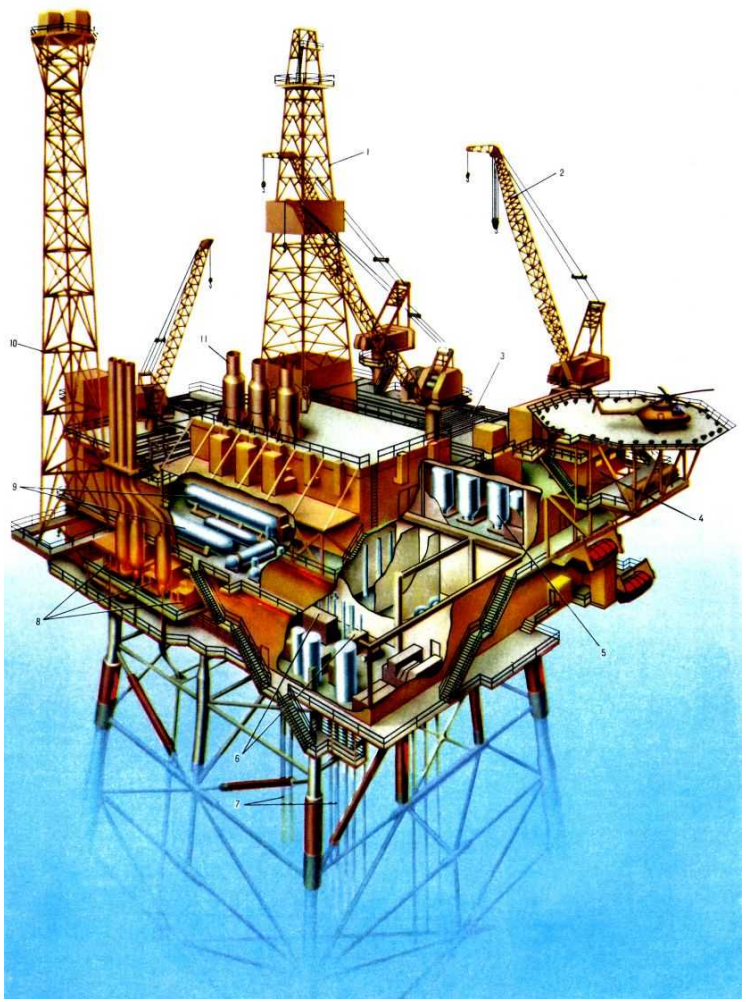
POSTGRESQL
AS APP PLATFORM

Created by niquola / @niquola

PostgreSQL is Operation System
for data

Simon Riggs

APP PLATFORM?



