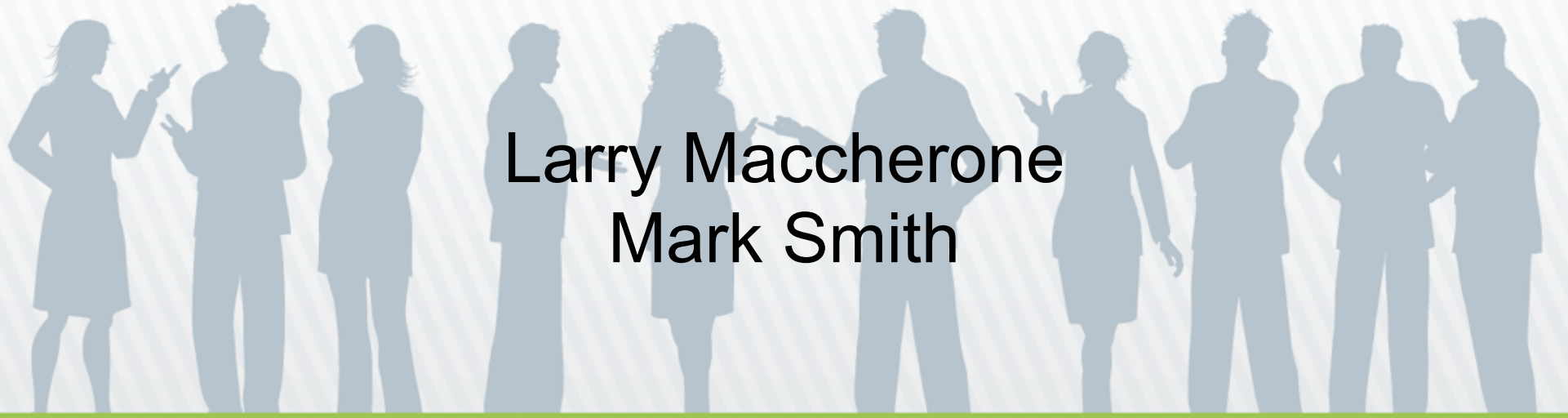
RALLYON!

