EtherCIS

A Pragmatic Open Source openEHR CDR

Christian Chevalley

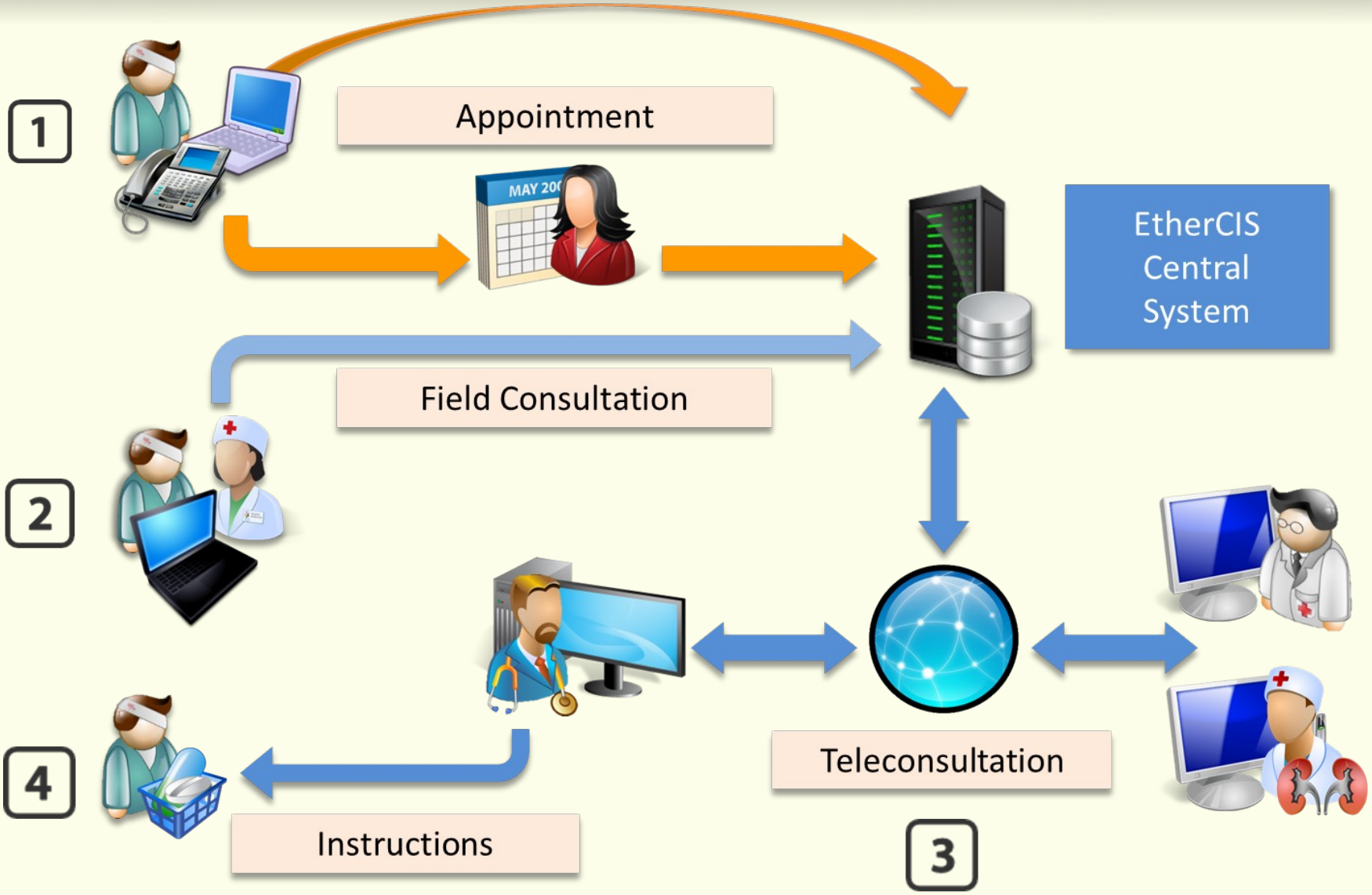
Ripple Foundation
openEHR Foundation
ADOC Software Development

WARNING: This is a Microsoft Free Presentation

Objectives

- Centralized clinical data: direct input, import + export
- Production of mandatory KPIs and reports
- Strong security/privacy in compliance to national regulation
- Discreet access to data according to patients' informed consent
- Action/event coordination (workflow, scheduler)
- Telemedicine

Synoptic


1
Appointment
MAY 20
EtherCIS
Central
System
Field Consultation
2
Teleconsultation
3
4
Instructions

Example KPIs

- KPIs are mandatory to claim subsidies
- Example: DRS & Follow-up Indicators:
  - Population coverage
  - Medical: nb patients without, mild/moderate/severe NPDR/DME etc.
  - Quality MD OPTH: Nb evaluating FO/DRS, Nb > 250 FO/DRS etc.
  - Quality Nurses: Nb patients w/bad fundus, w/tropicamid instillation, w/stereoscopic photos etc.
  - Management Nurses: Nb patients w/no problem, difficult/impossible

What is an openEHR backend?

- Create EHR
- Compositions & Contributions CRUD
- Query CDR (AQL)
- Manage Template & Archetypes
- Validate data w/Template constraints
- Logging/auditing
- Segregate Demographics & Clinical Data
- Expose resources with a REST API (but not only!)

EtherCIS Principles

- It’s All About the Database
  - First Trials: Domain → DB (ORM)
  - EtherCIS: DB → Domain (SQL)

    → The DB survives the application

    → Reading/Querying is the issue

    → Throughput/optimization
- jOOQ: Java Object Oriented (SQL) Query
- Benefits
  - Support modern SQL and Extensions (SQL:2016)
  - Referential Integrity
  - PostgreSQL: jsonb denormalization (not schema-less!) BUT no referential integrity yet here…
- Service Oriented Architecture (openEHR SM)
- REST API
- Knowledge Cache

Why Using a *traditional* DB Engine ?