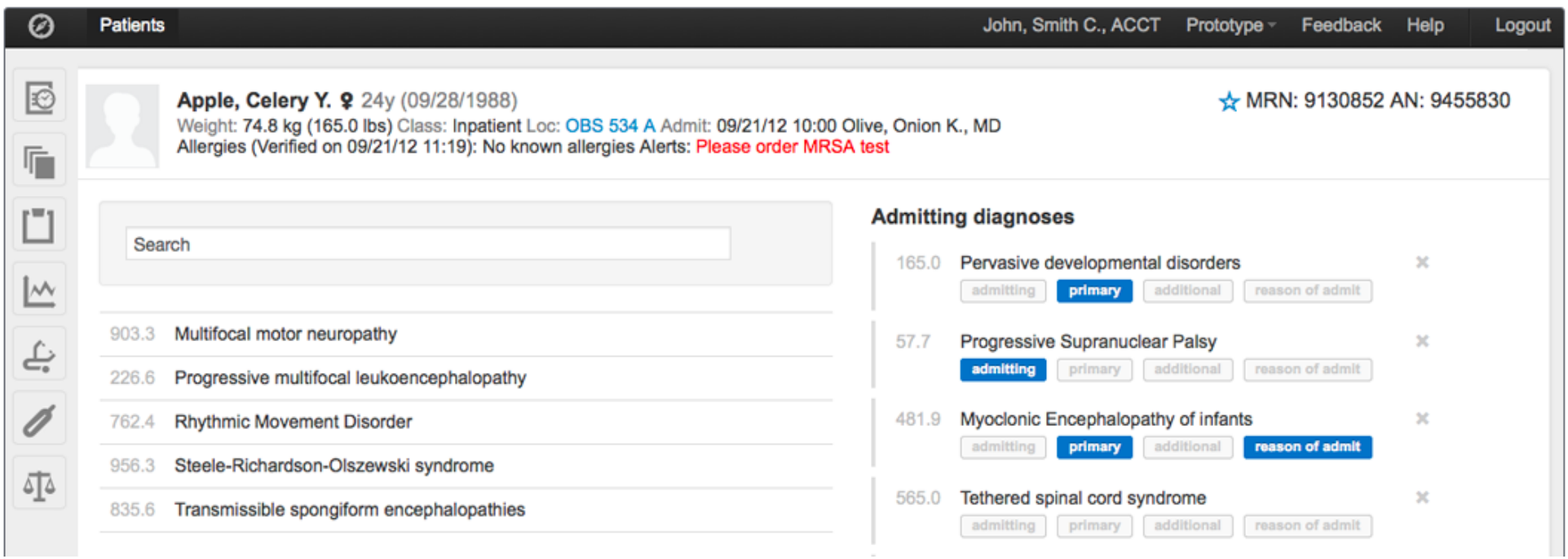
JAVASCRIPT INSIDE POSTGRESQL

NIKOLAY RYZHIKOV

HEALTHSAMURAI

MEDCLIENT EHR

CHOICE HOSPITAL SYSTEMS



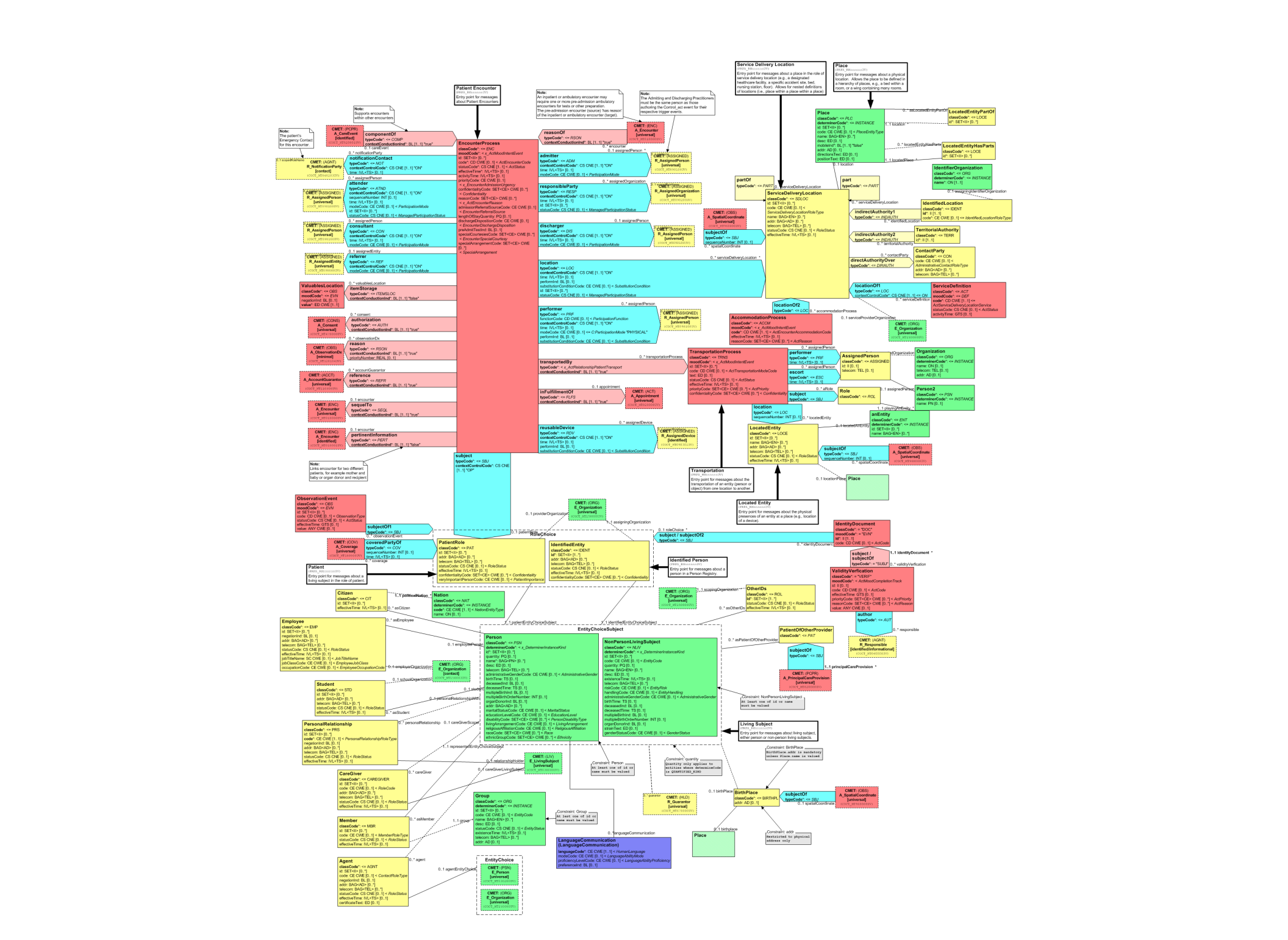
MEDCLIENT

- Monolith + 10 services
- ~0.5M ruby SLOC + jvm, js etc
- Auto install on AWS (chef/sensu/etc)
- 5 databases ~ 1000 tables
- no heigh load (100 sessions)
- no big data (0.5T)
- **structural complexity**