2012 • AGILE CONFERENCE

Analytics API

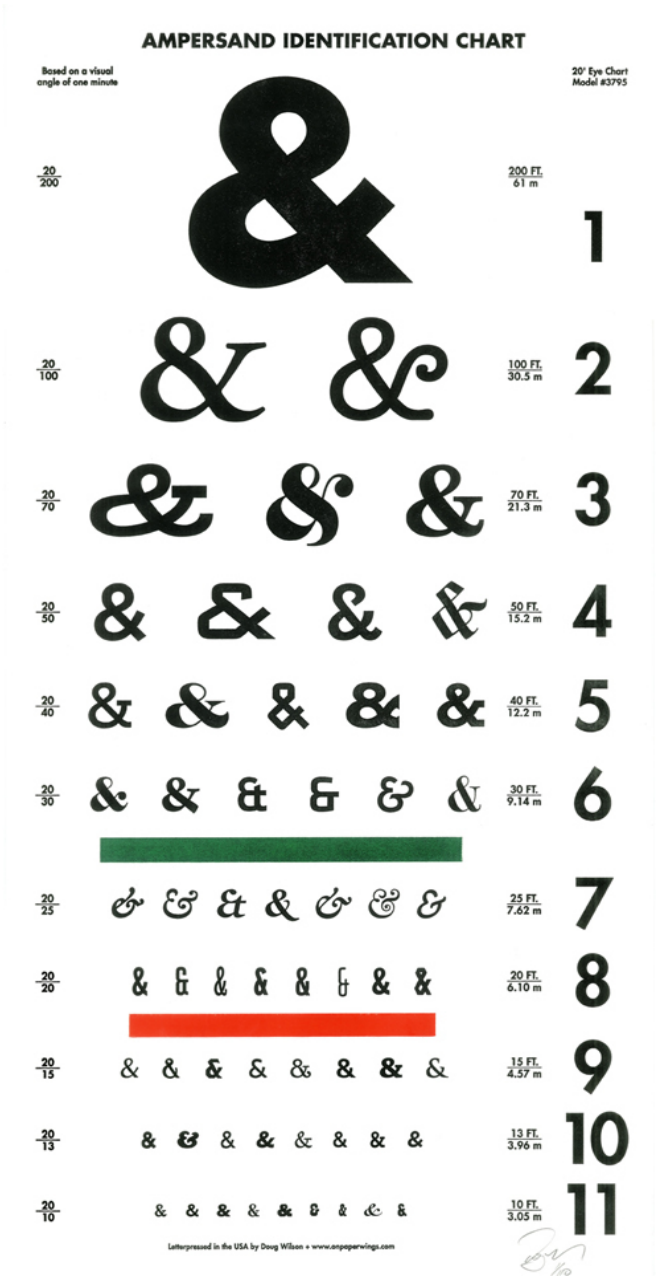


Larry Maccherone
Mark Smith

WHAT

Query-able revision history

- Any work item
- Any field (almost)
- Any time



CONCEPT

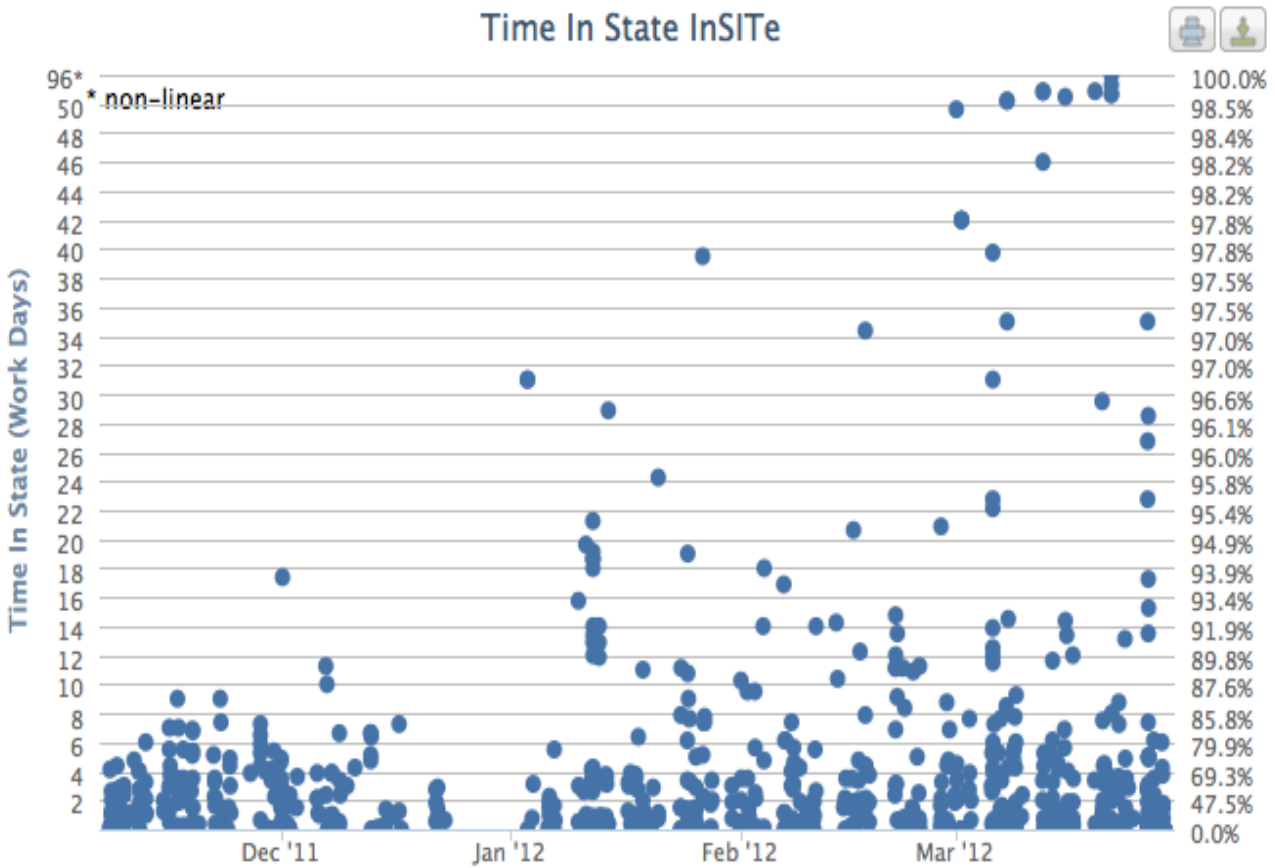
Total dogfood
App SDK
API = Data
Client-side render
Blazingly fast
Scalable