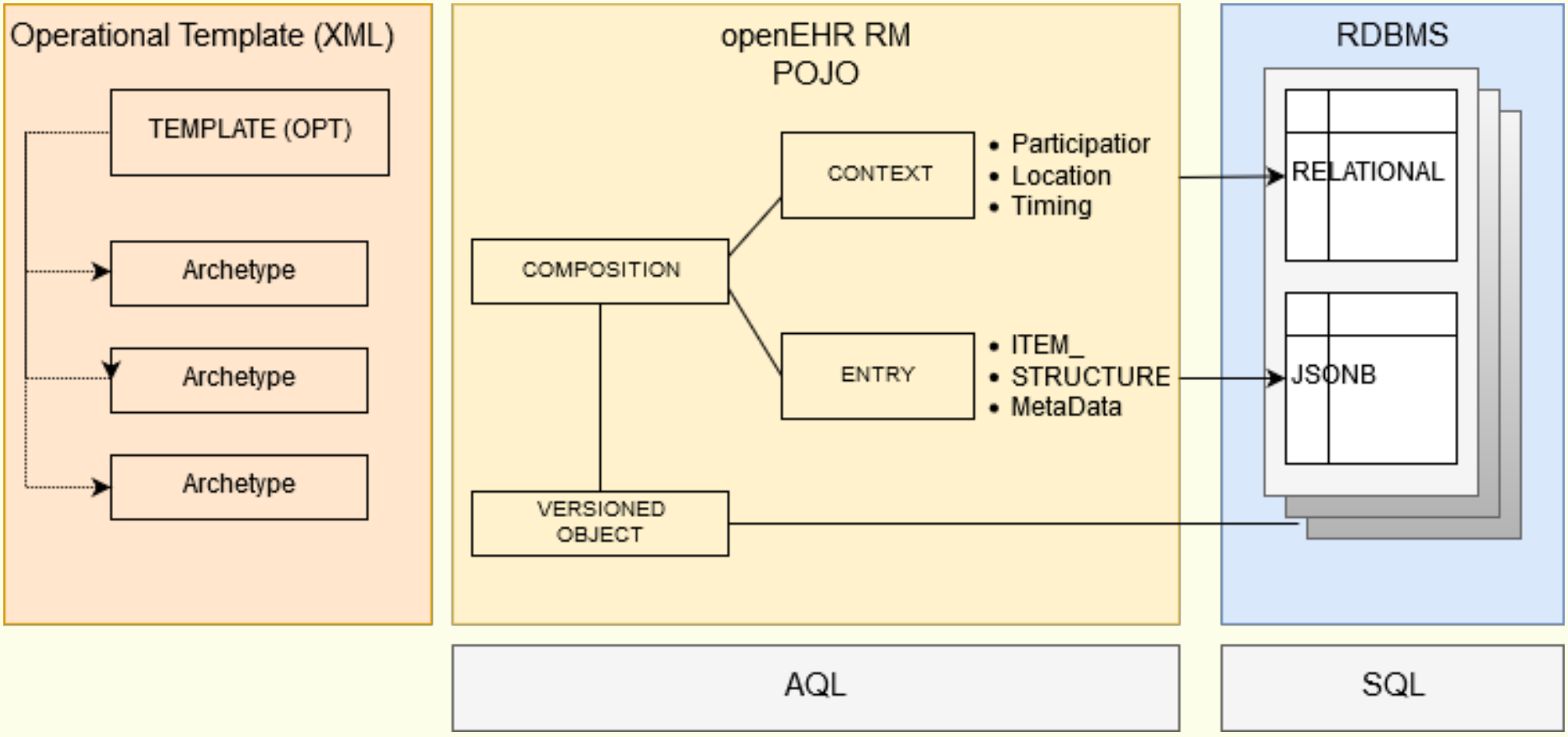
- DB DNA/Ecosystem
  - Analytics tools
  - Backup/recovery
  - External Data integration
  - Query optimization
  - Data management
  - Hardware/Software coupling
  - Monitoring
- ACID (MongoDB's "Eventual Consistency").

Modern SQL ?