ONC Certified HIT
SM
2014
EDITION


HL7
®
INTERNATIONAL

DOMAIN COMPLEXITY


Patient Encounter
Entry point for messages about Patient Encounters
EncounterProcess
classCode*: <= ENC
moodCode*: < x_ActMoodIntentEvent
id: SET<II> [0..*]
code*: CD CWE [0..1] < ActEncounterCode
statusCode*: CS CNE [1..1] < ActStatus
effectiveTime*: IVL<TS> [0..1]
activityTime: IVL<TS> [0..1]
priorityCode: CE CWE [0..1]
+/- x_EncounterAdmissionUrgency
confidentialityCode: SET<CE> CWE [0..*]
+ Confidentiality
reasonCode: SET<CE> CWE [0..*]
+/- x_ActEncounterReason
admissionReferralSourceCode: CE CWE [0..1]
+ EncounterReferralSource
lengthOfStayQuantity: PQ [0..1]
dischargeDispositionCode: CE CWE [0..1]
+ EncounterDischargeDisposition
preAdmitTestInd: BL [0..1]
specialCourtesiesCode: SET<CE> CWE [0..*]
+ EncounterSpecialCourtesy
specialArrangementCode: SET<CE> CWE [0..*]
+ SpecialArrangement
componentOf
typeCode*: <= COMP
contextConductionInd*: BL [1..1] "true"
CMET: (PCPR) A_CareEvent [Identified]
notificationContact
typeCode*: <= NOT
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
attender
typeCode*: <= ATND
contextControlCode*: CS CNE [1..1] "ON"
sequenceNumber: INT [0..1]
time: IVL<TS> [0..1]
modeCode: CE CWE [0..1] < ParticipationMode
id: SET<II> [0..*]
statusCode: CS CNE [0..1] < ManagedParticipationStatus
CMET: (ASSIGNED) R_AssignedPerson [universal]
consultant
typeCode*: <= CON
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
modeCode: CE CWE [0..1] < ParticipationMode
referrer
typeCode*: <= REF
contextControlCode*: CS CNE [1..1] "ON"
modeCode: CE CWE [0..1] < ParticipationMode
CMET: (ASSIGNED) R_AssignedEntity [universal]
itemStorage
typeCode*: <= ITEMSLOC
contextConductionInd*: BL [1..1] "false"
ValuablesLocation
classCode*: <= OBS
moodCode*: <= EVN
negationInd: BL [0..1]
value*: ED CWE [1..1]
authorization
typeCode*: <= AUTH
contextConductionInd*: BL [1..1] "true"
CMET: (CONS) A_Consent [universal]
reason
typeCode*: <= RSON
contextConductionInd*: BL [1..1] "true"
priorityNumber: REAL [0..1]
CMET: (OBS) A_ObservationDx [minimal]
reference
typeCode*: <= REFR
contextConductionInd*: BL [1..1] "true"
CMET: (ACCT) A_AccountGuarantor [universal]
sequeTo
typeCode*: <= SEQL
contextConductionInd*: BL [1..1] "true"
CMET: (ENC) A_Encounter [universal]
pertinentInformation
typeCode*: <= PERT
contextConductionInd*: BL [1..1] "false"
CMET: (ENC) A_Encounter [identified]
subject
typeCode*: <= SBJ
contextControlCode*: CS CNE [1..1] "OP"
reasonOf
typeCode*: <= RSON
contextConductionInd*: BL [1..1] "true"
admitter
typeCode*: <= ADM
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
modeCode: CE CWE [0..1] < ParticipationMode
CMET: (ASSIGNED) R_AssignedPerson [universal]
responsibleParty
typeCode*: <= RESP
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
id: SET<II> [0..*]
statusCode: CS CNE [0..1] < ManagedParticipationStatus
CMET: (ASSIGNED) R_AssignedOrganization [universal]
discharger
typeCode*: <= DIS
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
modeCode: CE CWE [0..1] < ParticipationMode
CMET: (ASSIGNED) R_AssignedPerson [universal]
location
typeCode*: <= LOC
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
performInd: BL [0..1]
substitutionConditionCode: CE CWE [0..1] < SubstitutionCondition
id: SET<II> [0..*]
statusCode: CS CNE [0..1] < ManagedParticipationStatus
CMET: (OBS) A_SpatialCoordinate [universal]
subjectOf
typeCode*: <= SBJ
sequenceNumber: INT [0..1]
performer
typeCode*: <= PRF
functionCode: CD CWE [0..1] < ParticipationFunction
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
modeCode: CE CWE [0..1] <= CParticipationModePHYSICAL*
performInd: BL [0..1]
substitutionConditionCode: CE CWE [0..1] < SubstitutionCondition
CMET: (ASSIGNED) R_AssignedPerson [universal]
transportedBy
typeCode*: < x_ActRelationshipPatientTransport
contextConductionInd*: BL [1..1] "true"
inFulfillmentOf
typeCode*: <= FLFS
contextConductionInd*: BL [1..1] "true"
CMET: (ACT) A_Appointment [universal]
reusableDevice
typeCode*: <= RDV
contextControlCode*: CS CNE [1..1] "ON"
time: IVL<TS> [0..1]
performInd: BL [0..1]
substitutionConditionCode: CE CWE [0..1] < SubstitutionCondition
CMET: (ASSIGNED) R_AssignedDevice [identified]
Service Delivery Location
Entry point for messages about a place in the role of service delivery location (e.g., a designated healthcare facility, a specific accident site, bed, nursing station, floor). Allows for nested definitions of locations (i.e., place within a place within a place)
Place
Entry point for messages about a physical location. Allows the place to be defined in a hierarchy of places, e.g., a bed within a room, or a wing containing many rooms.
Place
classCode*: <= PLC
determinerCode*: <= INSTANCE
id: SET<II> [0..*]
code: CE CWE [0..1] < PlaceEntityType
name: BAG<EN> [0..*]
desc: ED [0..1]
mobileInd*: BL [1..1] "false"
addr: AD [0..1]
directionsText: ED [0..1]
positionText: ED [0..1]
LocatedEntityPartOf
classCode*: <= LOCE
id*: SET<II> [0..*]
LocatedEntityHasParts
classCode*: <= LOCE
id*: SET<II> [0..*]
IdentifierOrganization
classCode*: <= ORG
determinerCode*: <= INSTANCE
name*: ON [1..1]
partOf
typeCode*: <= PART
part
typeCode*: <= PART
ServiceDeliveryLocation
classCode*: <= SDLOC
id: SET<II> [0..*]
code: CE CWE [0..1] < ServiceDeliveryLocationRoleType
name: BAG<EN> [0..*]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
identifiedLocation
classCode*: <= IDENT
id*: [1..1]
code*: CE CWE [0..1] <= IdentifiedLocationRoleType
indirectAuthority1
typeCode*: <= INDAUTH
indirectAuthority2
typeCode*: <= INDAUTH
TerritorialAuthority
classCode*: <= TERR
id*: [1..1]
directAuthorityOver
typeCode*: <= DIRAUTH
ContactParty
classCode*: <= CON
code: CE CWE [0..1] < AdministrativeContactRoleType
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
locationOf1
typeCode*: <= LOC
contextControlCode*: CS CNE [1..1] <= ON
ServiceDefinition
classCode*: <= ACT
moodCode*: <= DEF
code*: CD CWE [1..1] <= ActServiceDeliveryLocationService
statusCode: CS CNE [0..1] < ActStatus
activityTime: GTS [0..1]
locationOf2
typeCode*: <= LOC
AccommodationProcess
classCode*: <= ACCM
moodCode*: < x_ActMoodIntentEvent
code*: CD CWE [1..1] < ActEncounterAccommodationCode
effectiveTime: IVL<TS> [0..1]
reasonCode: SET<CE> CWE [0..*] < ActReason
CMET: (ORG) E_Organization [universal]
TransportationProcess
classCode*: <= TRNS
moodCode*: < x_ActMoodIntentEvent
id: SET<II> [0..*]
code: CD CWE [0..1] < ActTransportationModeCode
text: ED [0..1]
statusCode: CS CNE [0..1] < ActStatus
effectiveTime: IVL<TS> [0..1]
priorityCode: SET<CE> CWE [0..*] < ActPriority
confidentialityCode: SET<CE> CWE [0..*] < Confidentiality
performer
typeCode*: <= PRF
time: IVL<TS> [0..1]
escort
typeCode*: <= ESC
time: IVL<TS> [0..1]
subject
typeCode*: <= SBJ
AssignedPerson
classCode*: <= ASSIGNED
id: II [0..1]
telecom: TEL [0..1]
Role
classCode*: <= ROL
Organization
classCode*: <= ORG
determinerCode*: <= INSTANCE
name: ON [0..1]
telecom: TEL [0..1]
addr: AD [0..1]