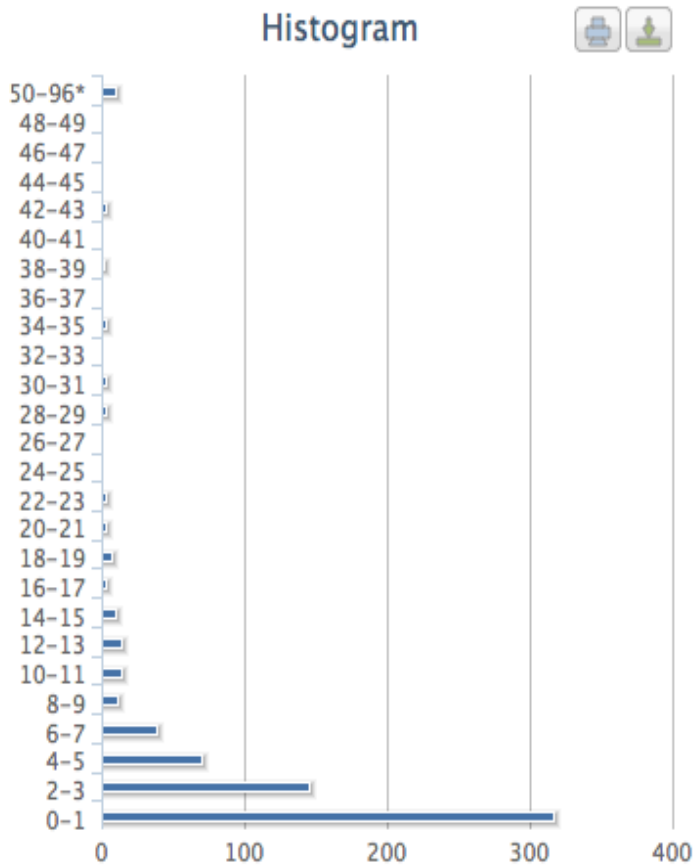
GOAL

Just enough to get started

- Begin with the end in mind
  - Time In State InSITe Chart
- Where to find references/resources/examples
- Concepts
  - MVCC-like data model
  - MongoDB-like query language
- Examples
  - Browser bar GET or XHR POSTer
  - Simple App SDK 2.0 example
  - Cumulative Flow Diagram
    - Cut-and-paste deploy
    - Code walk-through

Begin with the end in mind





RESOURCES

Everything is up on GitHub

- https://github.com/RallyApps/rally_analytics
  - Concepts
  - Raw API usage
  - Lumenize (time-series axis, timezone, etc.) calculations
  - Cycletime and CFD Complex HighCharts examples (but not App SDK 2.0)

- https://github.com/RallyApps/AnalyticsSnapshotGridApp
  - App SDK 2.0 Interactive tool for querying the Analytics API

- https://github.com/RallyApps/AnalyticsChartApp
  - App SDK 2.0
  - Simple HighCharts example
  - Simple Lumenize call

- https://github.com/RallyApps/BurnChartExample
  - App SDK 2.0
  - Advanced HighCharts example
  - Advanced Lumenize usage

MVCC Snapshot data model

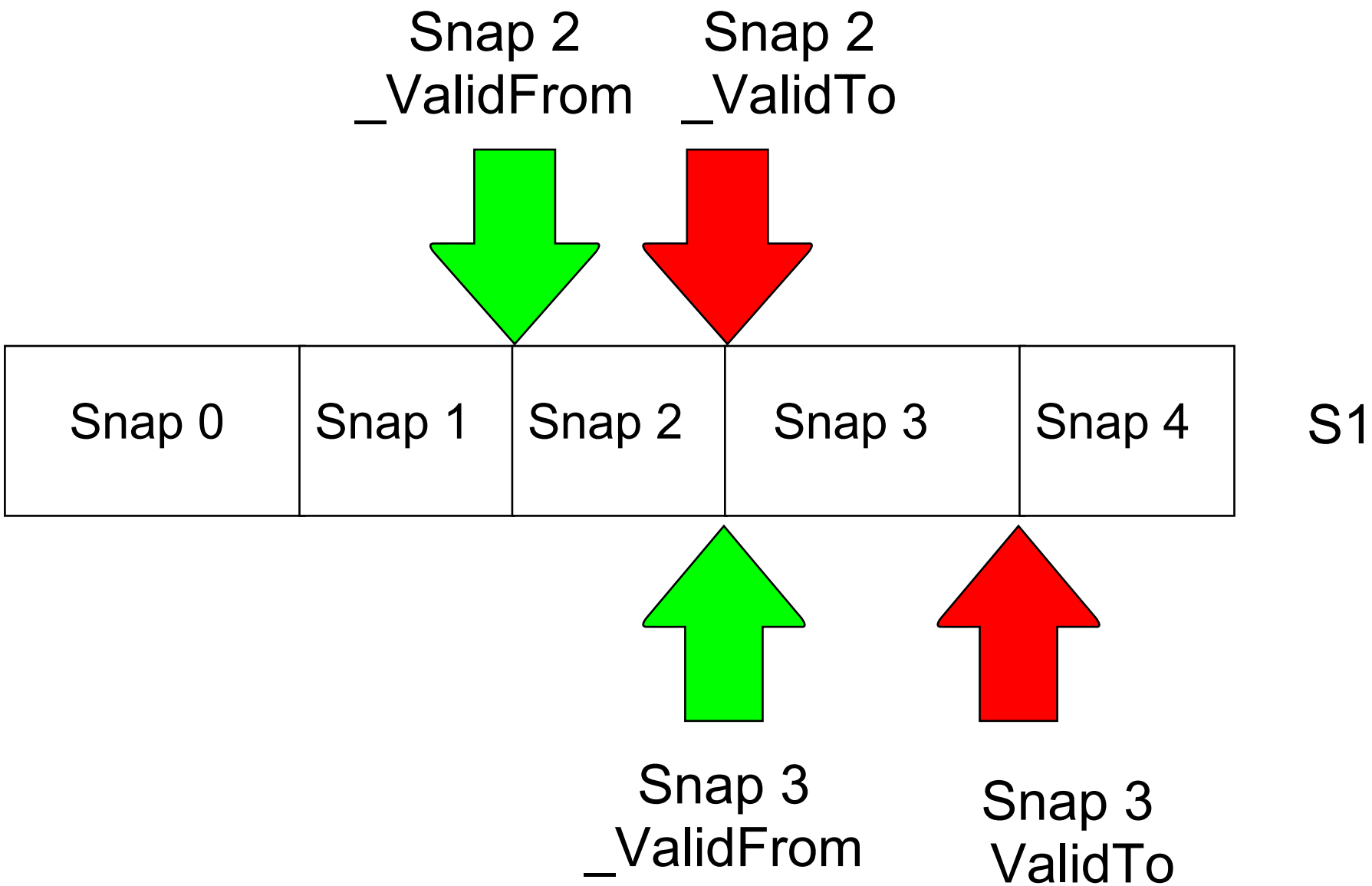
Start with this document

```
{
  _id: B2E,       // GUID just for analytics engine
  ObjectID: 777,  // objectID (OID) from Rally
  _UnformattedID: 2345,
  Name: "Footer disappears when using new menu",
  State: "Submitted",
  _ValidFrom: "2011-01-01T12:34:56Z",
  _ValidTo: "9999-01-01T00:00:00Z",    // "current" snapshot
  ... // Other fields not shown
}
```

