PG, JS AND HEALTH IT

pgconf 2016 by @niquola

NIKOLAI RYZHIKOV

CTO of Health Samurai

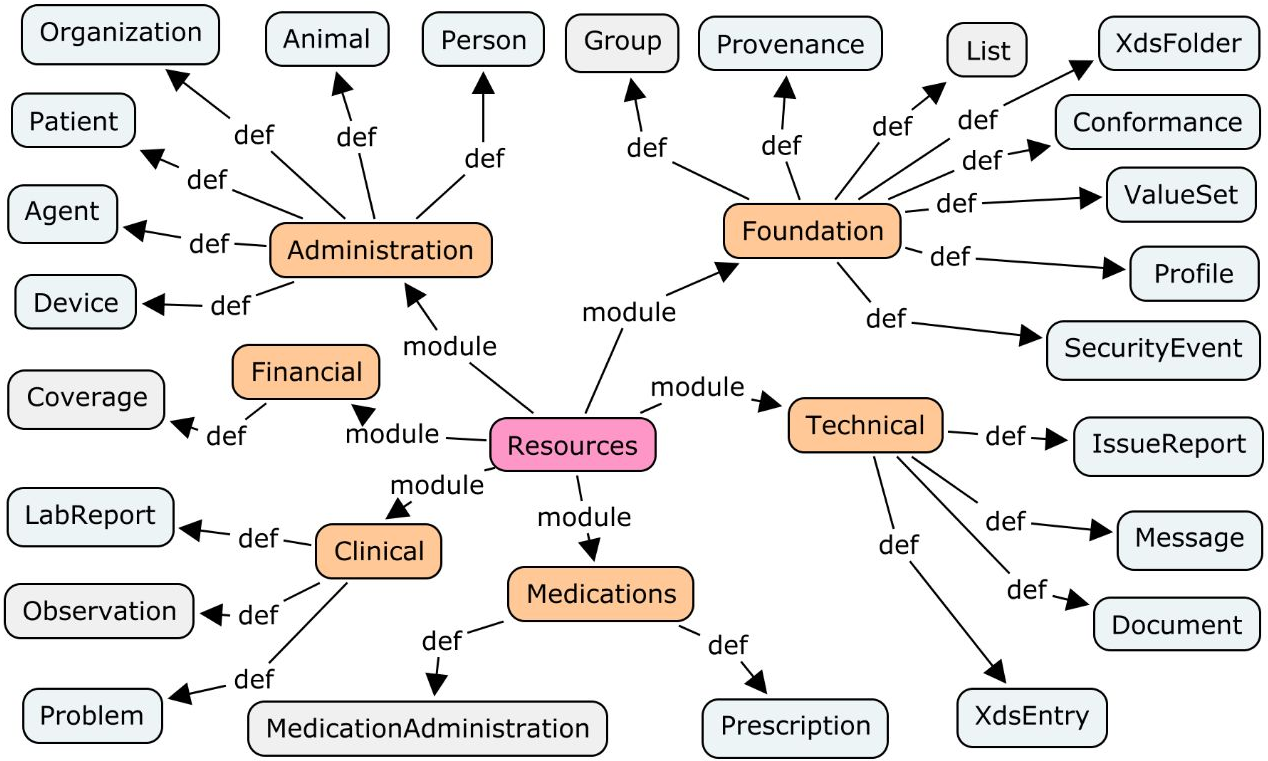
PATIENT CENTRIC

Evidence Based Medicine

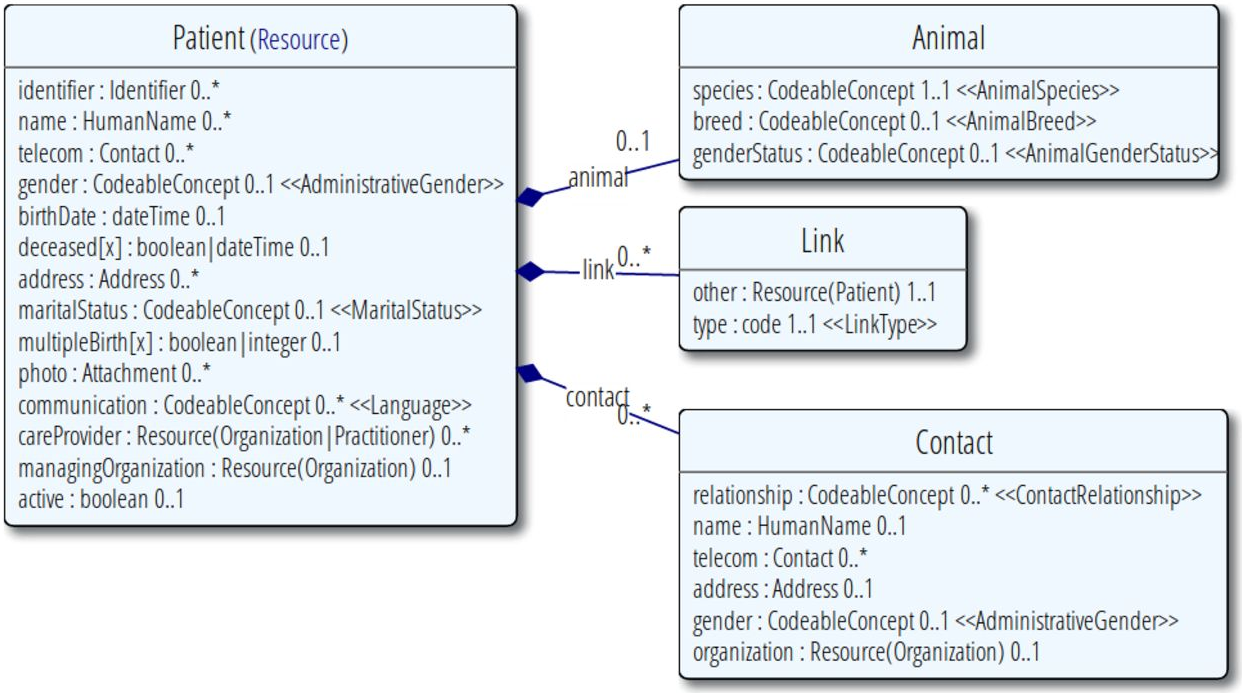
STANDARDS

Fast Health
Interoperability
Resources

INFORMATIONAL MODEL (~100)



PATIENT


Patient (Resource)
identifier : Identifier 0..*
name : HumanName 0..*
telecom : Contact 0..*
gender : CodeableConcept 0..1 <<AdministrativeGender>>
birthDate : dateTime 0..1
deceased[x] : boolean|dateTime 0..1
address : Address 0..*
maritalStatus : CodeableConcept 0..1 <<MaritalStatus>>
multipleBirth[x] : boolean|integer 0..1
photo : Attachment 0..*
communication : CodeableConcept 0..* <<Language>>
careProvider : Resource(Organization|Practitioner) 0..*
managingOrganization : Resource(Organization) 0..1
active : boolean 0..1
0..1
animal
Animal
species : CodeableConcept 1..1 <<AnimalSpecies>>
breed : CodeableConcept 0..1 <<AnimalBreed>>
genderStatus : CodeableConcept 0..1 <<AnimalGenderStatus>>
link
0..*
Link
other : Resource(Patient) 1..1
type : code 1..1 <<LinkType>>
contact
0..*
Contact
relationship : CodeableConcept 0..* <<ContactRelationship>>
name : HumanName 0..1
telecom : Contact 0..*
address : Address 0..1
gender : CodeableConcept 0..1 <<AdministrativeGender>>
organization : Resource(Organization) 0..1