Person2
classCode*: <= PSN
determinerCode*: <= INSTANCE
name: PN [0..1]
location
typeCode*: <= LOC
sequenceNumber: INT [0..1]
anEntity
classCode*: <= ENT
determinerCode*: <= INSTANCE
id: SET<II> [0..*]
name: BAG<EN> [0..*]
LocatedEntity
classCode*: <= LOCE
id: SET<II> [0..*]
name: BAG<EN> [0..*]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
subjectOf
typeCode*: <= SBJ
sequenceNumber: INT [0..1]
CMET: (OBS) A_SpatialCoordinate [universal]
Transportation
Entry point for messages about the transportation of an entity (person or object) from one location to another.
Place
Located Entity
Entry point for messages about the physical presences of an entity at a place (e.g., location of a device)
ObservationEvent
classCode*: <= OBS
moodCode*: <= EVN
id: SET<II> [0..*]
code*: CD CWE [0..1] < ObservationType
statusCode: CS CNE [0..1] < ActStatus
effectiveTime: GTS [0..1]
value: ANY CWE [0..1]
subjectOf1
typeCode*: <= SBJ
CMET: (COV) A_Coverage [universal]
coveredPartyOf
typeCode*: <= COV
sequenceNumber: INT [0..1]
time: IVL<TS> [0..1]
Patient
Entry point for messages about a living subject in the role of patient.
PatientRole
classCode*: <= PAT
id: SET<II> [0..*]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
confidentialityCode: SET<CE> CWE [0..*] < Confidentiality
veryImportantPersonCode: CE CWE [0..1] < PatientImportance
RoleChoice
IdentifiedEntity
classCode*: <= IDENT
id*: SET<II> [1..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
confidentialityCode: SET<CE> CWE [0..*] < Confidentiality
Identified Person
Entry point for messages about a person in a Person Registry.
CMET: (ORG) E_Organization [universal]
OtherIDs
classCode*: <= ROL
id*: SET<II> [1..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
subject / subjectOf2
typeCode*: <= SBJ
IdentityDocument
classCode*: = "DOC"
moodCode*: = "EVN"
id*: [1..1]
code: CD CWE [0..1] < ActCode
subject / subjectOf
typeCode*: = "SUBJ"
ValidityVerification
classCode*: = "VERIF"
moodCode*: < ActMoodCompletionTrack
id: II [0..1]
code: CD CWE [0..1] < ActCode
effectiveTime: GTS [0..1]
priorityCode: SET<CE> CWE [0..*] < ActPriority
reasonCode: SET<CE> CWE [0..*] < ActReason
value: ANY CWE [0..1]
author
typeCode*: <= AUT
CMET: (AGNT) R_Responsible [IdentifiedInformational]
Citizen
classCode*: <= CIT
id: SET<II> [0..*]
effectiveTime: IVL<TS> [0..1]
Nation
classCode*: <= NAT
determinerCode*: <= INSTANCE
code*: CE CWE [1..1] < NationEntityType
name: ON [0..1]
Employee
classCode*: <= EMP
id: SET<II> [0..1]
negationInd: BL [0..1]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
jobTitleName: SC CWE [0..1] < JobTitleName
jobClassCode: CE CWE [0..1] < EmployeeJobClass
occupationCode: CE CWE [0..1] < EmployeeOccupationCode
CMET: (ORG) E_Organization [contact]
Student
classCode*: <= STD
id: SET<II> [0..*]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
PersonalRelationship
classCode*: <= PRS
id: SET<II> [0..1]
code*: CE CWE [1..1] < PersonalRelationshipRoleType
negationInd: BL [0..1]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
CareGiver
classCode*: <= CAREGIVER
id: SET<II> [0..*]
code: CE CWE [0..1] < RoleCode
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
Member
classCode*: <= MBR
id: SET<II> [0..*]
code: CE CWE [0..1] < MemberRoleType
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
Agent
classCode*: <= AGNT
id: SET<II> [0..*]
code: CE CWE [0..1] < ContactRoleType
negationInd: BL [0..1]
addr: BAG<AD> [0..*]
telecom: BAG<TEL> [0..*]
statusCode: CS CNE [0..1] < RoleStatus
effectiveTime: IVL<TS> [0..1]
certificateText: ED [0..1]
EntityChoice
CMET: (PSN) E_Person [universal]
CMET: (ORG) E_Organization [universal]
EntityChoiceSubject
Person
classCode*: <= PSN
determinerCode*: < x_DeterminerInstanceKind
id*: SET<II> [0..*]
quantity: PQ [0..1]
name*: BAG<PN> [0..*]
desc: ED [0..1]
telecom: BAG<TEL> [0..*]
administrativeGenderCode: CE CWE [0..1] < AdministrativeGender
birthTime: TS [0..1]
deceasedInd: BL [0..1]
deceasedTime: TS [0..1]
multipleBirthInd: BL [0..1]
multipleBirthOrder: INT [0..1]
organDonorInd: BL [0..1]
addr: BAG<AD> [0..*]
maritalStatusCode: CE CWE [0..1] < MaritalStatus
educationLevelCode: CE CWE [0..1] < EducationLevel
disabilityCode: SET<CE> CWE [0..*] < PersonDisabilityType
livingArrangementCode: CE CWE [0..1] < LivingArrangement
religiousAffiliationCode: CE CWE [0..1] < ReligiousAffiliation
raceCode: SET<CE> CWE [0..*] < Race
ethnicGroupCode: SET<CE> CWE [0..*] < Ethnicity
NonPersonLivingSubject
classCode*: <= NLIV
determinerCode*: < x_DeterminerInstanceKind
id: SET<II> [0..*]
code: CE CWE [0..1] < EntityCode
quantity: PQ [0..1]
name: BAG<EN> [0..*]
desc: ED [0..1]
existenceTime: IVL<TS> [0..1]
telecom: BAG<TEL> [0..*]
riskCode: CE CWE [0..1] < EntityRisk
handlingCode: CE CWE [0..1] < EntityHandling
administrativeGenderCode: CE CWE [0..1] < AdministrativeGender
birthTime: TS [0..1]
deceasedInd: BL [0..1]
deceasedTime: TS [0..1]
multipleBirthInd: BL [0..1]
multipleBirthOrderNumber: INT [0..1]
organDonorInd: BL [0..1]
strainText: ED [0..1]
genderStatusCode: CE CWE [0..1] < GenderStatus
Constraint: NonPersonLivingSubject
At least one of id or name must be valued
Living Subject
Entry point for messages about living subject, either person or non-living subjects.
PatientOfOtherProvider
classCode*: <= PAT
subjectOf
typeCode*: <= SBJ
CMET: (PCPR) A_PrincipalCareProvision [universal]
Group
classCode*: <= ORG
determinerCode*: <= INSTANCE
id: SET<II> [0..*]
code: CE CWE [0..1] < EntityCode
name: BAG<EN> [0..*]
desc: ED [0..1]
statusCode: CS CNE [0..1] < EntityStatus
existenceTime: IVL<TS> [0..1]
telecom: BAG<TEL> [0..*]
addr: AD [0..1]
Constraint: Group
At least one of id or name must be valued
LanguageCommunication
(LanguageCommunication)
languageCode*: CE CWE [1..1] < HumanLanguage
modeCode: CE CWE [0..1] < LanguageAbilityMode
proficiencyLevelCode: CE CWE [0..1] < LanguageAbilityProficiency
preferenceInd: BL [0..1]
CMET: (HLD) R_Guarantor [universal]
BirthPlace
classCode*: <= BIRTHPL
addr: AD [0..1]
subjectOf
typeCode*: <= SBJ
CMET: (OBS) A_SpatialCoordinate [universal]
Place
Constraint: BirthPlace
BirthPlace.addr is mandatory unless Place.name is valued
Constraint: addr
restricted to physical address only
Constraint: Person
At least one of id or name must be valued
Constraint: quantity
Quantity only applies to entities where determinerCode is QUANTIFIED_KIND
Note: Supports encounters within other encounters
Note: The patient's Emergency Contact for this encounter.
Note: An inpatient or ambulatory encounter may require one or more pre-admission ambulatory encounters for tests or other preparation. The pre-admission encounter (source) 'has reason' of the inpatient or ambulatory encounter (target).
Note: The Admitting and Discharging Practitioners must be the same person as those authoring the Control_act event for their respective trigger events.
Note: Links encounter for two different patients, for example mother and baby or organ donor and recipient