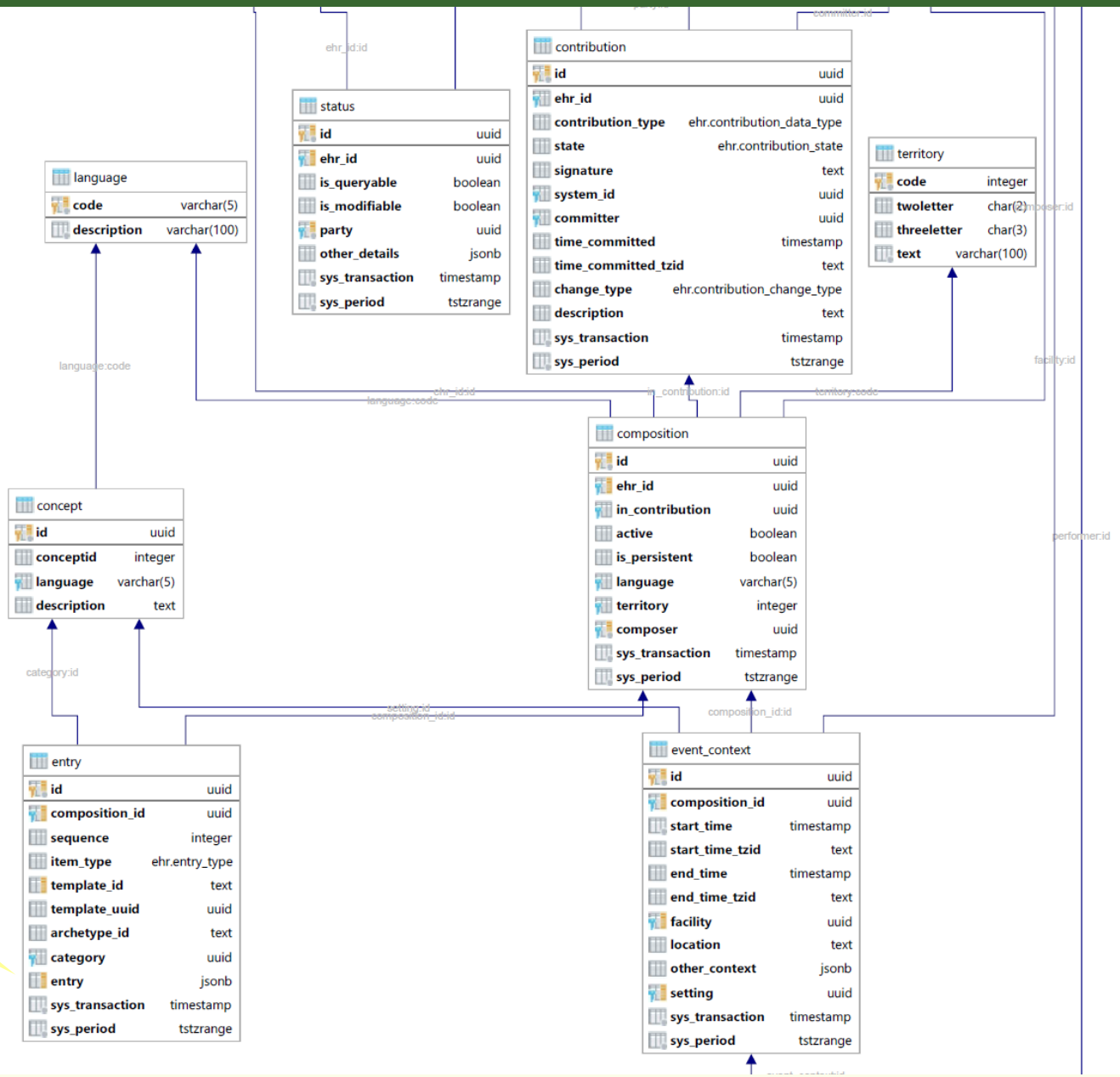
- LATERAL joins (for-each SQL)
- Grouping Sets (instead of UNION ALL)
- CTEs
- Recursive CTEs
- Partitioning (OVER)
- JSON
- Temporal Tables

See Markus Winand’s Blog (modern-sql.com)

Projections



DB Structure



ITEM STRUCTURE

ITEM STRUCTURE Persistence

```
"/data[at0003]": {
  "/items[at0004]": [
    {
      "/name": {
        "value": "Systolic"
      },
      "/value": {
        "units": "mm[Hg]",
        "accuracy": 0.0,
        "magnitude": 135.0,
        "precision": 0,
        "accuracyPercent": false
      },
      "/$PATH$": "/content[openEHR-EHR-SECTION.vital_signs.v1 and
name/value='Vital signs']/items[openEHR-EHR-OBSERVATION.blood_pressure.v1 and
name/value='Blood Pressure']/data[at0001]/events[at0006 and name/value='any
event']/data[at0003]/items[at0004 and name/value='Systolic']",
      "/$CLASS$": "DvQuantity"
    }
  ],
```