STRUCTURE DEFINITION (META DATA)

```
...
{
    "path": "Patient.id",
    "short": "Logical id of this artifact",
    "definition": "The logical id of the resource, as used in the URL for
    "comments": "The only time that a resource does not have an id is wher
    "min": 0,
    "max": "1",
    "type": [{"code": "id"}],
    "isSummary": true
}
...
```

JSON OR XML DOCUMENT

```
{
 "resourceType": "Patient",
 "identifier": [{
    "use": "usual",
    "label": "MRN",
    "system": "urn:oid:1.2.36.146.595.217.0.1",
    "value": "12345",
    "period": { "start": "2001-05-06" },
    "assigner": { "display": "Acme Healthcare"}
 }],
 "name": [{
    "use": "official",
    "family": [ "Chalmers" ],
    "given": [ "Peter", "James" ]
 }],
...
}
```

EXTENSIONS

```
{"resourceType": "Patient",
 "extension": [
    {
      "url": "http://hl7.org/fhir/Profile/us-core#race",
      "valueCodeableConcept": {
        "coding": [
          {
            "system": "http://hl7.org/fhir/v3/Race",
            "code": "1096-7"
...
```

REST API (OPERATIONS)

```
POST /patient
GET  /patient/<id>
PUT  /patient/<id>
GET  /patient?name=nicola

PATCH /patient/<id>
```

WHY FHIR?

- open
- community
- eco-system

FHIRBASE

DATABASE AS A LIBRARY

STORAGE

```
SELECT fhir_create_storage(plv8, resourceType: 'Patient')

CREATE TABLE patient (
  id text,
  resource jsonb
) inherits (resource);

CREATE TABLE patient_history (id text, resource jsonb);
```

API

```
SELECT fhir_create($JSONB$
  {
    resourceType: "Patient",
    name: "Ivan",
    birthDate: "1981-01-02"
  }
$JSONB$);

fhir_read(resource_type, logical_id)

fhir_update(resource::jsonb)

fhir_vread(resource_type, version_id)

fhir_delete(resource_type, logical_id)

fhir_history(resource_type, logical_id)

fhir_transaction(bundle::jsonb)
```

SEARCH

```
GET [base]/Encounter?length=gt20

GET [base]/Patient/23/Procedure?date=ge2010-01-01&date=le2011-12-31

GET [base]/ValueSet?url:below=http://acme.org/fhir/
# operators
GET [base]/Patient?gender:not=male
# chained params
GET [base]/DiagnosticReport?subject:Patient.name=peter
# eager loading
GET [base]/MedicationOrder?_include=MedicationOrder:patient&criteria...

GET [base]/MedicationOrder?_revinclude=Provenance:target&criteria...
```

HOW SERCH WORKS?

- Query
- StructureDefinition
- SearchParameter

SQL

SEARCH SQL

```
select fhir.search_sql('Patient',
 'given=mark&careprovider:Organization.name=Acme&_count=10');

SELECT *
  FROM patient
  JOIN organization
    ON idx_fns.index_as_reference(
         patient.content,
         '{careProvider}'
       ) && ARRAY[organization.logical_id]::text[]
   AND idx.index_as_string(
          organization.content,
          '{name}'
       ) ilike '%Acme%'
 WHERE idx_fns.index_as_string(
         patient.content,
         '{name,given}'
       ) ilike '%mark%')
 LIMIT 10
OFFSET 0
```

INDEX ON EXPRESSION

```
SELECT fhir_index_parameter($JSON$
  {"resourceType": "Patient", "name": "name"}
$JSON$);

CREATE INDEX
  patient_given_string_idx
  ON patient
  USING gin (
    fhirbase_idx_as_string(
      content, '{name,given}'::text[])
      gin_trgm_ops
   )
;
```

WAITING FOR:

```
select '{"a": {"b": [1,2,3]}}'::jsonb @@ 'a.b.# ($ = 2 | $ < 3)';
select '{"a": {"b": [1,2,3]}}'::jsonb @@ 'a.b.# IN (1,2,5)';
select '{"a": {"b": [1,2,3]}}'::jsonb @@ 'a.b @> [1,2]';
```