Change the State field, and you'll end up with these two documents

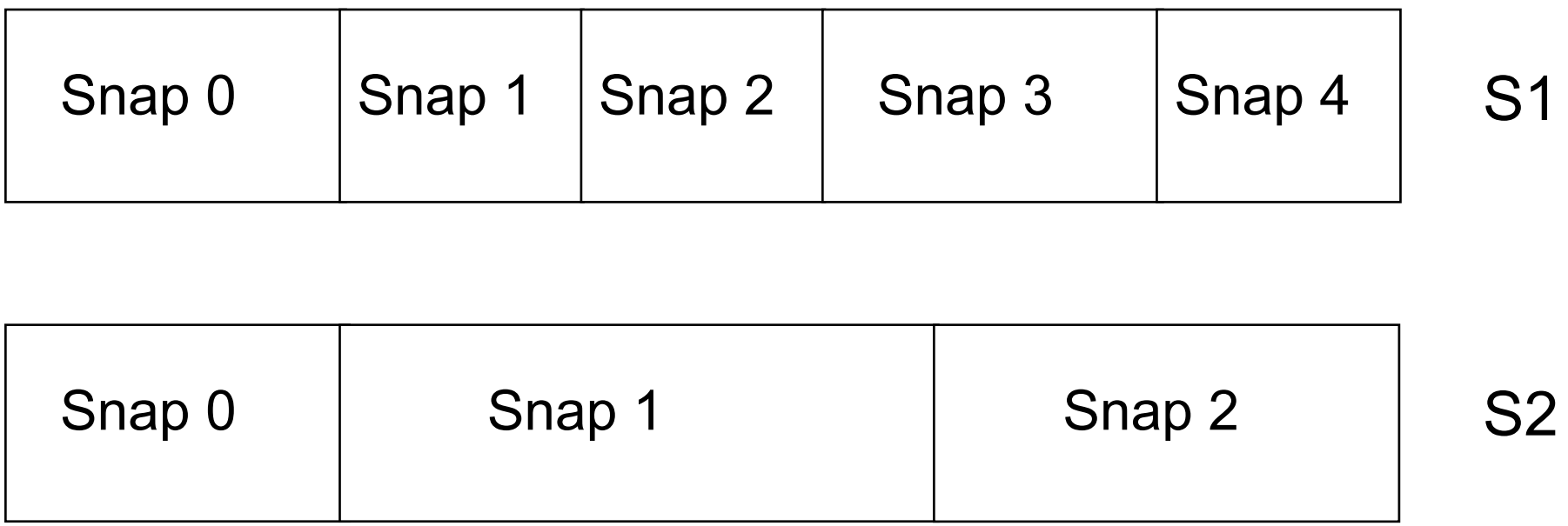
```
{
  _id: B2E,       // GUID just for analytics engine
  ObjectID: 777,  // objectID (OID) from Rally
  _UnformattedID: 2345,
  Name: "Footer disappears when using new menu",
  State: "Submitted",
  _ValidFrom: "2011-01-01T12:34:56Z",
  _ValidTo: "2011-01-02T12:00:00Z", // updated
  ... // Other fields not shown
}
{
  _id: A37,       // a new analytics "document" so it gets a new _id
  ObjectID: 777,  // same old Rally OID
  _UnformattedID: 2345,
  Name: "Footer disappears when using new menu",
  State: "Open",
  _ValidFrom: "2011-01-02:12:00:00Z",  // equals B2G's _ValidTo
  _ValidTo: "9999-01-01T00:00:00Z",
  _PreviousValues: {
    State: "Submitted"
  },
  ... // Other fields not shown
}
```

