Using Python and Paver to Control a Large Medical Informatics ETL Process

Dan Connolly¹, Alex F. Bokov², Russ Waitman¹, Alfredo Tirado-Ramos²

¹University of Kansas Medical Center
²University of Texas Health Science Center at San Antonio

PyData Silicon Valley
May 2014

This project is supported in part by NIH grant UL1TR000001 and PCORI Contract CDRN-1306-04631
Dan Connolly, KUMC Medical Informatics

writing software to support research

former lives: W3C, Convex, U.T. Austin, ...

GPC Software Dev Lead
informatics.kumc.edu
dconnolly@kumc.edu

informatics.gpcnetwork.org

1st open source release: XcRichText-1.5.tar.Z 1992
Remember gatekeeper.dec.com?

www.madmode.com@dckc
Alex Bokov, UTHSCSA

- Steady childhood diet of space opera science fiction
- BS in Cellular and Molecular Biology from University of Michigan, Ann Arbor
- PhD in Physiology from UTHSCSA
- Technical Lead, Clinical Informatics Research Division at UTHSCA Department of Epidemiology and Biostatistics.
Greater Plains Collaborative (GPC)

Ten hospitals and clinics working to enable research by facilitating access to electronic medical records (EMRs).

informatics.gpcnetwork.org
Why Mine Medical Data?

- Retrospective studies: discoveries hiding in plain sight?
- Feasibility studies: enough eligible patients for a clinical trial?
- Cohort discovery: whom to recruit for a clinical trial?
- Exploratory analysis: visualization, clustering, etc.
Our EMR Data Mining Challenges

Our EMR Data Mining Challenges

Performance

Size

Our EMR Data Mining Challenges

- Performance
- Repeatability
- Size
- Complexity

Our EMR Data Mining Challenges

Performance

Repeatability

Size

Researcher Freedom

Complexity

Patient Privacy

Size Challenge: 1.3B obs. on 2M patients

<table>
<thead>
<tr>
<th>Category</th>
<th>Observation</th>
<th>Patients</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allergy</td>
<td>468K</td>
<td>219K</td>
<td>KUH/O2/Epic</td>
</tr>
<tr>
<td>Cancer Cases</td>
<td>10.5M</td>
<td>71.6K</td>
<td>KUH Cancer Registry</td>
</tr>
<tr>
<td>Demographics</td>
<td>19.2M</td>
<td>2.02M</td>
<td>KUH Billing (✉️ O2 via SMS)</td>
</tr>
<tr>
<td></td>
<td>21.4K</td>
<td>21.4K</td>
<td>Frontiers participant registry</td>
</tr>
<tr>
<td></td>
<td>191.3K</td>
<td>191.3K</td>
<td>Social Security Death Index</td>
</tr>
<tr>
<td>Diagnoses (IDC9)</td>
<td>54M</td>
<td>727K</td>
<td>KUH/O2/Epic</td>
</tr>
<tr>
<td>History</td>
<td>22.5M</td>
<td>359K</td>
<td>KUH/O2/Epic</td>
</tr>
<tr>
<td>Lab Results</td>
<td>132M</td>
<td>336K</td>
<td>KUH/O2/Epic</td>
</tr>
</tbody>
</table>
Complexity Challenge: Rich Data From Heterogeneous Sources

Incorporate Clinical, Administrative, Research Datasources

- Inpatient and outpatient electronic medical records (Epic)
- Professional Services Billing and Scheduling (GE IDX)
- KUCC Biospecimen Shared Resource Samples Database
- Hospital (KUH) Tumor Registry (NAACCR format)
- Social Security Death Master File (NIST format)
- Technical Charges from hospital and clinics (UHC validated format)
- Research Data Capture (REDCap)
- Clinical Research Information System (Velos)

HERON’s current contents

- Demographics (master patient index)
- Race/Ethnicity
- Laboratory Results
- Nursing observations/vital signs
- Clinical Diagnoses (ICD9)
- Medications (dispensed, ordered, home meds, administered)
- Physician Orders
- Procedure charges (CPT)
- Outpatient Billing diagnoses (ICD9)
- Inpatient visit/provider service
- Family/Social/Surgical Past Medical History
- Microbiology
- Allergies/Reactions
- Specimen collected
- Tumor Staging and Grade
- Diagnosis and Treatment
- Survival and Progression
- Death per Social Security Administration
- MDRG, APDRG, LOS, Readmissions
- Technical Charge Diagnoses ICD9
- Service line, AHRQ, quality and JCAHO core measures
- Triple Negative Breast Cancer Registry initial pilot completed

