Introducing the MGRID HL7 Datatypes

Dijkstra | Enting | Havinga

Schiphol Airport, 27 October 2009
Today’s agenda

Introduction
  Requirements on Medical IT
  The Datatype Gap

MGRID Solution

Datatype examples
  PQ
  PQ & TS flavours
  IVL<T>
  II
  CV
  CV / SNOMED CT

ORM Example

Questions
Requirements on Medical IT

- Support growing information needs
  - without increasing costs
  - without decreasing performance

- Support interactive queries
  - Support online transaction processing
  - Information is used to guide treatment, archival becomes secondary

- Support different and changing information requirements
  - Storage should not be optimized for one particular question, but for all
  - Use standards for annotation, storage and query

The Datatype Gap

- Database datatypes are far removed from HL7 datatypes
  - fill gap using ORM: HL7 datatypes allow for complicated operations which can easily translate into full table scans
  - fill gap using custom code: standard is hard to interpret

- Current HL7 specific ORM implementations:
  - seem to consider only limited amount of data
  - have incomplete null handling
The MGRID solution

Native HL7 datatype support in the database

- Creates a query language that is powerful, fast and easy to learn
- Enables a lightweight hibernate interface for JavaSIG interoperability
- Makes terminfo / SNOMED CT implementations easy
create table obs (ptnt int, date ts, dosage pq);
insert into obs values (1, '200910011214', '10 ml');
... 
insert into obs values (1, '200910091214', '20 ml');

select ptnt, effectivetime::date,
    convert(sum(dosage),'ml') as "sum",
    convert(avg(dosage),'ml') as "average"
from obs
group by ptnt,effectivetime::date
having sum(dosage) > '100ml' order by ptnt,effectivetime;

<table>
<thead>
<tr>
<th>ptnt</th>
<th>effective</th>
<th>sum</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2009-10-01</td>
<td>120 ml</td>
<td>10 ml</td>
</tr>
<tr>
<td>1</td>
<td>2009-10-02</td>
<td>150 ml</td>
<td>50 ml</td>
</tr>
<tr>
<td>1</td>
<td>2009-10-04</td>
<td>600 ml</td>
<td>100 ml</td>
</tr>
<tr>
<td>1</td>
<td>2009-10-09</td>
<td>180 ml</td>
<td>20 ml</td>
</tr>
<tr>
<td>2</td>
<td>2009-10-02</td>
<td>300 ml</td>
<td>50 ml</td>
</tr>
<tr>
<td>2</td>
<td>2009-10-09</td>
<td>120 ml</td>
<td>20 ml</td>
</tr>
<tr>
<td>3</td>
<td>2009-10-01</td>
<td>150 ml</td>
<td>10 ml</td>
</tr>
<tr>
<td>3</td>
<td>2009-10-02</td>
<td>450 ml</td>
<td>50 ml</td>
</tr>
<tr>
<td>3</td>
<td>2009-10-04</td>
<td>900 ml</td>
<td>100 ml</td>
</tr>
<tr>
<td>4</td>
<td>2009-10-04</td>
<td>300 ml</td>
<td>100 ml</td>
</tr>
</tbody>
</table>
(10 rows)
PQ - Physical Quantity

- PQ supports all ucum units
- PQs with the same dimensions can be converted and compared
- Database can do PQ arithmetic and conversions
- Nullflavors handling considering QTY
create domain pq_time as pq
    constraint pq_time_compares_to_s
    check (value /= 's');

CREATE DOMAIN

create domain ts_date as ts
    constraint ts_date_prcs_le_8_no_tz
    check (isnull(toany(timezone(value)))) = 'true'::bn
and   "precision"(value) <= 8
and   calendar(value)= 'GREG');

CREATE DOMAIN

create domain ts_date_full as ts
    constraint ts_date_prcs_eq_8_no_tz
    check (isnull(toany(timezone(value)))) = 'true'::bn
and   "precision"(value) = 8
and   calendar(value)= 'GREG');