MVCC Snapshot data model



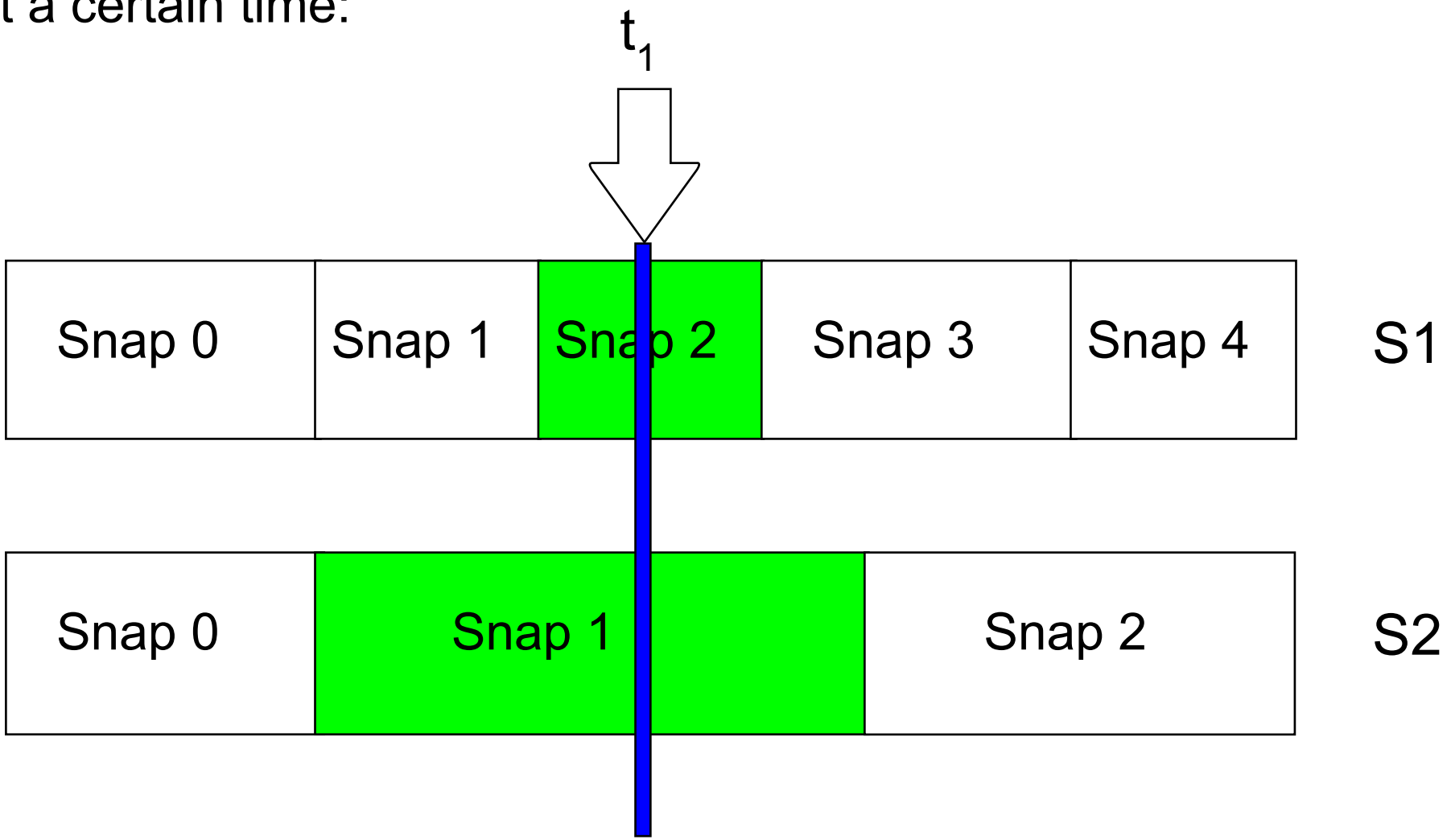
MVCC Snapshot data model

Multiple stories' timelines
don't necessarily line up:



