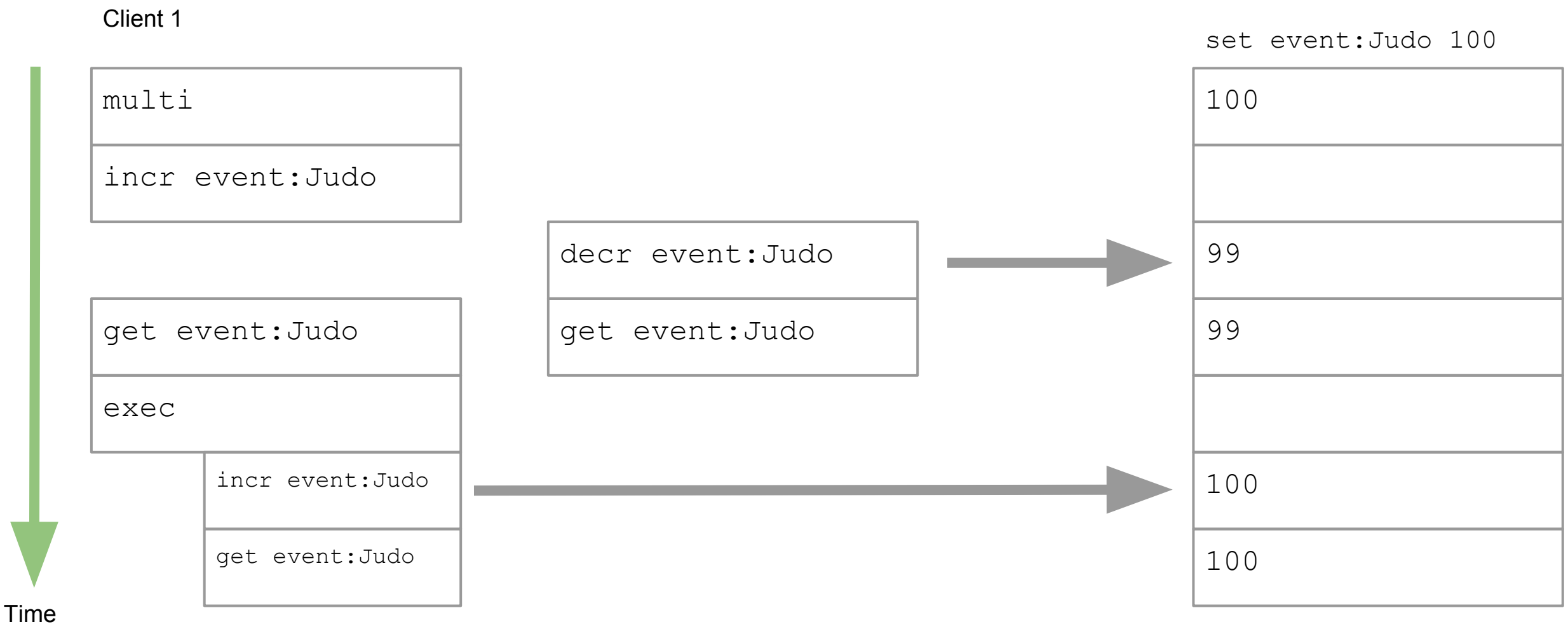
```
> multi
> set event:Judo 100
> incr event:Judo
> exec
```

```
> multi
> set event:Judo "Sold Out"
> discard
```

