Parse, persist and query HL7 CDA documents

Nov 2011 – RIMBAA meeting Amsterdam

Henk Enting <h.d.enting@mgrid.net>
Overview

Introduction
Recap
CDA parsing
About MGRID

- Spin off of Portavita in 2007
- Goal: create a storage layer specifically designed for eHealth applications
- Funding in 2009 by T-Systems Venture Fund
- Release 1.0 in October 2010
Recap
Recap

- October 2009: Native support for HL7 datatypes
- February 2011: Inheritance
- Today: CDA parser
Recap - datatypes

October 2009: native support for HL7 datatypes

February 2011: inheritance
RIM database model generation

Today: CDA parser
Native support for ISO datatypes

- Enable precise database mapping of HL7v3 artefacts
  - no locally added constructs or workarounds, no caveats
  - knowing HL7v3 means knowing the database

- Create a query language that is powerful, fast and easy to learn
  - *powerful* query language – SQL & all PL/ languages
  - *fast*; most datatypes support indexes
  - *easy* to learn; the application programmer now has powerful, intuitive primitives
PQs used to document observations

Based on Unified Code for Units of Measure
- 294 units – a.o. units from SI, ISO 1000, ISO 2955, ANSI X3.50, CGS, unified U.S. & British Imperial units

Operations supported:
- Comparison: <, > and friends
- Arithmetic: +, −, /, *, power
- Aggregation: min, max, avg, sum, var, stddev

Indexable
```sql
CREATE TABLE patient (name text, height pq, weight pq);

INSERT INTO patient
VALUES
    ('Jack', '1.92 m', '92 kg'),
    ('Julia', '150 cm', '50 kg'),
    ('John', '188 cm', '84.3 kg'),
    ('Luke', '78 cm', '11800 g');

CREATE FUNCTION bmi(height pq, weight pq)
RETURNS pq
AS $$
    SELECT convert($2, 'kg') / convert($1, 'm')^2;
$$
LANGUAGE SQL IMMUTABLE;

SELECT *, bmi(height, weight) FROM patient WHERE height > '1.70 m'
ORDER BY weight;
```

```
<table>
<thead>
<tr>
<th>name</th>
<th>height</th>
<th>weight</th>
<th>bmi</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>188 cm</td>
<td>84.3 kg</td>
<td>23.8512901765504753 kg/m²</td>
</tr>
<tr>
<td>Jack</td>
<td>1.92 m</td>
<td>92 kg</td>
<td>24.9565972222222222 kg/m²</td>
</tr>
</tbody>
</table>

(2 rows)
```
create table patient (name text, height pq, weight pq);
CREATE TABLE

insert into patient values
  ('Jack', '1.92 m', '92 kg'),
  ('Julia', '150 cm', '50 kg'),
  ('John', '188 cm', '84.3 kg'),
  ('Luke', '78 cm', '11800 g');
INSERT 0 4

create or replace function bmi(height pq, weight pq)
returns pq
as $$
  select convert($2, 'kg') / convert($1, 'm')^2;
$$
language sql immutable;
CREATE FUNCTION

select *, bmi(height, weight) from patient where height > '1.70 m'
order by weight;
name | height | weight | bmi
+------|--------+---------+-------------------------+
John  | 188 cm | 84.3 kg | 23.8512901765504753 kg/m2
Jack  | 1.92 m | 92 kg   | 24.9565972222222222 kg/m2
(2 rows)

/* And now for something completely different:
* what is the mean travel time of light, from the sun to the earth?
*/

```sql
select convert(pq '1 AU' / '[c]', 's');
```

```
convert
------------------------
499.0047838061356433 s
(1 row)
```
Controlled vocabularies in medical informatics
- record information unambiguously
- allow machine reasoning

HL7v3 Coded Value implementation

Support for a large number of codesystems:
- **Systemized Nomenclature of Medicine – Clinical Terms**
- HL7v3 vocabularies all Editions
- **Logical Observation Identifiers Names and Codes**