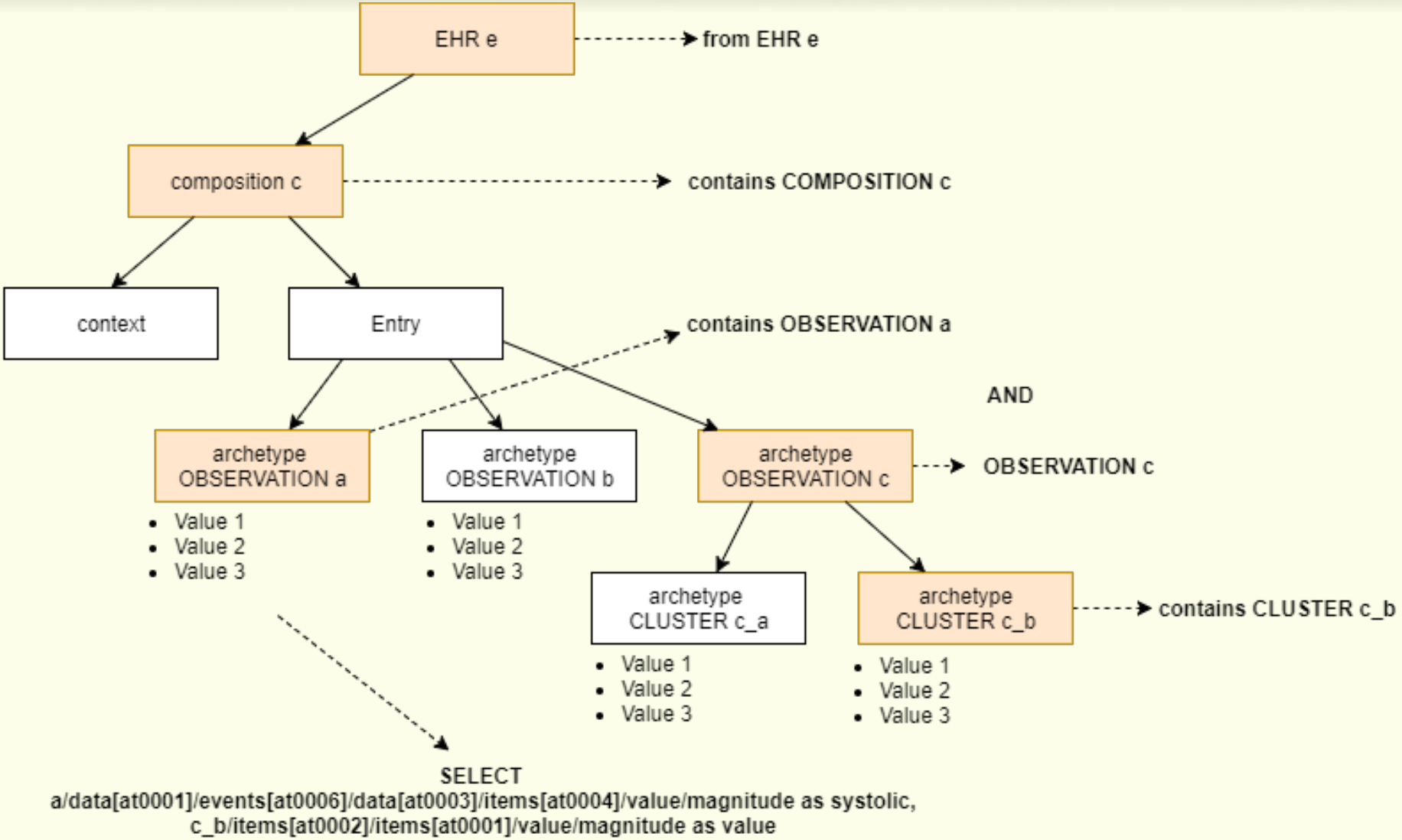
Persistence

- PostgreSQL supports the combining Tables and Document (json) data
- Can be manipulated/queried within the same DB engine
- Statics => Table: EHR, EVENT_CONTEXT, CONTRIBUTION, COMPOSITION, PARTY_IDENTIFIED etc.
  - Referential Integrity
  - Joins
- Archetyped (ITEM_STRUCTURE): jsonb
  - Composition.ENTRY

More Persistence

- Versioning:
  - Bi Temporal Tables:
    - application period
    - Transaction time
  - Performed via Triggers on CUD
- CONTAINS clause resolution
  - Nested Sets: ‘ltree’
  - Build an index of archetype path depending on Templates
  - Supports partial search (wildcard)

Querying


EHR e
from EHR e
composition c
contains COMPOSITION c
context
Entry
contains OBSERVATION a
AND
archetype
OBSERVATION a
archetype
OBSERVATION b
archetype
OBSERVATION c
OBSERVATION c
Value 1
Value 2
Value 3
Value 1
Value 2
Value 3
archetype
CLUSTER c_a
archetype
CLUSTER c_b
contains CLUSTER c_b
Value 1
Value 2
Value 3
Value 1
Value 2
Value 3
SELECT
a/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/magnitude as systolic,
c_b/items[at0002]/items[at0001]/value/magnitude as value

AQL Processing

- Compiler AQL → SQL (postgresql)
- Sequence
  - Parse expression (ANTLR) → AST
  - Pass 1
    - Map CONTAINS expressions into nested sets
  - Pass 2
    - Map all SELECT variables, aggregation etc.
    - Identify ORDER BY, TOP etc.
  - Bind to SQL (PostgreSQL)
    - Resolve variable paths → use CONTAINMENT table

AQL Select BP

```
select a/uid/value as uid,
    a/context/start_time/value as date_created,
    o_bp/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/magnitude as systolic,
    o_bp/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/units as systolic_units,
    o_bp/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value/magnitude as diastolic,
    o_bp/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value/units as diastolic_units
    from EHR e
    contains COMPOSITION a contains OBSERVATION o_bp[openEHR-EHR-OBSERVATION.blood_pressure.v1]
    where a/context/start_time/value >= '2016-01-01T00:00:00'  and a/context/start_time/value <=
'2016-09-31T00:00:00'
    and e/ehr_status/subject/external_ref/id/value  = '99999-1234'
    and e/ehr_status/subject/external_ref/namespace = 'testIssuer'
    ORDERBY date_created ASC
```

SQL Select BP

```
select
  "ehr"."comp_expand"."composition_id"||'::'||'test-server'||'::'||(
  select (count(*) + 1)
  from "ehr"."composition_history"
  where "ehr"."composition_history"."id" = 'fadbd0d2-d3e0-4f06-bfa8-3e304dc231d7'
) as "uid",
  to_char("ehr"."comp_expand"."start_time",'YYYY-MM-DD"T"HH24:MI:SS')||"ehr"."comp_expand"."start_time_tzid" as "date_created",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],0,/items[openEHR-EHR-
OBSERVATION.blood_pressure.v1],0,/data[at0001],/events,/events[at0006],0,/data[at0003],/items[at0004],0,/value,magnitude}' as
"systolic",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],0,/items[openEHR-EHR-
OBSERVATION.blood_pressure.v1],0,/data[at0001],/events,/events[at0006],0,/data[at0003],/items[at0004],0,/value,units}' as
"systolic_units",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],0,/items[openEHR-EHR-
OBSERVATION.blood_pressure.v1],0,/data[at0001],/events,/events[at0006],0,/data[at0003],/items[at0005],0,/value,magnitude}' as
"diastolic",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],0,/items[openEHR-EHR-
OBSERVATION.blood_pressure.v1],0,/data[at0001],/events,/events[at0006],0,/data[at0003],/items[at0005],0,/value,units}' as
"diastolic_units",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],1,/items[openEHR-EHR-
OBSERVATION.heart_rate.v1],0,/data[at0002],/events,/events[at0003],0,/data[at0001],/items[at0004],0,/value,magnitude}' as
"rate",
  "ehr"."comp_expand"."entry"->(select json_object_keys("ehr"."comp_expand"."entry"::json)) #>> '{/content[openEHR-EHR-
SECTION.simple_section_name.v1],1,/items[openEHR-EHR-
OBSERVATION.heart_rate.v1],0,/data[at0002],/events,/events[at0003],0,/data[at0001],/items[at0004],0,/value,units}' as
"rate_units"
from "ehr"."comp_expand"
where "ehr"."comp_expand"."composition_id" = 'fadbd0d2-d3e0-4f06-bfa8-3e304dc231d7'
```

