Linked Object Model

Principles and usage report

Petr Štěpánek,
Director for Research and Development

e-Fractal Ltd.

Petr.Stepanek@eFractal.cz

Content

- ✓ What do I call "standard" object model?

- ✓ In what situation the "standard" is not sufficient

- ✓ What chances do we have to remedy it

- ✓ Linked object model - What is it
- ✓ Linked object model - Technical aspects
- ✓ Linked object model - Technical Pros/Cons
- ✓ Linked object model - Business cost/impact

- ✓ Closing – discussion

What do I call “standard” object model?

Conceptual level concepts of *Composition* and *Association* are implemented via inst vars

fine…

How to keep track of data changes made over time?

In what situation the “standard” is not sufficient

1) Who changed what when from what value to what value

2) We need “composite” changes … what state was the model on Timestamp now subtractDays: 10

3) Automatic data feeds cause the changes

Why we bothered?

- ✓ Call Center client
  - ✓ Wants thought-of CRM system
  - ✓ The data model not very known & its complexity will grow
  - ✓ Wants fast & complex queries, no specs yet
  - ✓ Wants us to develop on live system running 24x7

What chances do we have to remedy it

Some possible solutions:

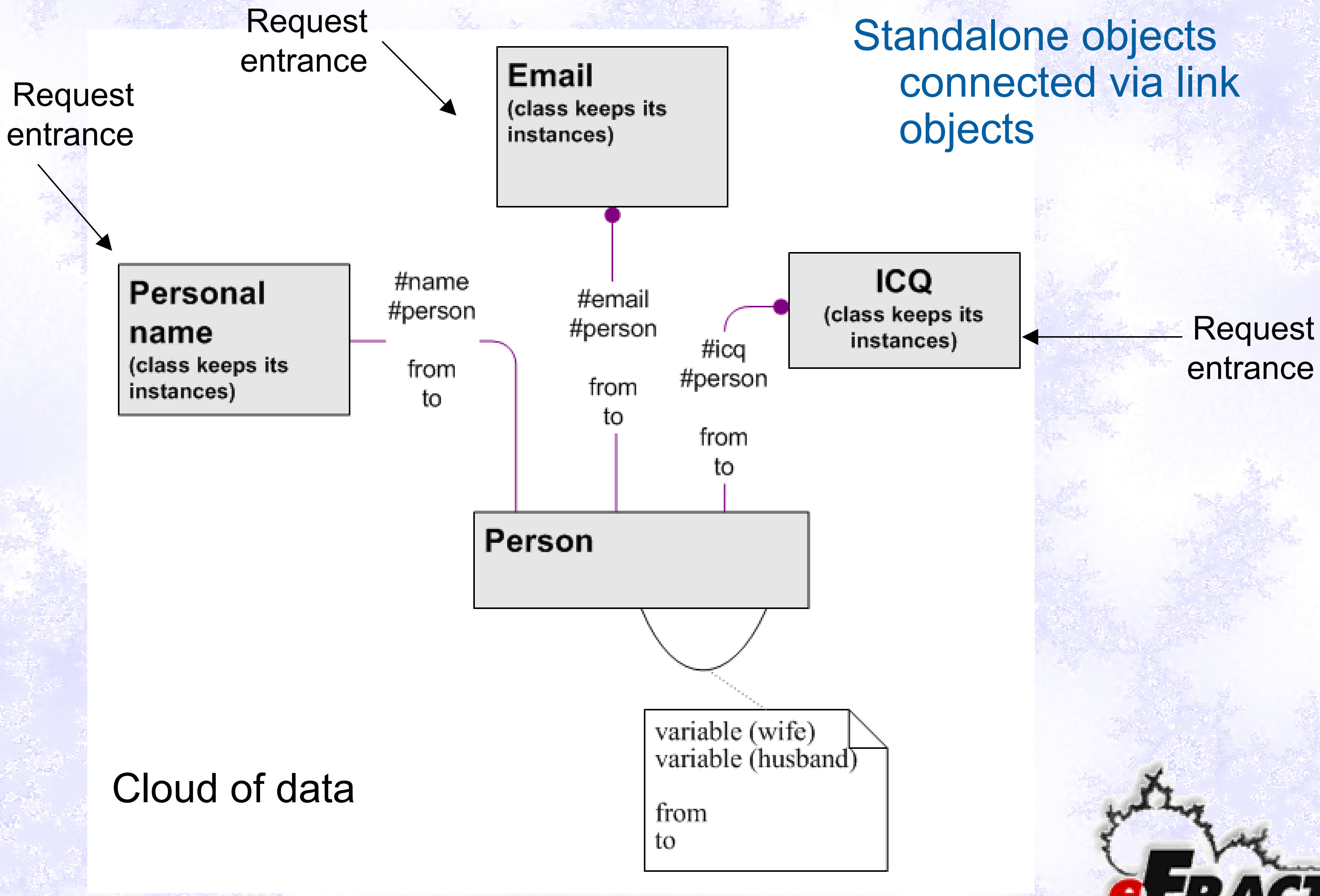
1) HistoryRecord and collection of these
   … cool, but history is accessed differently