CREATE DOMAIN

select '10 ml':pq_time;
ERROR: value for domain pq_time violates check constraint "pq_time_compares_to_s"
Example: IVL<PQ>

```sql
select '<8m':ivl_pq;
    ivl_pq
-------------------------
]Nullflavor.NINF m;8 m[
(1 row)

select contains('<8m':ivl_pq, '20cm':pq);
contains
---
true
(1 row)

select '2m-8m':ivl_pq;
    ivl_pq
-----------
[2 m;8 m]
(1 row)

select '[10ml; 20m3]':ivl_pq;
    ivl_pq
------------------
[1e-05 m3;20 m3]
(1 row)
```
Example: IVL<TS>

```sql
select '2008'::ts, promotion('2008'::ts), demotion(promotion('2008'::ts));
<table>
<thead>
<tr>
<th>ts</th>
<th>promotion</th>
<th>demotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>[2008;2009[</td>
<td>2008</td>
</tr>
</tbody>
</table>
(1 row)

select '200801 .. 02'::ivl_ts;
<table>
<thead>
<tr>
<th>ivl_ts</th>
</tr>
</thead>
<tbody>
<tr>
<td>[200801;200803[</td>
</tr>
</tbody>
</table>
(1 row)

select '200801 .. 02'::ivl_ts <@ '2001 .. 2010'::ivl_ts;
<table>
<thead>
<tr>
<th>column?</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
</tr>
</tbody>
</table>
(1 row)

select '<2005'::ivl_ts <@ '2001 .. 2010'::ivl_ts;
<table>
<thead>
<tr>
<th>column?</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
</tr>
</tbody>
</table>
(1 row)

select intervalafter('2001 .. 2010'::ivl_ts, '20080102'::ts);
<table>
<thead>
<tr>
<th>intervalafter</th>
</tr>
</thead>
<tbody>
<tr>
<td>]20080102;2011[</td>
</tr>
</tbody>
</table>
(1 row)
```
All 7 IVL forms are supported

Comparisons between interval forms are hard to program and easy to get wrong

Additional functions:
- overlaps, contains, intervalbefore, union
II - Instance Identifier

```sql
select '2.16.840.1.113883.5.1011: any extension'::ii;

2.16.840.1.113883.3.5.1011: any extension
(1 row)

select uuid_generate_v1mc()::ii;

9fb6a05c-c237-11de-ab08-47231b73ad4b
(1 row)

select 'wrong ii'::ii;
ERROR: could not parse root 'wrong ii'
LINE 1: select 'wrong ii'::ii;

HINT: The root must be either an OID or a UUID

create table testii (id ii);
insert into testii values ('2.16.840.1.113883.5.1011: any extension');
insert into testii values ('c478b896-c235-11de-8436-ebddef4f1dbe:another extension');
insert into testii select uuid_generate_v1mc() from generate_series(1,10000);

explain analyze select * from testii where id='c478b896-c235-11de-8436-ebddef4f1dbe';
```

---

```
Index Scan using iii on testii (cost=0.00..8.27 rows=1 width=25)
  Index Cond: (id = 'c478b896-c235-11de-8436-ebddef4f1dbe'::ii)
Total runtime: 0.179 ms
(3 rows)
```
CV - Coded Value

- CV is a CD with no translations and only a single concept
- CS is a CV with an implicit code system which is clear from the context. This codesystem is not visible in the string representation.
Example: CV

```sql
select oioid, oiname, oitype, oireleaseid, oistatus from pg_oid where oiname='ActClass';

<table>
<thead>
<tr>
<th>oioid</th>
<th>oiname</th>
<th>oitype</th>
<th>oireleaseid</th>
<th>oistatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.16.840.1.113883.5.6</td>
<td>ActClass</td>
<td>5</td>
<td>145</td>
<td>complete</td>
</tr>
<tr>
<td>2.16.840.1.113883.1.11.11527</td>
<td>ActClass</td>
<td>11</td>
<td>-1</td>
<td>-</td>
</tr>
</tbody>
</table>
```

(2 rows)

```sql
select 'OBS:2.16.840.1.113883.5.6'::cv;

<table>
<thead>
<tr>
<th>cv</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBS:2.16.840.1.113883.5.6</td>
</tr>
</tbody>
</table>
```

(1 row)

```sql
select 'OBS:2.16.840.1.113883.5.7'::cv;
ERROR: invalid code OBS for codeSystem ActPriority (7)
LINE 1: select 'OBS:2.16.840.1.113883.5.7'::cv;
  ^
```
Example: CV

```
select * from pg_conceptdomain where coname = 'ActClass';

<table>
<thead>
<tr>
<th>coid</th>
<th>coname</th>
</tr>
</thead>
<tbody>
<tr>
<td>362</td>
<td>ActClass</td>
</tr>
</tbody>
</table>

(1 row)

select 'OBS'::cv('ActClass');

<table>
<thead>
<tr>
<th>cv</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBS:2.16.840.1.113883.5.6@2009-08-30:2.16.840.1.113883.1.11.11527@2009-08-30</td>
</tr>
</tbody>
</table>

(1 row)
```
create table testcv (class cv('ActClass'), mood cv('ActMood'));

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>cv('ActClass')</td>
<td></td>
</tr>
<tr>
<td>mood</td>
<td>cv('ActMood')</td>
<td></td>
</tr>
</tbody>
</table>

insert into testcv values ('wrong', 'values');
ERROR: invalid code wrong for codeSystem ActClass (6)
insert into testcv values ('OBS', 'EVN.CRT');
insert into testcv values ('PROC', 'APT');
select class from testcv;

<table>
<thead>
<tr>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

select code(class), displayname(class), code(mood), displayname(mood) from testcv;