Supports code systems with hierarchies

Indexable
ISO datatypes: CV
Fracture of bone

- Fracture of humerus
- Fracture of femur

Injury of lower extremity

- Leg sprain
ISO datatypes: CV

```sql
select name, code(disorder), codesystemname(disorder), displayname(disorder) from observation;
```

<table>
<thead>
<tr>
<th>name</th>
<th>code</th>
<th>codesystemname</th>
<th>displayname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Willem</td>
<td>71620000</td>
<td>SNOMED-CT</td>
<td>Fracture of femur</td>
</tr>
<tr>
<td>Yeb</td>
<td>66308002</td>
<td>SNOMED-CT</td>
<td>Fracture of humerus</td>
</tr>
<tr>
<td>Henk</td>
<td>262994004</td>
<td>SNOMED-CT</td>
<td>Leg sprain</td>
</tr>
</tbody>
</table>

(3 rows)

```sql
select name, displayname(disorder) from observation
where disorder << '284003005|Fracture of bone'::cv('SNOMED-CT');
```

<table>
<thead>
<tr>
<th>name</th>
<th>displayname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Willem</td>
<td>Fracture of femur</td>
</tr>
<tr>
<td>Yeb</td>
<td>Fracture of humerus</td>
</tr>
</tbody>
</table>

(2 rows)

```sql
select name, displayname(disorder) from observation
where disorder << '127279002|Injury of lower extremity'::cv('SNOMED-CT');
```

<table>
<thead>
<tr>
<th>name</th>
<th>displayname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Willem</td>
<td>Fracture of femur</td>
</tr>
<tr>
<td>Henk</td>
<td>Leg sprain</td>
</tr>
</tbody>
</table>

(2 rows)
Recap - Inheritance

- October 2009: Native support for HL7 datatypes
- February 2011: Inheritance RIM database model generation
- Today: CDA parser
Inheritance - generated model
Inheritance - example

```
henk@henk-laptop:~ $ psql rim219 -U mgrid
Type "help" for help.

```rim219#$ select * from "Act";
```
Recap - Inheritance

October 2009: Native support for HL7 datatypes

February 2011: Inheritance RIM database model generation

Today: CDA parser
Objective:
- Parse HL7 CDA documents
- Persist them in a RIM database
- Query the database for reporting

Focus today: parse and persist
Step 1: create the database

CDA_R2_Discharge_Summary.xml

HL7v3 Edition2005 core, mif, DT XSDs

- HL7 standard definitions
- generated code/db
- manual coded by MGRID
- XML input data

XSL Transforms, HL7v3 datatypes

- pure HL7v3 R/M Database:
  * vocabulary checks
  * context conduction
Step 2: create the parser

- CDA R2 XSD
- XML Parser Generator
- CDA Parser
- HL7v3 Edition2005 core/ml, DT XSDs
- XSL Transforms, HL7v3 datatypes
- pure HL7v3 R/M Database:
  * vocabulary checks
  * context conduction
Step 3: determine the mappings

CDA R2 MIF → XSL Transforms → MIF class maps, constrained attributes → RIM class and datatype definitions → Rim analysis XSL Transforms → HL7v3 Edition2005 coremif, DT XSDs

CDA R2 XSD → XML Parser Generator → CDA Parser

CDA_R2_Dischage_Summary.xml

HL7 standard definitions
- generated code / db
- manual coded by MGRID
- XML input data

pure HL7v3 RIM Database:
- vocabulary checks
- context conduction
Step 4: combine parser with mapper

Converting MIF instances into SQL using model driven design
Example CDA

- CDA R2 test instances provided by Tiani SPIRIT