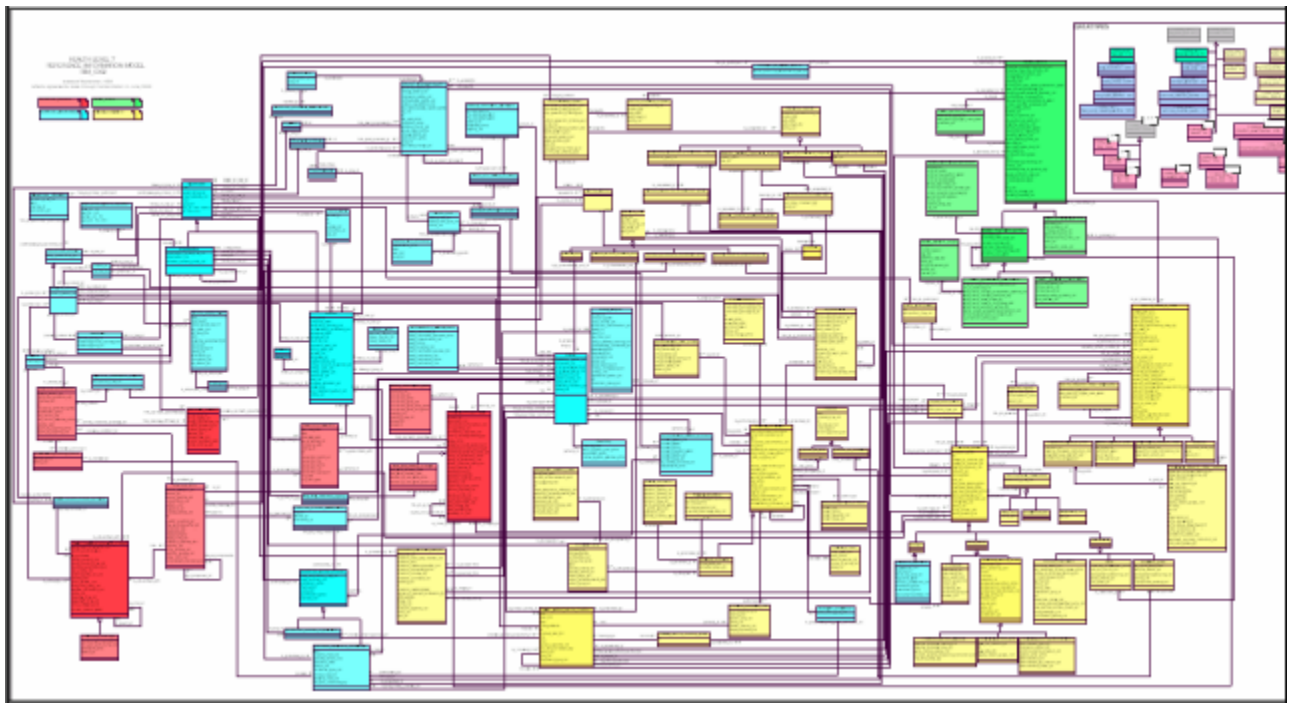
WE NEED STANDARD!