Status as of July 18, 2013
Patient Privacy Challenge

- HIPAA (Health Insurance Portability and Accountability Act)
- Security
- IRB (Institutional Review Board)
- De-identification
  - Patient IDs
  - Dates
How GPC Overcomes These Challenges: The HERON Architecture

Healthcare Enterprise Repository for Ontological Narration

I2B2: Query Interface

Drag concepts from upper left into panels on the right
I2B2: Query Interface

Dragging over the second condition
I2B2: Query Interface

Run the Query
Query took 4 seconds
497 patient in cohort
I2B2: Query Result
I2B2 Data Model
How GPC Overcomes These Challenges: The HERON Architecture

Healthcare Enterprise Repository for Ontological Narration
Work on B While A is in Production

last month’s ETL

A

I2B2 prod

Scientist

this month’s ETL

B

I2B2 dev

Tester
Then Flip/Flop and Repeat

wipe and run new ETL

becomes the production data warehouse

Scientist

Tester
Trade-offs of Flip/Flop Strategy

Pro:
- short (~1 hr) monthly down-time
  - vs. ~50hr ETL time

Con:
- 2x storage space
  - disk is cheaper than customer, dev time
  - even using FusionIO solid state storage
ETL Job: Road(s) Not Taken

- Tools: Talend, Pentaho
  - evaluated
  - each row through the client?!
  - abstraction facilities lacking
    - I’m not about to manually maintain N copies of list X, Y, and Z of columns, tables, steps, etc.

- SSIS
  - Main data source is Oracle

- Oracle PL/SQL
Features, Issues in “Lights Out” ETL Tools

- Powerful Job Control UI
- Logging, Monitoring, Testing
- Visualization
- Training, Community
  - Team scaling
  - Separation of Concerns
SQL and Python Approaches

1. Starter SQL code from i2b2 community
   a. Manually run SQL in Oracle sqldeveloper
   b. no python
2. Python dev puts SQL in strings in python code
   a. Training problem for SQL devs
3. SQL goes in .sql files; python code reads .sql files
   a. ~20KLoc SQL
   b. ~5KLoc Python

oops!
def iter_statement(txt):
    '''Iterate over SQL statements in a script.

    >>> list(iter_statement("drop table foo; create table foo"))
    [(1, '', 'drop table foo'), (1, '', 'create table foo')]

    ...
paver: Modular CLI

- ETL Job Control UI?
  - quick-n-dirty: command line
- How to run just one (new) part?
- How to arrange parts?

paver @task ≈ Makefile target

% paver config=test.ini heron_load

@task
def heron_load():
    ...

paver by Kevin Dangoor
@task Dependencies

% paver load_epic_labs runs make_epic_lab_views first.
from paver.easy import task, needs
...

task
@needs('load_labs_regex', 'load_component_whitelist')
@view_loader('epic_labs_transform.sql', **epic_view_kwds)
def make_epic_lab_views(options):
    ...

task
@needs('make_epic_lab_views')
@multi_loader([('observation_fact_lab', 'Epic Lab observation facts')], **epic_facts_loader_kwds)
def load_epic_labs(options):
    ...
Visualizing Task Dependencies

- task-deps.dot.pdf diagram
- task-deps.csv tabular data used to build the diagram
- datadeps.py - static analysis tool to extract the dependencies
Visualizing Task Dependencies

- [task-deps.dot.pdf](#) diagram
- [task-deps.csv](#) tabular data used to build the diagram
- [datadeps.py](#) - static analysis tool to extract the dependencies
Visualizing Task Dependencies

- task-deps.dot.pdf diagram
- task-deps.csv tabular data used to build the diagram
- datadeps.py - static analysis tool to extract the dependencies
class ETLModuleScanner(ast.NodeVisitor):
    def __init__(self, sources):
        ast.NodeVisitor.__init__(self)
        self._deps = []  # mutable state. ew.
        self._sources = sources

    def visit_FunctionDef(self, node):
        dest = node.name
        if self.is_task(node):
            for src, relationship in self.needs(node):
                self._deps.append((src, dest, relationship))
class ETLModuleScanner(ast.NodeVisitor):
...