Transactions in Action #1



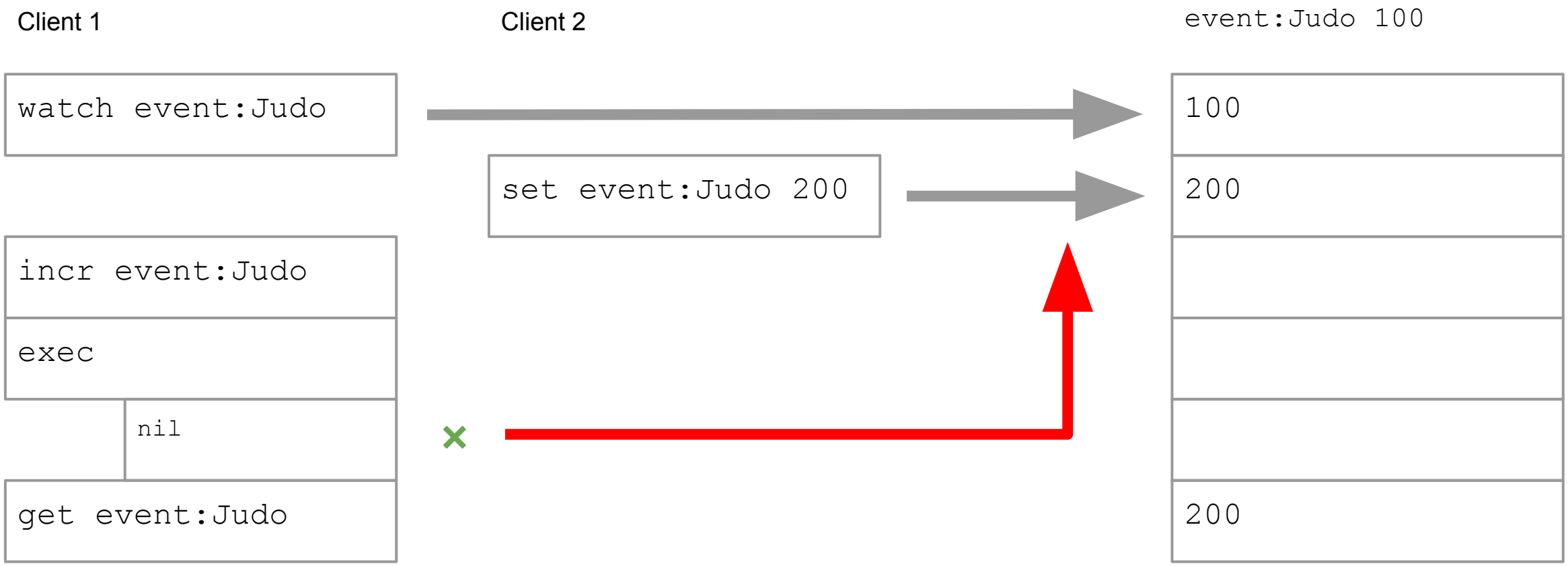
Transactions in Action #2



Optimistic Concurrency

- WATCH
- UNWATCH

Optimistic Concurrency in Action #1



Why are Transaction important?

- Allows data to be safely modeled in many Keys
- Watches allow for Optimistic Concurrency Control

Really, are cool now?