WHICH?

- HL7 v2
- HL7 v3
- OpenEHR
- **FHIR**

MGRID

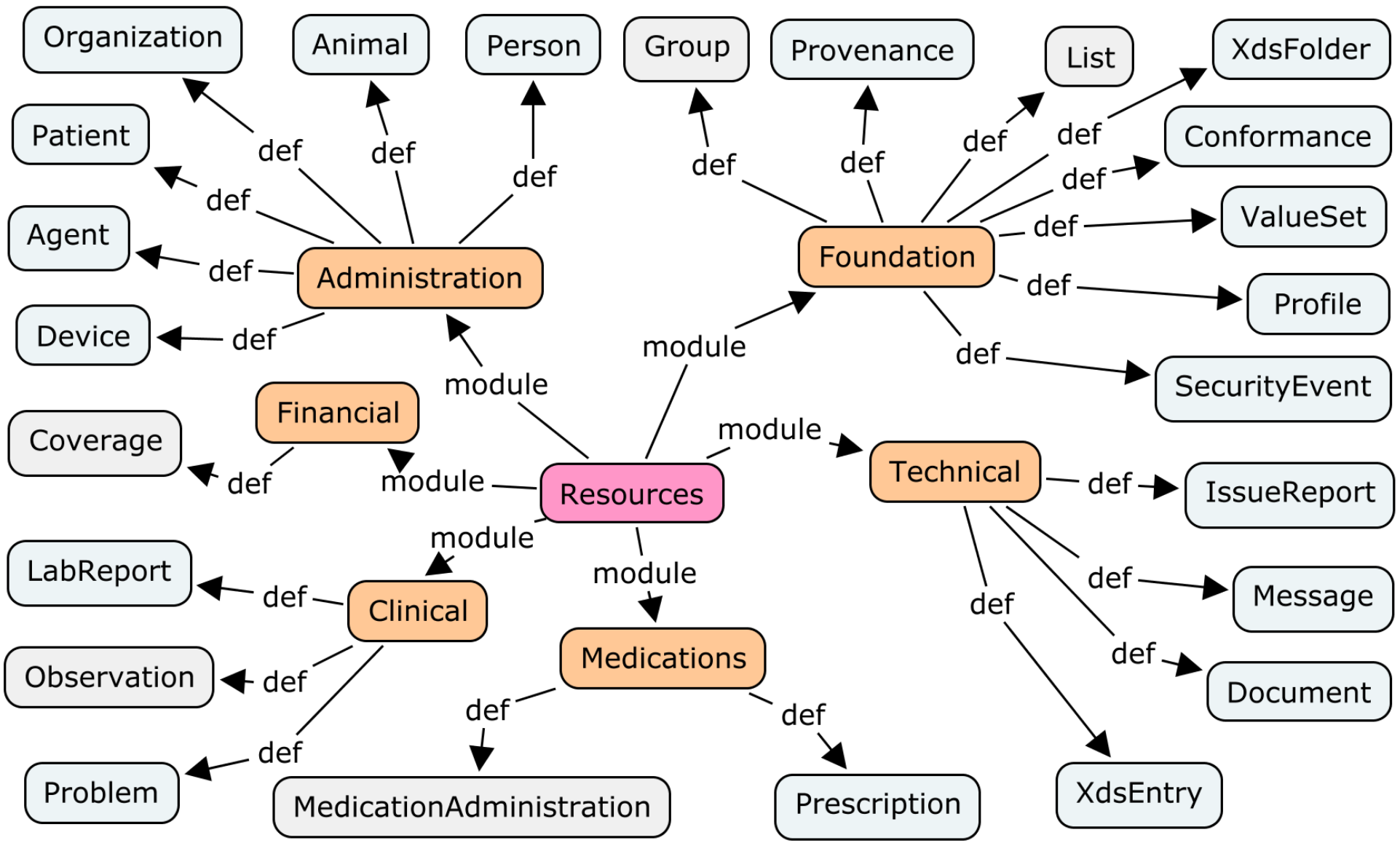
Database for HL7 v3 RIM (Postgres)



FHIR

Fast Healthcare Interoperability Resources

FHIR: ~100 RESOURCES



FHIR: REST API

- CRUD
- History
- Search
- Terminology
- ...

POLYGLOT

RUBY, CLOJURE, JVM, JAVASCRIPT, .NET ...

FHIRBASE

Relational Storage
for FHIR resources

FHIRBASE

```
SELECT fhir.create($JSON$
 {
  resourceType: "Patient",
  name: [{text: "Ivan"}],
  birthDate: "1981-01-02"
 }
$JSON$);

SELECT fhir.search('Patient', 'name=ivan&birthdate=>1970');
```

WORKS!?

- fhirface, Aidbox - clojure
- Netrika (SPb) - .NET
- Kainos (UK) - Java
- ...

BUT, PROGRAMMERS UX IS :(

UX & TOOLING

- own preprocessor
- own modules
- own test framework
- own migrations

PREPROCESSOR

```
func _build_url(_cfg_ jsonb, VARIADIC path text[]) RETURNS text
  SELECT _cfg_->>'base' || '/' || (SELECT string_agg(x, '/')
    FROM unnest(path) x)


CREATE OR REPLACE FUNCTION
module._build_url(_cfg_ jsonb, VARIADIC path text[]) RETURNS text
AS $$
  SELECT _cfg_->>'base' || '/' || (SELECT string_agg(x, '/')
    FROM unnest(path) x)
$$ language SQL immutable;
```