JS QUERY

PERFORMANCE

10M PT; MACBOOK AIR (CPU: 1,7 GHZ INTEL CORE I7; MEM: 8GB)

query	ms
patient with unique name	51.238
all Johns in database	220.397
name=John&gender=female	172.586
name=John&gender=male	80.154
name=John&gender=male&active=true&address=YALUMBA	8031.600
name=John&gender=male&_sort=name	234.077
name=John&gender=male&_sort=active	209.552

- Kainos (UK)
- mgrid (ND)
- MIAC & Netrika (RU)
- HSPC (US)
- ePrescriptions (BY)

PROBLEMS

- language
- modularity
- dev. experience

SQL PG/PLSQL

```
func _expand_search_params(_resource_type text, _query text) RETURNS setof query_param
  WITH RECURSIVE params(parent_resource, link_path, res, chain, key, operator, value) AS (
    SELECT null::text as parent_resource, -- we start with empty parent resoure
           '{}'::text[] as link_path, -- path of reference attribute to join
           _resource_type::text as res, -- this is resource to apply condition
           ARRAY[_resource_type]::text[] || key as chain, -- initial chain
           key as key,
           operator as operator,
           value as value
    FROM fhirbase_params._parse_param(_query)
    WHERE key[1] NOT IN ('_tag', '_security', '_profile', '_sort', '_count', '_page')
    UNION
    SELECT res as parent_resource, -- move res to parent_resource
           fhirbase_coll._rest(ri.path) as link_path, -- remove first element
           this.get_reference_type(x.key[1], re.ref_type) as res, -- set next res in chain
           x.chain AS chain, -- save search path
           fhirbase_coll._rest(x.key) AS key, -- remove first item from key untill only one key left
           x.operator,
           x.value
     FROM  params x
     JOIN  searchparameter ri
```

PLV8: V8 JAVASCRIPT IN PG

- Scalar function calls
- Trigger function calls
- Mapping between JS and DB types
- Prepared Statements and Cursors
- Subtransaction & Window function API
- Remote debugger
- Runtime separation across users
- Heroku & RDS

PLV8: FUNCTIONS

```
CREATE FUNCTION plv8_test(keys text[], vals text[])
RETURNS text AS $$
  var o = {};
  for(var i=0; i<keys.length; i++){
      o[keys[i]] = vals[i];
  }
  return JSON.stringify(o);
$$ LANGUAGE plv8 IMMUTABLE STRICT;

SELECT plv8_test(ARRAY['name', 'age'], ARRAY['Tom', '29']);
--         plv8_test
---------------------------
-- {"name":"Tom","age":"29"}
```

PLV8: RETURNING FUNCTION CALLS

```
CREATE TYPE rec AS (i integer, t text);
CREATE FUNCTION set_of_records() RETURNS SETOF rec AS
$$
    // plv8.return_next() stores records in an internal tuplestore,
    // and return all of them at the end of function.
    plv8.return_next( { "i": 1, "t": "a" } );
    plv8.return_next( { "i": 2, "t": "b" } );

    // You can also return records with an array of JSON.
    return [ { "i": 3, "t": "c" }, { "i": 4, "t": "d" } ];
$$
LANGUAGE plv8;

SELECT * FROM set_of_records();
```

PLV8: TRIGGERS

```
CREATE FUNCTION test_trigger() RETURNS trigger AS
$$
    plv8.elog(NOTICE, "NEW = ", JSON.stringify(NEW));
    plv8.elog(NOTICE, "OLD = ", JSON.stringify(OLD));
    plv8.elog(NOTICE, "TG_OP = ", TG_OP);
    plv8.elog(NOTICE, "TG_ARGV = ", TG_ARGV);
    if (TG_OP == "UPDATE") {
        NEW.i = 102;
        return NEW;
    }
$$ LANGUAGE "plv8";

CREATE TRIGGER test_trigger
    BEFORE INSERT OR UPDATE OR DELETE
    ON test_tbl FOR EACH ROW
    EXECUTE PROCEDURE test_trigger('foo', 'bar');
```