Geospatial
Add point

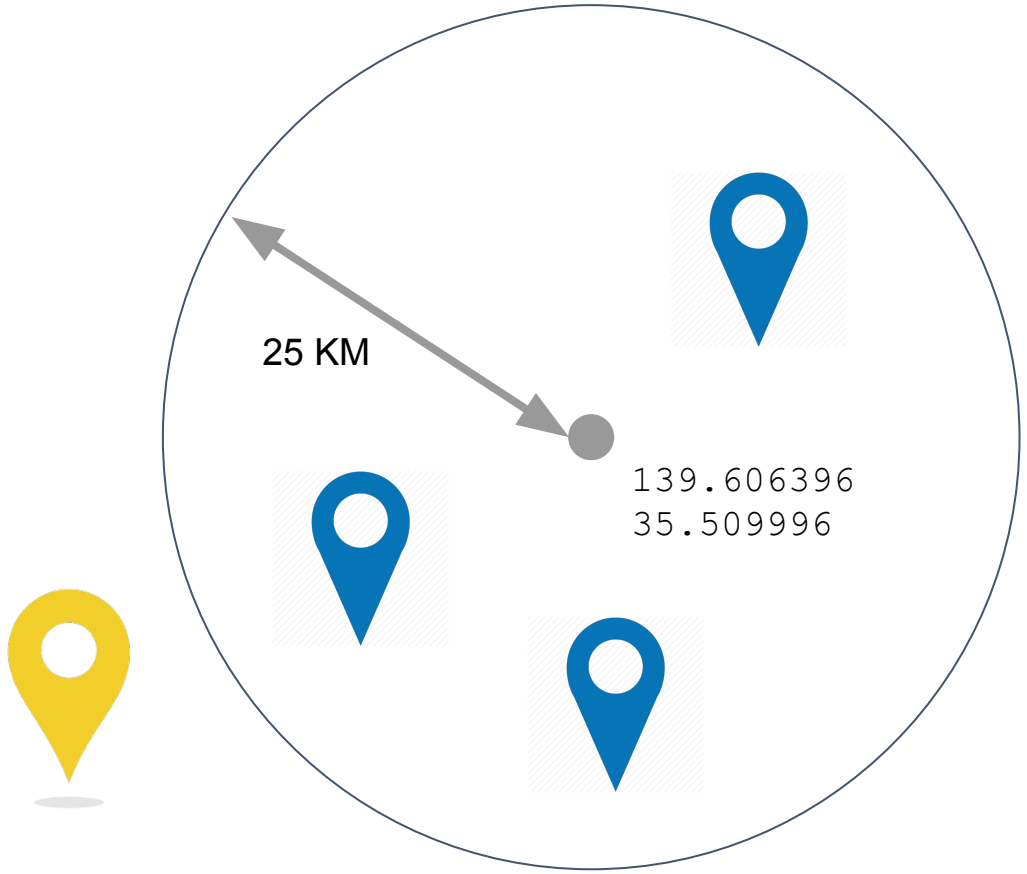
```
> geoadd venues 139.75 35.693333
"Nippon Budokan"
```

Geospatial

- Named point
- Position
  - Long/Lat coordinates
  - 52 bit Geo hash computed
- Stored in Sorted Set

Geospatial
Within radius of point

```
> georadius venues 139.606396 35.509996 25 km
```