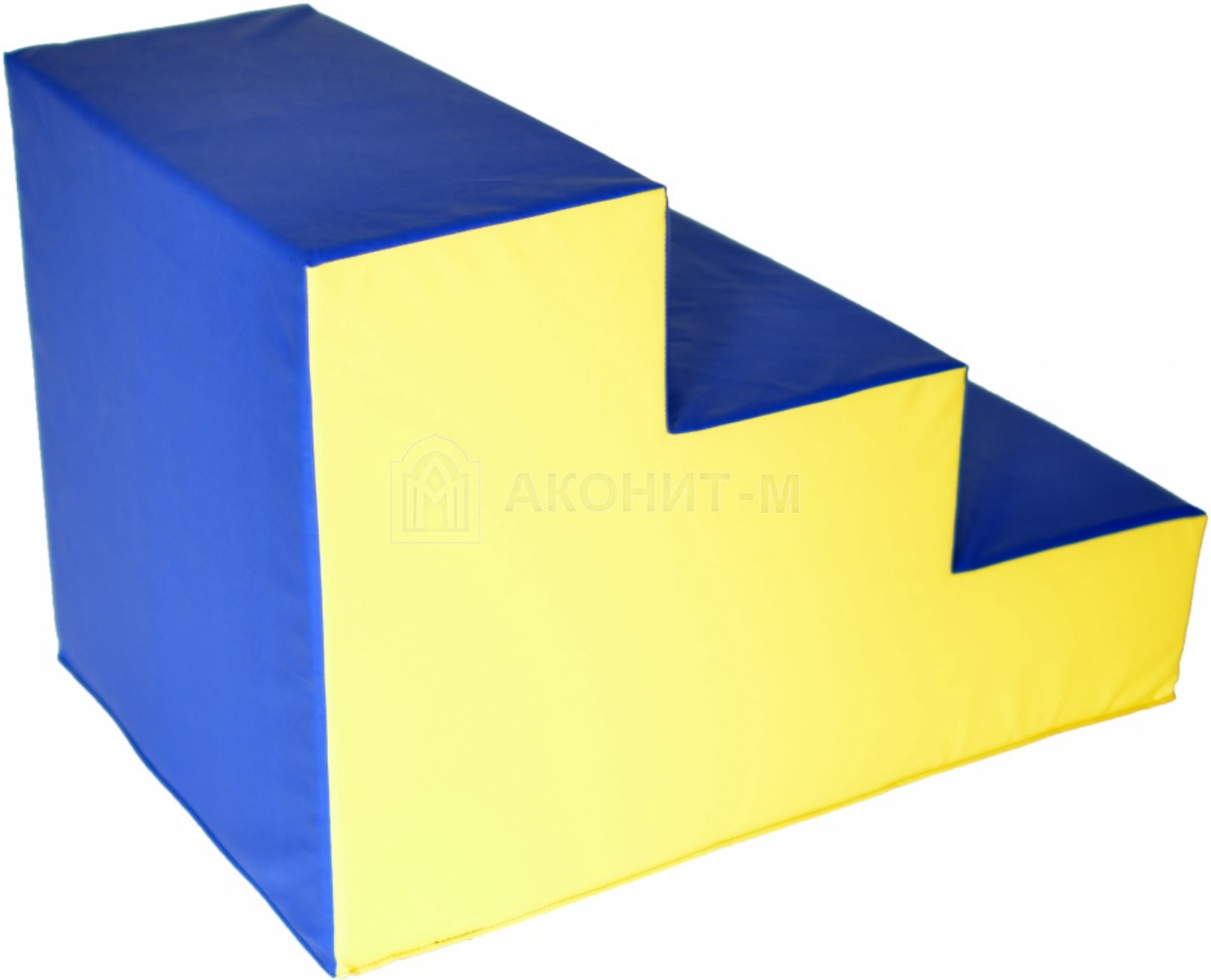
MODULES

```
-- #import ./fhirbase_json.sql
-- #import ./fhirbase_gen.sql
-- #import ./fhirbase_coll.sql
-- #import ./fhirbase_util.sql
-- #import ./fhirbase_generate.sql

func _build_url(_cfg_ jsonb, VARIADIC path text[]) RETURNS text
  SELECT _cfg_->>'base' || '/' || (SELECT string_agg(x, '/')
    FROM unnest(path) x)
```

TESTS

```
BEGIN;
_extract_id('rid/_history/vid') => 'rid'
-- SELECT expect(_extract_id('rid/_history/vid'),'rid')

SELECT fhirbase_generate.generate_tables('{Patient}');

setv('createOutcome',
  fhirbase_crud.create('{}'::jsonb, :'pt_json')
);

getv('createOutcome')->>'resourceType' => 'OperationOutcome'
getv('createOutcome')#>>'{issue,0,code,coding,1,code}' => '400'

ROLLBACK;
```

THRESHOLD


АКОНИТ-М

```
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
       ON  ri.name = split_part(key[1], ':',1)
      AND  ri.base = x.res
     JOIN  structuredefinition_elements re
       ON  re.path = ri.path
    WHERE array_length(key,1) > 1
  )
  SELECT
    parent_resource as parent_resource,
    link_path as link_path,
    res as resource_type,
    fhirbase_coll._butlast(p.chain) as chain,
    ri.search_type,
    ri.is_primitive,
    ri.type,
    fhirbase_coll._rest(ri.path)::text[] as field_path,
    fhirbase_coll._last(key) as key,
    operator,
    value
  FROM params p
  JOIN searchparameter ri
```

LOGIC IN DB

LOGIC IN DB (SMART STORAGE): PRO

- Performance (faster transactions, data locality)
- Consistency (like incapsulation)
- Integration by db
- Reuse