2) ValueWithHistory – Kent Beck’s idea of not having real objects in inst vars, but have Decorators there. One of these can be change watcher
   … cool, but history is accessed differently

3) Linked object model
   … cooler, history is NOT accessed differently

Linked object model - What is it

Standalone objects connected via link objects



Linked object model - Business cost/impact

Good for models that are
- complicated
- evolving

Past data is equaly important as current data … CRM systems!

Stochastic requests … Call Center

Fast responses for requests commit from any direction. Why? It prepares for the answer

Linked object model – Technical aspects

Node class: *links*

Link class: *validFrom validTo meaning12 meaning21 node1 node2*

What DB paradigm?

RDB – two huge tables (+additional tables for Node subclasses)

OODB - ☺ we use GemStone/S

How to get/change data?

Write accessors!

Linked object model – Technical aspects II

Setter:

```
Link new
    node1: aPerson;
    node2: aName;
    meaning12: ‘name’;
    meaning21: ‘person’;
    hook
```

Getter:

```
Person >> nameAt: aTimestamp
    ^OneLinkTraverser new
        node1: self;
        meaning12: ‘name';
        validAt: Timestamp now;
        node2.
```

Link

Methods:

hook
invalidateNow
invalidateAt:
unhook

OneLinkTraverser

inst vars:

node1 node2
meaning12 meaning21
validFrom validTo

Methods:

node2
node2OrNil
nodes2
links…

Linked object model – Technical aspects III

1) **Keep links in GemStone**

2) **Implement OneLinkTraverser in GemStone**

Node instVarNamed: ‘links’	- is aRcIdentityBag
OneLinkTraverser	- can be created in ST, but evaluates itself in GS

3) **Cache**

Person has inst var *nameCache*.

When aLink gets hooked/unhooked/invalidated, it notifies its nodes

Linked object model - Technical Pros/Cons

Cons:

Complicated - not a natural approach

Confusing - cannot inspect (can with Trippy's #inspectorExtraAttributes)

Must have OODB

Slow - can be, but can be fought against (caching)

More complicated data consistency checking

Pros:

Less work than with the other approaches

Past data handled the same way as current data

May change model with no GS class migration on live system

High chance of commiting transaction in GS

Can still use GS indexes

Past data handled the same way as current data?

1) Set global *currentTimestamp* variable.

```
System _sessionStateAt: 21 put: aTimestamp
```

2) Have “default” methods like #name refer to the currentTime

```
Person >> #name

    ^OneLinkTraverser new
            node1: self;
            meaning12: ‘name';
            validAt: (System _sessionStateAt: 21);
            node2
```

Thank you !

Petr Štěpánek

Director for Research and Development

Petr.stepanek@eFractal.cz