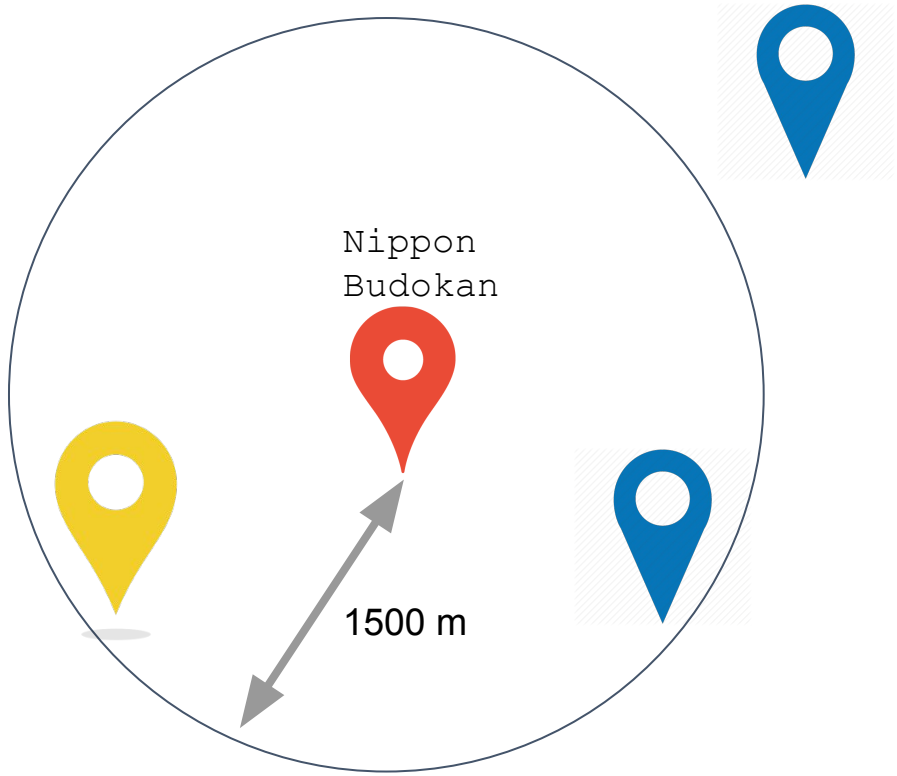


Geo Radius

- Distance from Long/Lat
  - Units Meters, Km, Feet, Miles
- Return
  - Distance from point
  - Coordinates of point
  - Geo Hash

Geospatial
Within radius of Named point

```
> georadiusbymember venues "Nippon Budokan" 1500 m
```

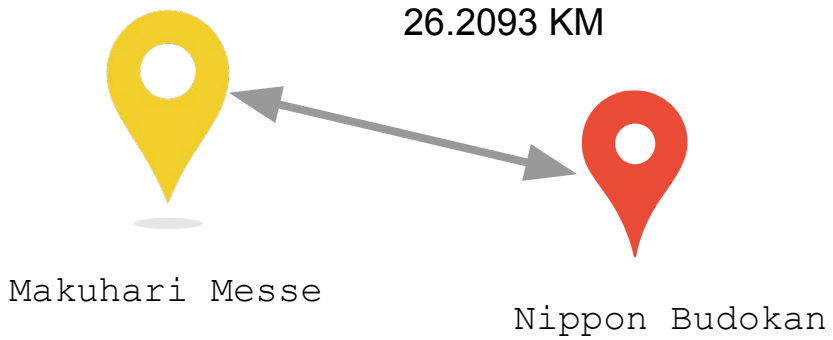


Geo Radius

- Distance from Named point
  - Units Meters, Km, Feet, Miles
- Return
  - Distance from point
  - Coordinates of point
  - Geo hash

Geospatial
Distance between named points

```
> geodist venues "Makuhari Messe" "Nippon Budokan" km
```



Geo Distance

- Distance between two points
  - Units Meters, Km, Feet, Miles
- Return
  - Distance from point
  - Coordinates of point

Why Geospatial is cool

Use case

- AdTech
  - Find audience to broadcast offer
  - Find offers near me
- Social
  - Which friends are close
  - Where's a good place to eat near Market & Church?
- Logistics
  - Drop off / Pick ups, routing

Publish / Subscribe

```
> publish news:Judo "Event starts in 1 hour"
```

```
> subscribe news:Judo
```