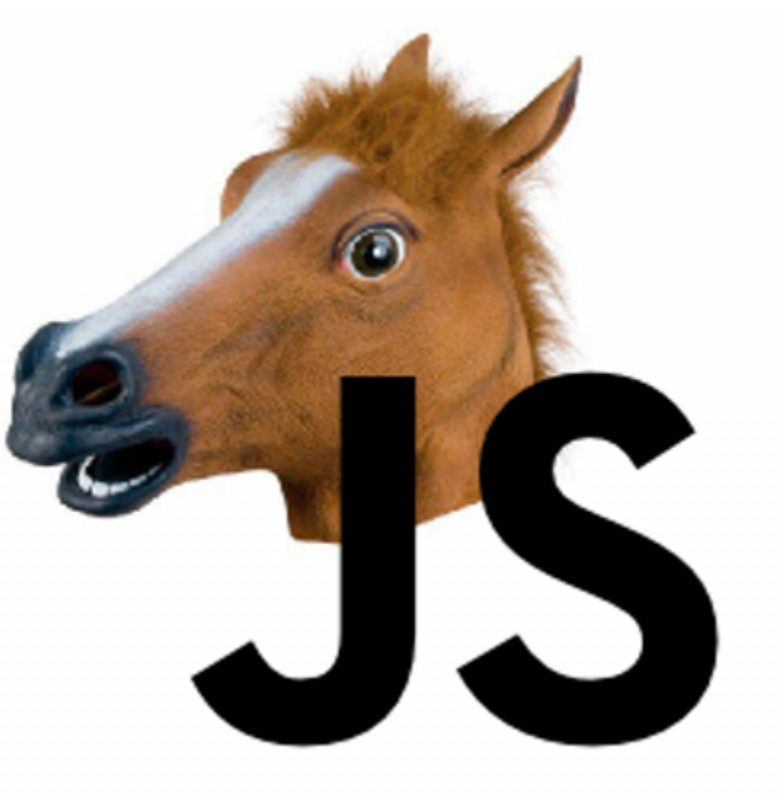
LOGIC IN DB: CONTRA

- Overload database
- No good pracitces (TDD, modules etc)
- Archaic languages
- Slow development

UNRESOLVABLE?

- +modern language
- +modularity
- +tooling

WHICH LANG?



- everywhere (interop)
- everyone knows
- fast

- most misunderstood language
- last programming language
- githut
- 2015

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

PLV8: FUNCTIONS

```
CREATE FUNCTION plv8_test(keys text[], vals text[])
RETURNS json AS $$
  var obj = {};
  for(var i=0; i<keys.length; i++){
      obj[keys[i]] = vals[i];
  }
  return obj;
$$ 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
$$
    plv8.elog(NOTICE, JSON.stringify(NEW));
    plv8.elog(NOTICE, JSON.stringify(OLD));
    plv8.elog(NOTICE, TG_OP);
    plv8.elog(NOTICE, TG_ARGV);
    if (TG_OP == UPDATE) {
        NEW.i = 102;
        return NEW;
    }
$$ 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;
```

PLV8: BENCH

alg	sql	v8	pgsql
noop	1	3.9	5.0
add	1	3.1	4.8
str	~	1	1.8
get key	1	4.4	2.8
iter	~	1	1.2
exec	~	1.2	1
polynom	~	1	200

improve

NOT ONE LANGUAGE

- CoffeeScript
- TypeScript
- ClojureScript
- PureScript
- pgsql :)
- your?

JSX -> JSQL

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});
// why not
var x = 5;
(SELECT .* FROM users where id = x)
.map(function(u){
   return u.name;
})
```

WE NEED MORE

- + modularity
- + tooling

Event-driven I/O server-side
JavaScript environment based on V8

PG + PLV8 + NODEJS?

PG.JS: CONCEPT

- write in node
- compile into plv8

PG.JS: MOCK PLV8

```
Client = require('pg-native')
client = new Client
client.connectSync(conn_string)

global.INFO="INFO"
module.exports =
  execute: = ->
    client.querySync.apply(client, arguments).map(coerse)
  elog: (x, msg) ->
    console.log "#{x}:", msg
  quote_literal: (str)->
    str && client.pq.escapeLiteral(str)
  quote_ident: (str)->
    str && client.pq.escapeIdentifier(str)
```

PG.JS: WRITE IN NODE

```
util = require('./util')
uuid = (plv8)->
  plv8.execute('select gen_random_uuid() as uuid')[0].uuid

exports.uuid = uuid
create = (plv8, resource)->
  table_name = util.table_name(resource_type)
  # ...
  json  = JSON.stringify(resource)
  plv8.execute """
    INSERT INTO #{table_name}
    (id, version_id, content)
    VALUES ($1,$2,$3)
    """, [id, version_id, json]
  resource
```

PG.JS: TEST IN NODE

```
plv8 = require('../lib/plv8')
crud = require('../src/crud')
schema = require('../src/schema')

describe "CRUD", ()->
  beforeEach ()->
    schema.generate_table(plv8, 'Patient')

  it "read", ()->
    pt = {resourceType: 'Patient', name: {text: 'Albert'}}
    pt_created = crud.create(plv8, pt)
    expect(pt_created.id).toBeTruthy()
    expect(pt_created.meta.versionId).toBeTruthy()
```

PG.JS: COMPILE INTO PLV8

```
Module = require("module")
oldrequire = Module::require
Module::require = (fl) ->
  currentModule = fl
  oldrequire.apply this, arguments

oldcompile = Module::_compile
Module::_compile = (answer, filename) ->
  for k,v of @exports when v.plv8?
      plv8_exports[k] ={fn: v, filename: filename}
```

github

PG.JS: COMPILE INTO PLV8

```
CREATE OR REPLACE FUNCTION #{def_fn} AS $$
var deps = {}
var cache = {}
#{modules_js}
var require = function(dep){
  if(!cache[dep]) {
    var module = {exports: {}};
    deps[dep](module, module.exports, require);
    cache[dep] = module.exports;
  }
  return cache[dep]
}
return require('#{mod}').#{k}#{def_call};
$$ LANGUAGE plv8 IMMUTABLE STRICT;
```

PG.JS: CALL IN POSTGRES

```
select fhir.read('StructureDefinition', 'Patient') as read
```

EXPERIMENTS ON GITHUB

- pgpp
- plpl
- plpl-sampl

PG.JS: ROAD MAP

- remove death code (Google closure comp)
- extend plv8 - require, native fn call,....
- deploy
- apply to fhirbase

THX

Q?