Validation

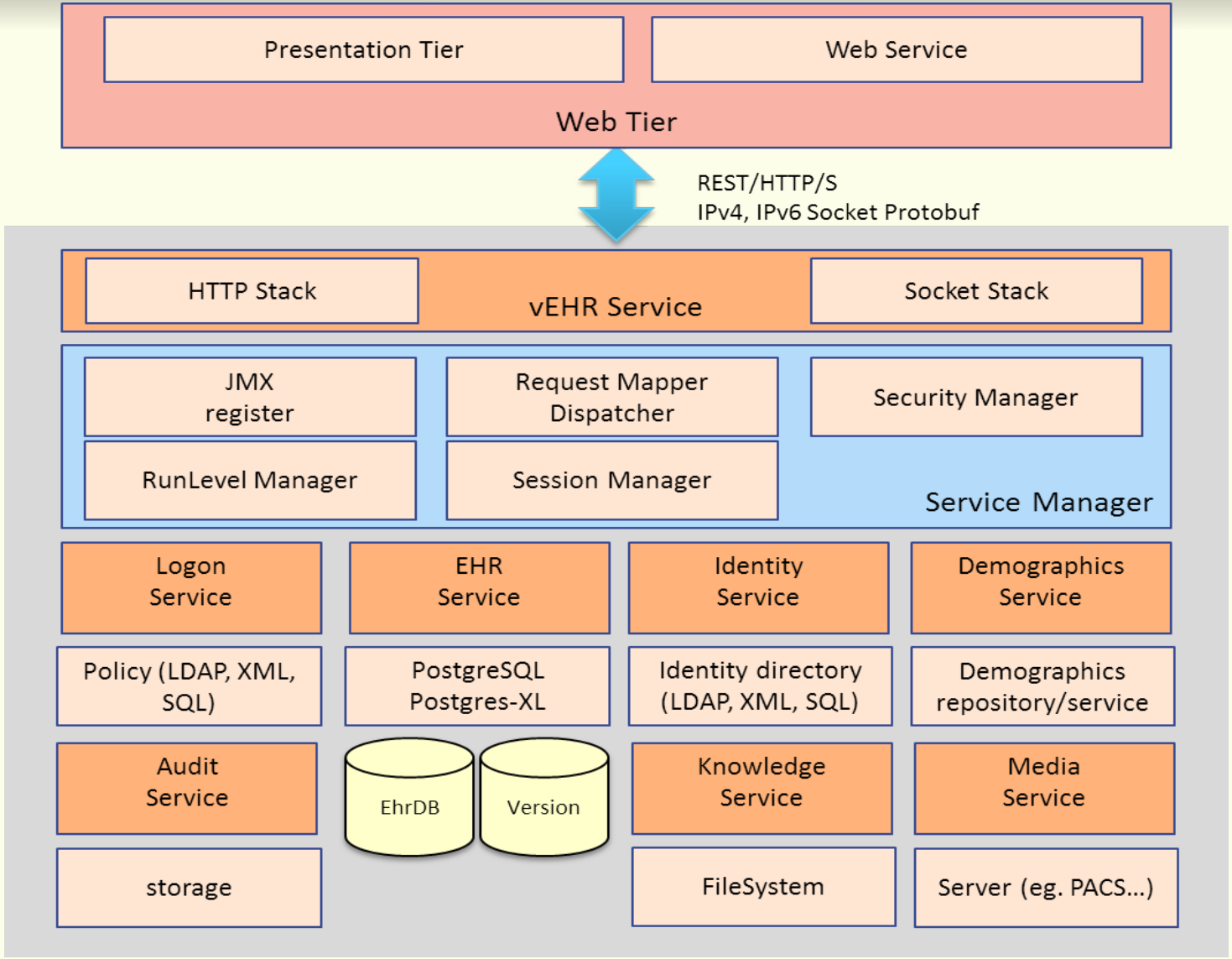
- Specified by AOM
  http://www.openehr.org/releases/AM/latest/docs/AOM2/AOM2.html
- Path based
- Cache Maps and Lists of constraints per Template
  - Type specific constraint (validate units f.e.)
  - Cardinality
  - Existence/Not Allowed
  - Valid path
- Invoked whenever a composition is