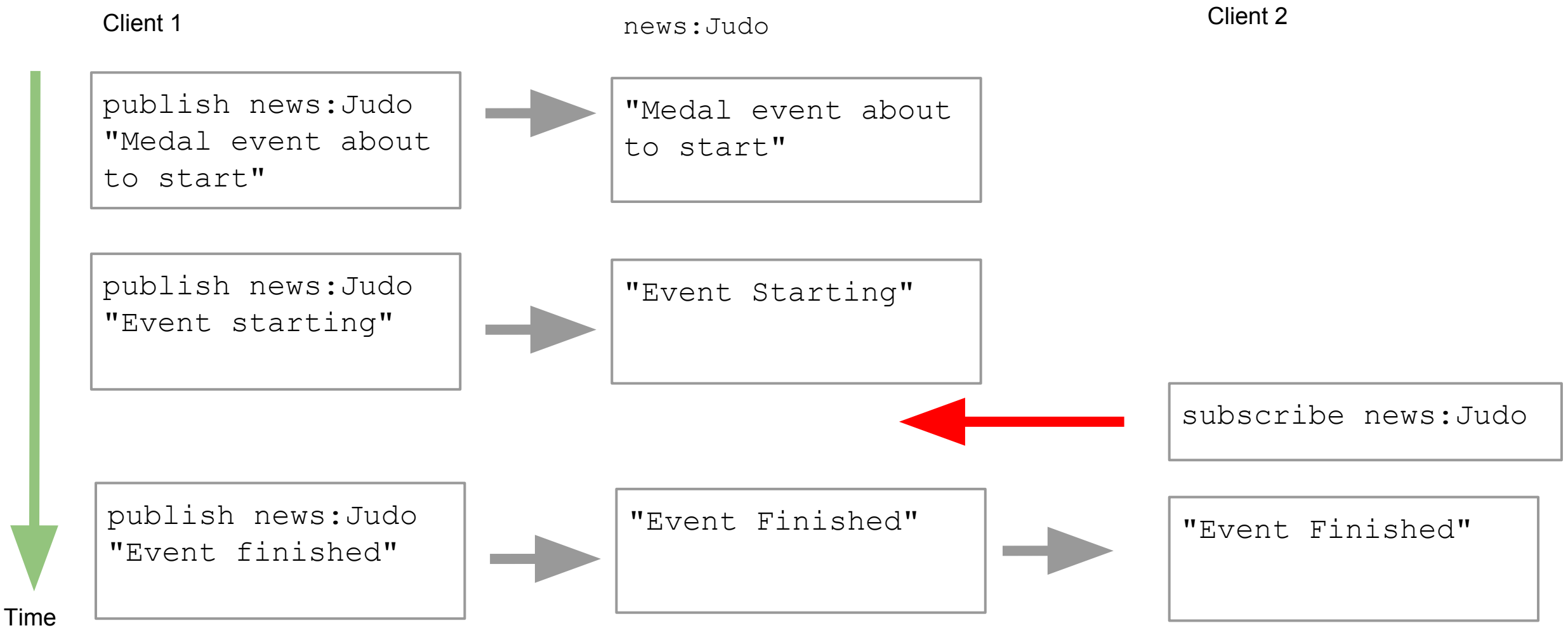
```
> psubscribe news:*
```

```
> unsubscribe news:Judo
```

Publish / Subscribe

- Publish to a single channel
- Subscribe
  - One or more channels
  - Glob-style wildcards

Publish / Subscribe



Why Publish / Subscribe is cool

Use case

- Interprocess communications

Conclusions

Data is modeled different in Redis

- Understand
  - Your Use Case & operations required
  - Data cardinality & Distribution
  - Select Data Structure

- Think about
  - Flat name space - what is your naming convention?
  - Transactions - atomic, enable safe multi key writes
  - Relationships - flatten into Hashes, or combine with Sets

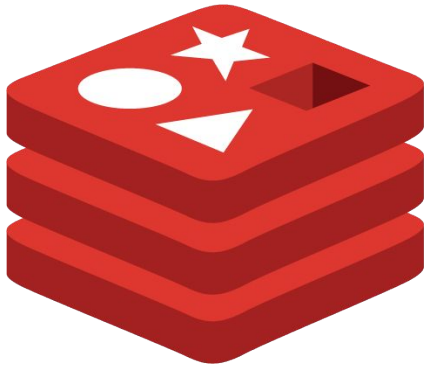
And things we did not talk about

HyperLogLog - probabilistic counting structure

Bitfields - compact bit structures

Modules

- ReJSON - full JSON support
- RediSearch - text search



redis
UNIVERSITY

And finally…

- RU101 - An Introduction to Redis Data Structures
- Registration starts 25th April
- Course starts early June
  - Online, classroom style
  - 6 weeks duration
  - 2-3 hours commitment per week
  - Virtual Labs for every student
  - Certificate of Completion

EVERYWHERE.

Thank You

Key Namespaces

Key Name

```
event:Judo
```

Key namespace

- Logical Database
  - Default is db[0]
- Flat namespace
- Typically used for separation within an application / domain