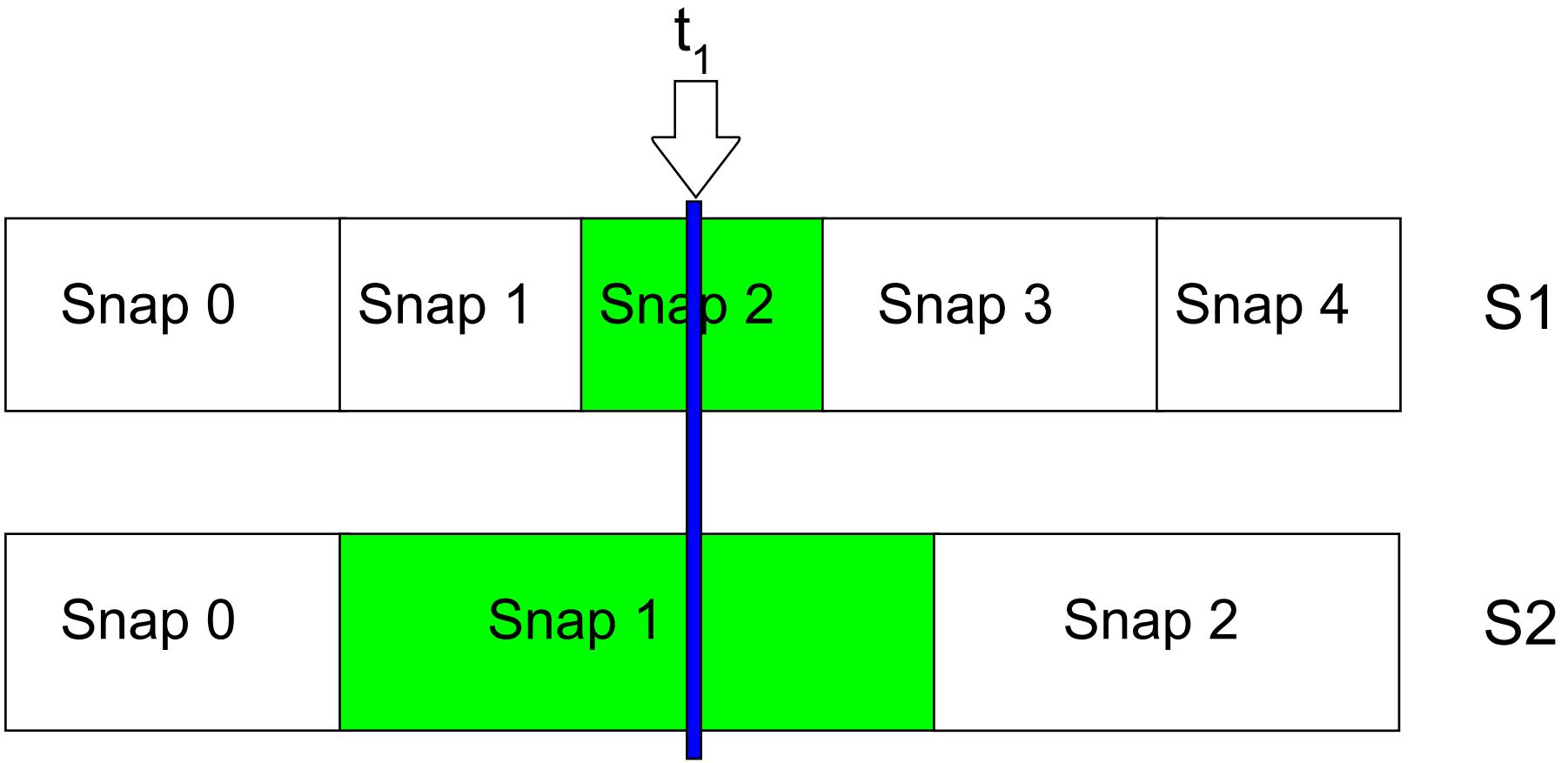
MVCC Snapshot data model

Can look across the world
at a certain time:



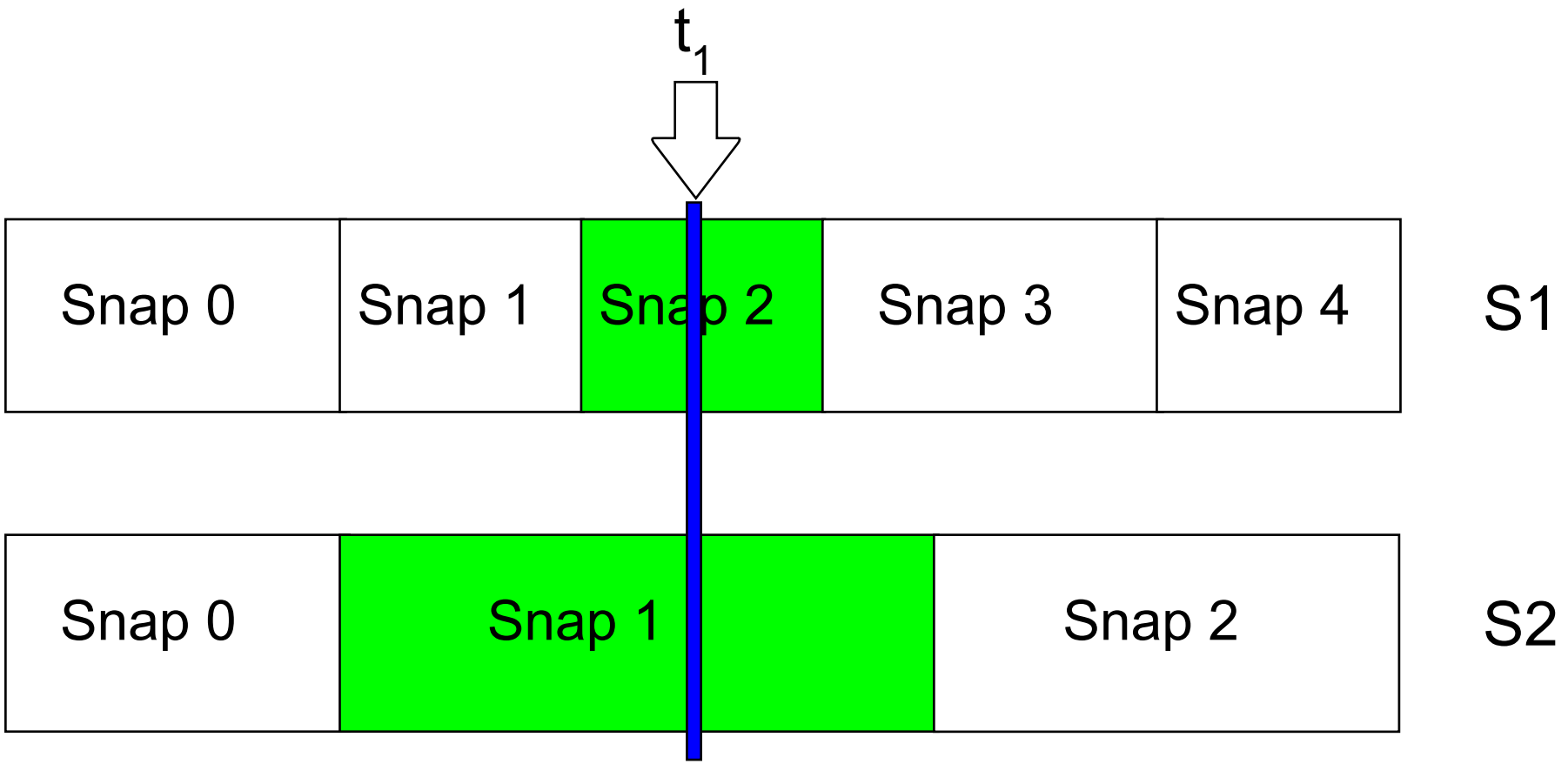
MVCC Snapshot data model: _ _At

find={FormattedID:{$in:["S1", "S2"]},
_ _ **At:** t_1}



MVCC Snapshot data model

find={FormattedID:{$in:["S1", "S2"]},
_ValidTo:{ $gt:t1 }, _ValidFrom:{ $lte:$t_1$ } }



Snapshot data model notes

1. The _PreviousValues field stores the values that were replaced when this particular snapshot was added
2. All timestamps are stored in GMT
3. Big/rich text fields and attachments are excluded but Name is included and so is pretty much every other field including custom ones

POST contents (or GET parameters)

POST

```
{
 find: { // Required
   // Query clauses
 },
fields: ["State", "PlanEstimate"], // Field list. Use true for all.
 pagesize: 1000000, // replaced with min([MAX_RECORDS, <what you provide>)
 start: 0, // Specifies at which object it should begin returning
 sort: {_id: 1},
}
```

or

GET
https://rally1.rallydev.com/analytics/1.27/1234/artifact/snapshot/query.js?find={...}&fields=[...]&... where 1234 is the ObjectID of the workspace.