PLV8: CURSORS

```
var plan = plv8.prepare(
 'SELECT * FROM tbl WHERE col = $1', ['int']
);
var rows = plan.execute( [1] );
var sum = 0;
for (var i = 0; i < rows.length; i++) {
  sum += rows[i].num;
}
plan.free();

return sum;
```

JS IN PG

- better languages
- performance
- eco-system

PG.JS: MOCK PLV8

```
Client = require('pg-native')
global.INFO="INFO"
global.ERROR="ERROR"
global.DEBUG="DEBUG"

module.exports =
  execute: ->
    client.querySync.apply(client, arguments).map(x) ->
      obj = {}
      ...
  elog: (x, msg) ->
    console.log "#{x}:", msg
    return
  quote_literal: (str)-> str && client.pq.escapeLiteral(str)
  quote_ident: (str)-> str && client.pq.escapeIdentifier(str)
  call: (fn, args...)->
    ...
  require: (nm)->
    require('./loader').scan(nm)
  cache: {}
```

DEV IN NODE

```
namings = require('../core/namings')
pg_meta = require('../core/pg_meta')
...

fhir_create_resource = (plv8, query)->
  resource = query.resource
  throw new Error("expected arguments {resource: ...}") unless resource
  errors = validate_create_resource(resource)
  return errors if errors

  [table_name, hx_table_name, errors] = ensure_table(plv8, resource.resour
  ...

exports.fhir_create_resource = fhir_create_resource
exports.fhir_create_resource.plv8_signature = ['json', 'json']
```

TEST IN NODE

```
plv8 = require('../../plpl/src/plv8')
crud = require('../../src/fhir/crud')
assert = require('assert')

describe "CORE: CRUD spec", ->
  beforeEach ->
    plv8.execute("SET plv8.start_proc = 'plv8_init'")
    schema.fhir_drop_storage(plv8, resourceType: 'Users')
    schema.fhir_create_storage(plv8, resourceType: 'Users')

  it "create", ->
    created = crud.fhir_create_resource(plv8, resource: {resourceType: 'Us
    assert.notEqual(created.id , false)
    assert.notEqual(created.meta.versionId, undefined)
    assert.equal(created.name, 'admin')
```

DEPLOY TO POSTGRES

```
$ ./plpl reload
```

USE FROM POSTGRES

```
SELECT fhir_create_resource($JSONB$
  {
   "resource":
     {
       "resourceType": "Patient",
       "id": "smith",
       "name": [{"given": ["Smith"]}]
     }
   }
$JSONB$);
```

VALIDATION BY JSON SCHEMA

```
{
    "title": "Example Schema",
    "type": "object",
    "properties": {
      "firstName": {"type": "string"},
      "lastName": {"type": "string"},
      "age": {
       "description": "Age in years",
       "type": "integer",
       "minimum": 0
       }
    },
    "required": ["firstName", "lastName"]
}
```

JSON PATCH

```
[
  { "op": "replace", "path": "/baz", "value": "boo" },
  { "op": "add", "path": "/hello", "value": ["world"] },
  { "op": "remove", "path": "/foo"}
]
```

PROBLEMS

- native modules support in plv8
- deploy without re-connect
- faster index functions

DATA PLATFORM (NOBACKEND)

for Data Driven Systems


PostgREST
Serve a RESTful API from any Postgres database

FIREBASE.COM

```
var myFirebaseRef = new Firebase("https://?.firebaseio.com/");

myFirebaseRef.set({
  title: "Hello World!",
  author: "Firebase",
  location: {
    city: "San Francisco",
    state: "California",
    zip: 94103
  }
});
```

APP CODE?

- REST API
- security
- validation

DATA DSL

?