Template Constraints

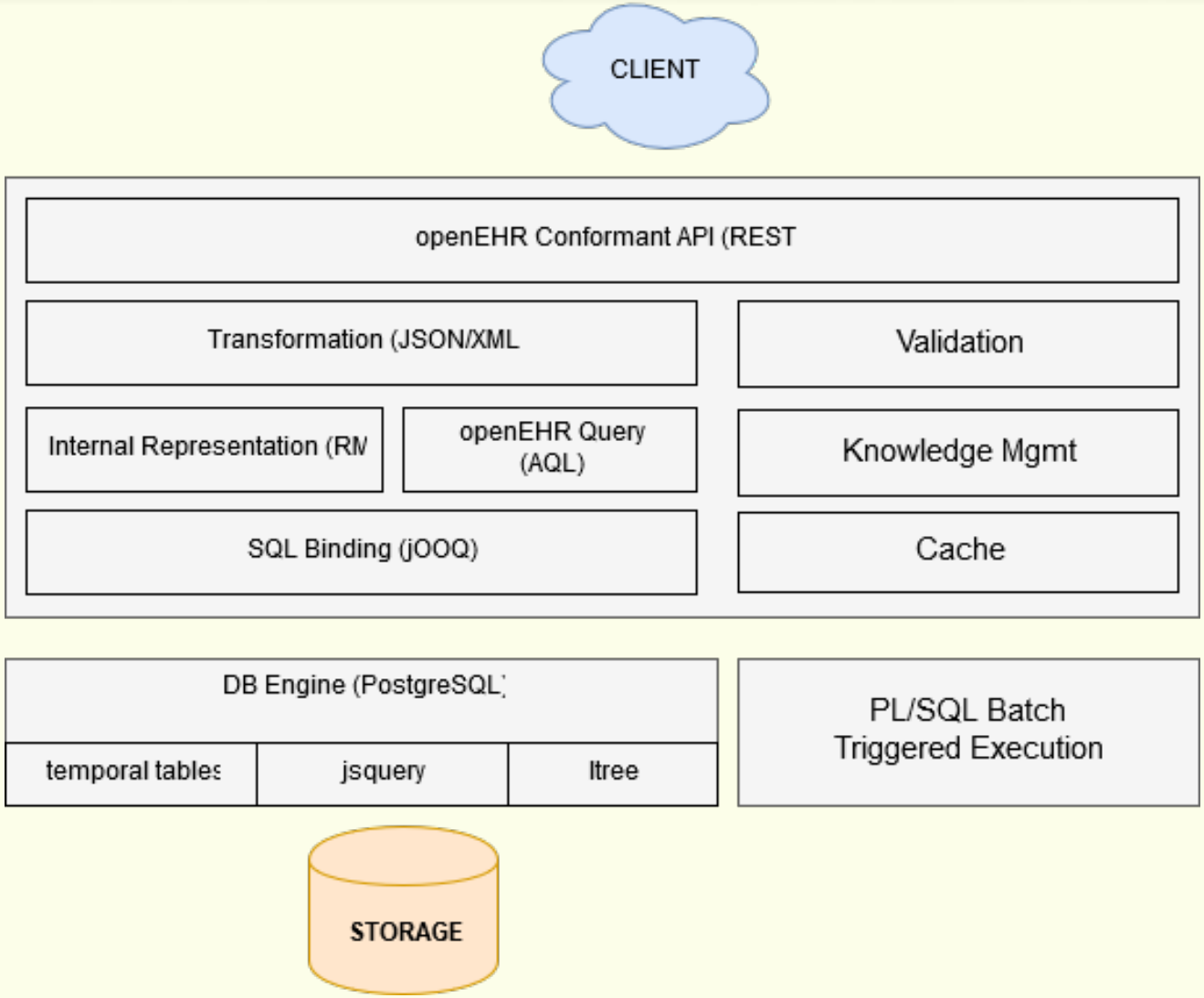
```
 <v1:rm_type_name>ELEMENT</v1:rm_type_name>
  <v1:occurrences>
    <v1:lower_included>true</v1:lower_included>
    <v1:upper_included>true</v1:upper_included>
    <v1:lower_unbounded>false</v1:lower_unbounded>
    <v1:upper_unbounded>false</v1:upper_unbounded>
    <v1:lower>0</v1:lower>
    <v1:upper>1</v1:upper>
  </v1:occurrences>
  <v1:node_id>at0004</v1:node_id>
  <v1:attributes xsi:type="C_SINGLE_ATTRIBUTE">
...
  <v1:children xsi:type="C_COMPLEX_OBJECT">
  <v1:rm_type_name>DV_QUANTITY</v1:rm_type_name>
  <v1:occurrences>
    <v1:lower_included>true</v1:lower_included>
    <v1:upper_included>true</v1:upper_included>
    <v1:lower_unbounded>false</v1:lower_unbounded>
    <v1:upper_unbounded>false</v1:upper_unbounded>
    <v1:lower>1</v1:lower>
    <v1:upper>1</v1:upper>
  </v1:occurrences>
  <v1:node_id/>
  <v1:property>
    <v1:terminology_id>
      <v1:value>openehr</v1:value>
    </v1:terminology_id>
    <v1:code_string>382</v1:code_string>
  </v1:property>
  <v1:list>
    <v1:magnitude>
      <v1:lower_included>true</v1:lower_included>
      <v1:lower_unbounded>false</v1:lower_unbounded>
      <v1:upper_unbounded>true</v1:upper_unbounded>
      <v1:lower>0</v1:lower>
    </v1:magnitude>
    <v1:precision>
...
    </v1:precision>
    <v1:units>/min</v1:units>
  </v1:list>
```

www.openehr.org/releases/1.0.2/architecture/terminology.pdf
ConceptId 382 = frequency