Demo

Querying via GET

https://rally1.rallydev.com/analytics/1.27/41529001/artifact/snapshot/query.js?find={FormattedID:"S34854"}&fields=true

Querying via POST - use XHR POSTER

url:
https://rally1.rallydev.com/analytics/1.27/41529001/artifact/snapshot/query.js

Content-Type: application/json

content:

```
{
  "find": {"FormattedID": "S34854"},
  "fields": true
}
```

Demo

./examples/minimal-node.coffee

```
{AnalyticsQuery} = require('../')
XMLHttpRequest = require('node-XMLHttpRequest').XMLHttpRequest

config =
  'X-RallyIntegrationName': 'My Chart'
  'X-RallyIntegrationVendor': 'My Company'
  'X-RallyIntegrationVersion': '0.1.0'
  # Get workspaceOID, username, and password from environment variables
  # RALLY_WORKSPACE, RALLY_USER, and RALLY_PASSWORD

query = new AnalyticsQuery(config, XMLHttpRequest)

query.find({FormattedID:"S34854"})
     .fields(true)
     .debug()

callback = () ->
  console.log(this.allResults)

query.getAll(callback)
```

Find - MongoDB operators

- {a: 10} - docs where a is 10 or an array containing the value 10
- {a: 10, b: "hello"} - docs where a is 10 and b is "hello"
- {a: {$gt: 10}} - docs where a > 10, also $lt, $gte, and $lte
- {a: {$ne: 10}} - docs where a != 10
- {a: {$in: [10, "hello"]}} - docs where a is either 10 or "hello"
- {a: {$exists: true}} - docs containing an "a" field
- {a: {$exists: false}} - docs not containing an "a" field
- {a: {$type: 2}} - docs where a is a string (see bsonspec.org for more types)
- {a: /foo.*bar/} - docs where a matches the regular expression "foo.*bar"
- {"a.b": 10} - docs where a is an embedded document where b is 10
- {$or: [{a: 1}, {b: 2}]} - docs where a is 1 or b is 2
- {$and: [{a: 1}, {b: 2}]} - docs where a is 1 and b is 2

Demo (http://try.mongodb.org)

```
db.rally2.save({name:'Westley', nature:'good', weight:150})
db.rally2.save({name:'Roberts', nature:'bad', weight:150})
db.rally2.save({name:'Buttercup', nature:'good', weight:120})
db.rally2.save({name:'Fezzik', nature:'good', weight:340})
db.rally2.save({name:'Humperdinck', nature:'bad', weight:185})

db.rally2.find(null, {_id:0})
// shows all documents

db.rally2.find({nature:'bad'}, ['name'])
// Roberts and Humperdinck

db.rally2.find({weight:{$gte:185}}, {_id:0})
// Fezzik and Humperdinck

db.rally2.find({nature:'good', weight:{$gte:130, $lt:300}}, {_id:0})
// Westley

db.rally2.find({$or: [{nature:'good'}, {name:'Roberts'}]}, {_id:0})
// Westley, Roberts, Buttercup, and Fezzik

db.rally2.find({weight:{$gte: 130}, weight:{$lt: 300}}, {_id:0})
// BAD - DON'T USE A KEY MORE THAN ONCE IN THE SAME MAP
```

Demo hierarchy

```
db.rally.save({ObjectID: 1, _ItemHierarchy:[1]})
db.rally.save({ObjectID: 2, _ItemHierarchy:[1, 2]})
db.rally.save({ObjectID: 3, _ItemHierarchy:[1, 2, 3]})
db.rally.save({ObjectID: 4, _ItemHierarchy:[1, 4]})
db.rally.save({ObjectID: 5, _ItemHierarchy:[5]})

db.rally.find(null, {_id:0})
// shows all documents

db.rally.find({_ItemHierarchy:1}, {_id:0})
// shows documents 1-4

db.rally.find({_ItemHierarchy:2}, {_id:0})
// shows documents 2, and 3
```

Work item hierarchy

Similarly for HiearchicalRequirements (User Stories), if you have this:

- Story 333
  - Story 444
    - Story 555
      - Story 666
        - Defect 777 (via the Requirement field)
          - Task 12
        - Task 13
    - Story 888
    - Story 999

The document for Story 666 would look like this:

```
{
  ObjectID: 666,
  _Type: "HierarchicalRequirement",
  Parent: 555,
  _ItemHierarchy: [333, 444, 555, 666],
  ...
}
```

To get all Stories that descend from 333 (includes 333, 444, 555, 666, 888, and 999 but not Defect 777):

```
{
  _ItemHierarchy: 333,
  _Type: "HierarchicalRequirement"
}
```

Getting leaf nodes only

(1) for just Stories:

```
Children: null
```

(2) for Stories or Portfolio Items:

```
$or: [
    {_Type: "HierarchicalRequirement", Children: null},
    {_Type:"PortfolioItem", Children: null, UserStories: null}
]
```

github Search... Explore Gist Blog Help

RallyApps (Rally Software)

Repositories (5)

Find a Repository...

All Repositories Public Private Sources Forks Mirrors

AnalyticsSnapshotGridApp Ruby 1 1

Sample Analytics API App, using Rally App SDK 2.0 to query and display snapshots in a grid.

Last updated 4 hours ago

all commits commits by owner 52 week participation

markmsmith-work (Mark Smith)
2 public repos, 0 followers