```xml
<?xml version="1.0" encoding="utf-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
  <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3"/>
  <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.1"/>
  <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.2"/>
  <templateId extension="7d062328-1f36-102c-b83a-000c2915b919" root="1.2.40.0.32.6.1.10.1.1"/>
  <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.4"/>
  <id extension="20E8558A-9A0C-2A02-5611-0944F9DEE4F2" root="2.16.17.710.777.1001.902.1.1.3.2"/>
  <code code="18842-5" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="SUMMARIZATION OF EPISODE NOTE"/>
  <title>Discharge Summarization</title>
  <effectiveTime value="20091028102753+0200"/>
  <confidentialityCode code="N" codeSystem="2.16.840.1.113883.5.25" codeSystemName="Confidentiality" displayName="Normal"/>
  <languageCode code="en-ZA"/>
  <setId root="2.16.17.710.777.1001.902.1.1.3.2"/>
  <versionNumber value="1"/>
  <recordTarget>
    <patientRole>
      <id extension="005.12567212651421" root="1.3.6.1.4.1.21367.2009.2.2.795"/>
      <addr>
        <country>AUSTRIA</country>
        <state>Wien</state>
        <city>Wien</city>
        <postalCode>1010</postalCode>
        <streetName/>
      </addr>
      <telecom use="HP" value="tel:+43123456789"/>
      <telecom use="WP" value="tel:+43198765432"/>
    </patient>
    <name>
```
### CDA persisted in MGRID

<table>
<thead>
<tr>
<th>relname</th>
<th>reltuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientEncounter</td>
<td>1</td>
</tr>
<tr>
<td>SubstanceAdministration</td>
<td>3</td>
</tr>
<tr>
<td>Organization</td>
<td>4</td>
</tr>
<tr>
<td>Patient</td>
<td>1</td>
</tr>
<tr>
<td>Observation</td>
<td>10</td>
</tr>
<tr>
<td>Person</td>
<td>4</td>
</tr>
<tr>
<td>Document</td>
<td>1</td>
</tr>
<tr>
<td>ManufacturedMaterial</td>
<td>3</td>
</tr>
<tr>
<td>ActRelationship</td>
<td>39</td>
</tr>
<tr>
<td>Act</td>
<td>25</td>
</tr>
<tr>
<td>Participation</td>
<td>9</td>
</tr>
<tr>
<td>Entity</td>
<td>1</td>
</tr>
<tr>
<td>Role</td>
<td>8</td>
</tr>
</tbody>
</table>

(13 rows)
Query on persisted CDA

- List of medication administrations per person:
In this case: participations
Performed on insert time
Less joins necessary, so
  more concise query
  better performance

<table>
<thead>
<tr>
<th>relname</th>
<th>reltuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientEncounter</td>
<td>1</td>
</tr>
<tr>
<td>SubstanceAdministration</td>
<td>3</td>
</tr>
<tr>
<td>Organization</td>
<td>4</td>
</tr>
<tr>
<td>Patient</td>
<td>1</td>
</tr>
<tr>
<td>Observation</td>
<td>10</td>
</tr>
<tr>
<td>Person</td>
<td>4</td>
</tr>
<tr>
<td>Document</td>
<td>1</td>
</tr>
<tr>
<td>ManufacturedMaterial</td>
<td>3</td>
</tr>
<tr>
<td>ActRelationship</td>
<td>39</td>
</tr>
<tr>
<td>Act</td>
<td>25</td>
</tr>
<tr>
<td>Participation</td>
<td>123</td>
</tr>
<tr>
<td>Entity</td>
<td>1</td>
</tr>
<tr>
<td>Role</td>
<td>8</td>
</tr>
</tbody>
</table>

(13 rows)
Query with context conduction
test=# select displayname('7947003'::CV('SNOMED-CT'));
    displayname
              ------
             Aspirin

---

test=# select displayname(code) from "ManufacturedMaterial"
     where (code) << '7947003'::cv('SNOMED-CT');
    displayname
              ---------------------
             Aspirin 300mg soluble tablet
(1 row)
Next steps

CDA_R2_Discharge_Summary.xml → SQL translator → pure HL7v3 RIM Database:
- vocabulary checks
- context conduction

→ Datawarehouse:
- customized schema for the research queries
- use of HL7 datatypes only if wanted

→ Report Generation
  → Feature Extraction
  → Reports

From documents to reports: The SQL translator is part of a model driven architecture tool suite that enables clinical data integration into a single generic schema, which can be used to populate a datawarehouse in order to generate business reports.
Questions?