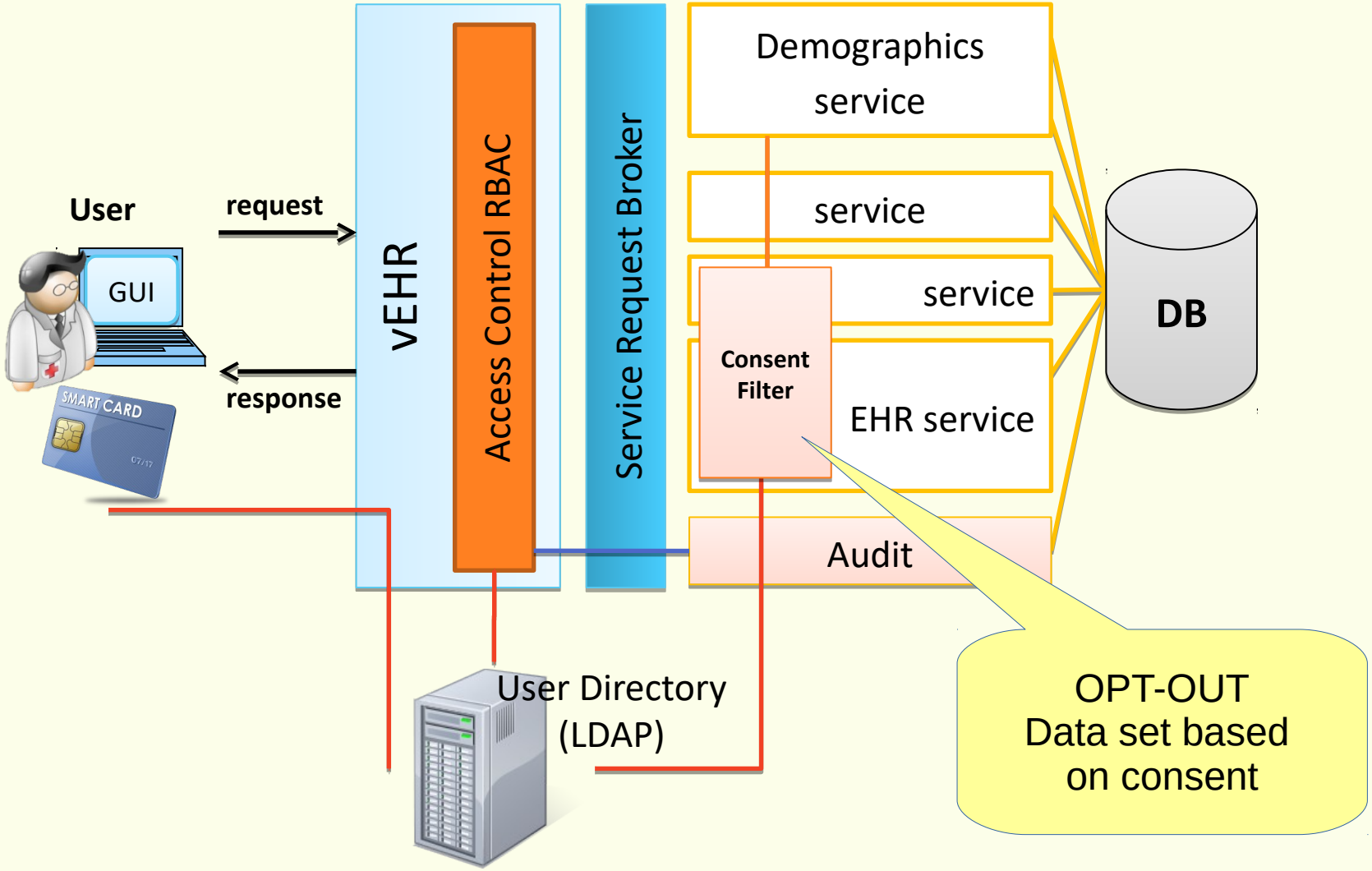
OpenEHR SM: vEHR



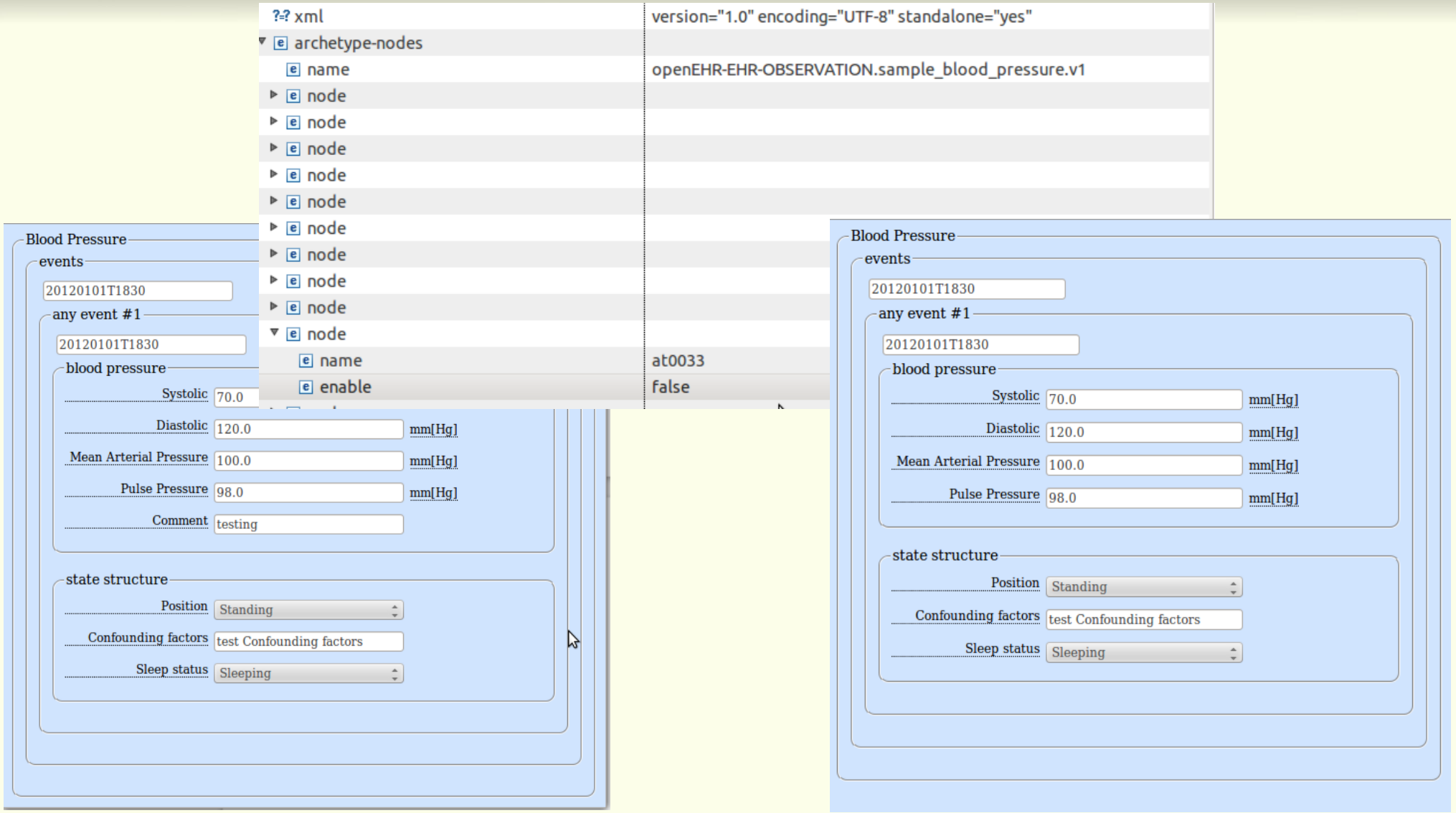
EtherCIS Internals



Data Access Control



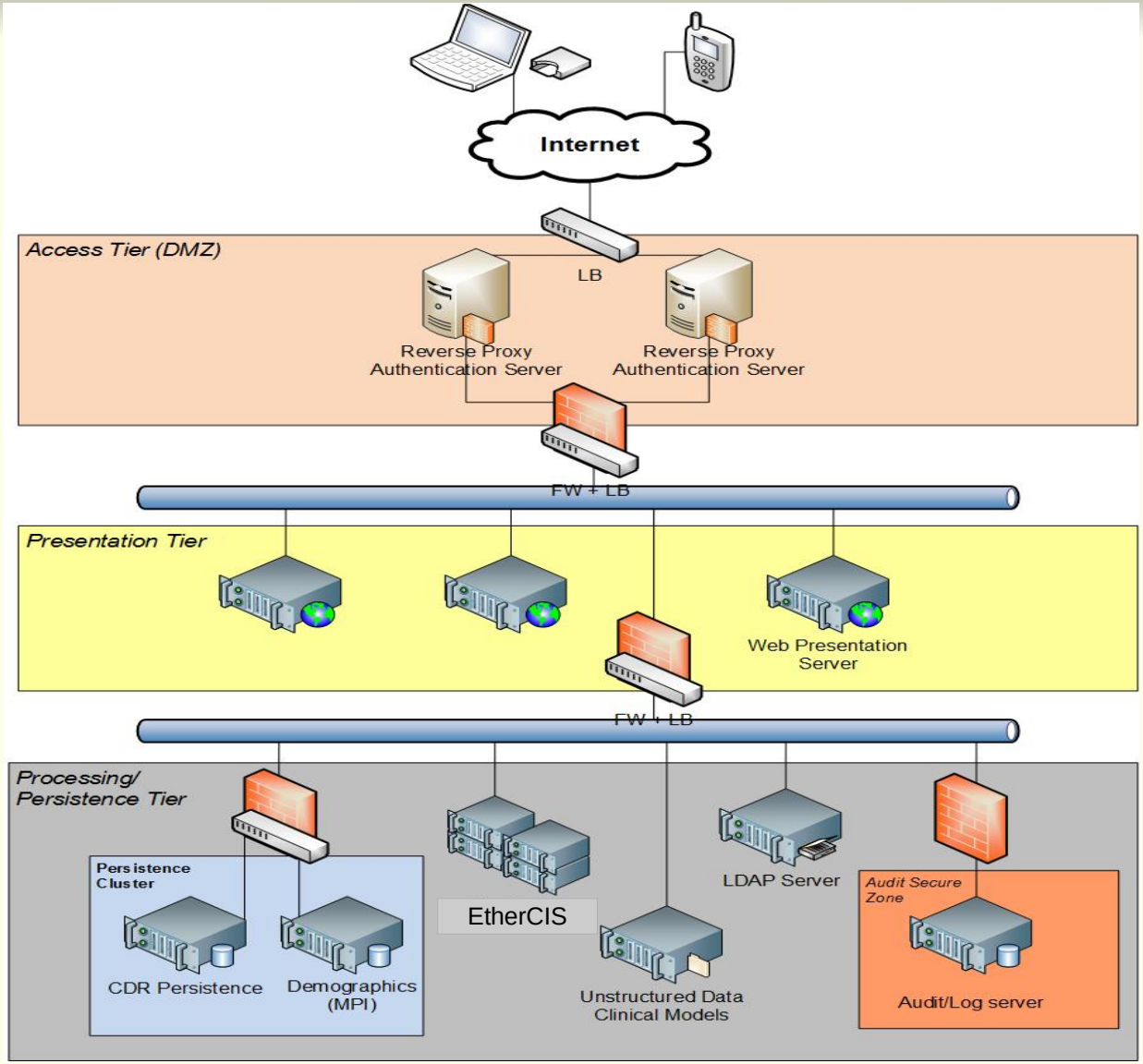
Consent Based Data Access



More Info?

- EtherCIS http://ethercis.org/
- Ripple http://ripple.foundation/
- OpenEHR http://www.openehr.org/
- EtherCIS code base
  https://github.com/ethercis

Deployment


Internet
Access Tier (DMZ)
LB
Reverse Proxy
Authentication Server
Reverse Proxy
Authentication Server
FW + LB
Presentation Tier
Web Presentation
Server
FW + LB
Processing/
Persistence Tier
Persistence
Cluster
CDR Persistence
Demographics
(MPI)
EtherCIS
Unstructured Data
Clinical Models
LDAP Server
Audit Secure
Zone
Audit/Log server