<table>
<thead>
<tr>
<th>code</th>
<th>displayname</th>
<th>code</th>
<th>displayname</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBS</td>
<td>observation</td>
<td>EVN.CRT</td>
<td>event criterion</td>
</tr>
<tr>
<td>PROC</td>
<td>procedure</td>
<td>APT</td>
<td>appointment</td>
</tr>
</tbody>
</table>

(2 rows)
Example: SNOMED CT

```sql
select conceptid::text::cv('SNOMED-CT') into cvgi from
(select conceptid from snomed.concepts order by random() limit 1000) a;

describe cvgi;

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>conceptid</td>
<td>cv   ('SNOMED-CT')</td>
<td></td>
</tr>
</tbody>
</table>

select code(conceptid),displayname(conceptid),codesystemname(conceptid)
from cvgi where conceptid << '404684003 '::cv('SNOMED-CT') limit 10;

code | displayname | codesystemname
---+---------------------------------------+----------------
5919001 | Rupture of papillary muscle | SNOMED-CT
193666005 | Unspecified visual field defect | SNOMED-CT
22395006 | Oesophageal body web | SNOMED-CT
254943007 | Benign tumour of choroid plexus | SNOMED-CT
293041006 | Sulphaguanidine adverse reaction | SNOMED-CT
199617004 | Large-for-dates unspecified | SNOMED-CT
168624001 | Plain X-ray of radius/ulna normal | SNOMED-CT
24825006 | Central alveolar hypoventilation syndrome | SNOMED-CT
57361003 | Anomaly of chromosome pair 5 | SNOMED-CT
269468002 | Malignant neoplasm of short bones of leg | SNOMED-CT

(10 rows)

select '71620000|fracture of femur|: |
116676008|associated morphology|=:21947006|compression fracture|
,363698007|finding site|=:29627003|structure of neck of femur|
'::cv('SNOMED-CT');

cv--------------------------------------------------- --------------------------------------
71620000|fracture of femur|: |
116676008|associated morphology|=:21947006|compression fracture|
,363698007|finding site|=:29627003|structure of neck of femur|:2.16.840.1.113883.6.96

(1 row)
```
In an Act instance where Act.code attribute is a SNOMED CT expression, the expression must represent a type of [363787002|observable entity], [«129125009|procedure with explicit context] or [«272379006|event]

```sql
create table act (class cv('ActClass'), code cv);

alter table act add constraint act_code_snomedct CHECK (
  (codesystem(code)= '2.16.840.1.113883.6.96') ?
  (code << '363787002:2.16.840.1.113883.6.96'::cv or
   code << '129125009:2.16.840.1.113883.6.96'::cv or
   code << '272379006:2.16.840.1.113883.6.96'::cv));

insert into act values ('PROC', '224166006':cv('SNOMED-CT'));
ERROR: new row for relation "act" violates check constraint "act_code_snomedct"

insert into act values ('ACT', '3974003':cv('SNOMED-CT'));

select code(class),displayname(code) from act;
class | displayname
-------+---------------------------
ACT | Contact with sharp leaves
(1 row)
ORM MGRID using Hibernate

- Consider for instance storing a table of PQs;
- Differences using ‘standard datatypes’ versus HL7 datatypes:

<table>
<thead>
<tr>
<th>versus</th>
<th>standard</th>
<th>HL7 datatypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>java api</td>
<td>JavaSIG</td>
<td>JavaSIG + MGRID mappings</td>
</tr>
<tr>
<td>table definition</td>
<td>2 columns; value unit</td>
<td>1 column; pq</td>
</tr>
<tr>
<td>query</td>
<td>full scan, make canonical, filter</td>
<td>partial scan</td>
</tr>
</tbody>
</table>
Hibernate mapping is trivial

- provide own `nullSafeSet` and `nullSafeGet`

```java
public void nullSafeSet(PreparedStatement st, Object rawValue, int index)
    throws HibernateException {
    try {
        if (rawValue == null) {
            st.setNull(index, java.sql.Types.OTHER);
        } else {
            PGobject object = new PGobject();
            PQ pq = (PQ) rawValue;
            object.setType("pq");
            object.setValue("" + pq.value() + " " + pq.unit());
            st.setObject(index, object);
        }
    } catch (SQLException ex) {
        throw new HibernateException(ex);
    }
}
```
Hibernate performance comparison

- Slightly better performance when using full table scans

- Enormous performance boost when using indexes
Questions
Contact

ir. Willem Dijkstra
Managing Director

www.mgrid.net
T +31 886 474 302
F +31 886 474 301
M +31 611 144 118
w.p.dijkstra@mgrid.net

Oostenburgervoorstraat 106-114
1018 MR Amsterdam

PO BOX 1287
1000 BG Amsterdam
The Netherlands