@classmethod
def needs(cls, task):
    decorators = task.decorator_list
    if not decorators:
        return []
    return [(arg.s, d.func.id)  # assume @needs() args are string literals
            for d in [d for d in decorators
                      if isinstance(d, ast.Call)
                      and isinstance(d.func, ast.Name)
                      and d.func.id in ('needs', 'might_call')]
            for arg in d.args]
HERON Development Process

- Understand how Scientist will use Features
  - Enhancement #2160: Query Epic/O2 Allergy and Reaction in HERON
  - User story: How many patients get hives from penicillin?
- Analysis
  - Find data in source system
  - Develop SQL queries to answer query sketches
  - Transform to i2b2 fact table, metadata (terms) table
- Development:
  - SQL to run “lights out”
  - Automated testing
/* ... make sure age does not exceed 90 */
select case when count(*) > 0 then 1/0
else 1 end deid_age_over_90_pd from(
    select pat.age_in_years_num
    from blueherondata.patient_dimension@deid pat
    where pat.age_in_years_num > 90
);
Example: ETL for Labs

CLARITY (source)

identified i2b2 star schema

de-identified i2b2 star schema
ETL Part I:
Transform Source to i2b2 schema
Our Target: I2B2 Schema

- **patient_dimension**
  - PK: Patient_Num
  - Birth_Date
  - Death_Date
  - Vital_Status_CD
  - Age_Num
  - Gender_CD
  - Race_CD
  - Ethnicity_CD

- **observation_fact**
  - PK: Patient_Num, Encounter_Num
  - Concept_CD
  - Observer_CD
  - Start_Date
  - Modifier_CD
  - Instance_Num
  - End_Date
  - ValType_CD
  - TVal_Char
  - NVal_Num
  - ValueFlag_CD
  - Observation_Blob

- **visit_dimension**
  - PK: Encounter_Num
  - Start_Date
  - End_Date
  - Active_Status_CD
  - Location_CD

- **concept_dimension**
  - PK: Concept_Path
  - Concept_CD
  - Name_Char

- **observer_dimension**
  - PK: Observer_Path
  - Observer_CD
  - Name_Char
ETL Part I: Transform Source to I2B2

--Create a view of smoking tobacco history
create or replace view smoking_tob_use
as
select to_char(shx.pat_enc_csn_id) as encounter_ide   ,
    shx.pat_id as patient_ide                            ,
    'KUMC|SMOKING_TOB_USE:||stu.smoking_tob_use_c as concept_cd,
    shx.contact_date as start_date                       ,
    '@' as modifier_cd                                   ,
    1 as instance_num                                    ,
    shx.contact_date as end_date                         ,
    shx.contact_date as update_date                      ,
    mod(shx.pat_enc_csn_id, &&heron_etl_chunks)+1 as part,
from clarity.social_hx shx
join clarity.zc_smoking_tob_use stu on shx.smoking_tob_use_c = stu.
    discrete_fact_constants fconst
where shx.contact_date is not null
;
ETL Part I:
Transform Source to I2B2

--Create a view of smoking tobacco history
create or replace view smoking_tob_use
as
select to_char(shx.pat_enc_csn_id) as encounter_ide,
    shx.pat_id as patient_ide,
    'KUMC|SMOKING_TOB_USE:'||stu.smoking_tob_use_c as concept_cd,
    shx.contact_date as start_date,
    '@' as modifier_cd,
    1 as instance_num,
    shx.contact_date as end_date,
    shx.contact_date as update_date,
    mod(shx.pat_enc_csn_id, &&heron_etl_chunks)+1 as part,
    fconst.* -- add in default values for provider_id, valtype_cd etc.
from clarity.social_hx shx
join clarity.zc_smoking_tob_use stu on shx.smoking_tob_use_c = stu.smoking_tob_use_c,
    discrete_fact_constants fconst
where shx.contact_date is not null;
**ETL Part I:**

**Transform Source to I2B2**

```
--Create a view of smoking tobacco history
create or replace view smoking_tob_use
as
    select to_char(shx.pat_enc_csn_id) as encounter_ide ,
           shx.pat_id as patient_ide ,
           'KUMC|SMOKING_TOB_USE:'||stu.smoking_tob_use_c as concept_cd,
           shx.contact_date as start_date ,
           '@' as modifier_cd ,
           1 as instance_num ,
           shx.contact_date as end_date ,
           shx.contact_date as update_date ,
           mod(shx.pat_enc_csn_id, &&heron_etl_chunks)+1 as part,
    fconst.* -- add in default values for provider_id, valtype_cd etc.
    from clarity.social_hx shx
    join clarity.zc_smoking_tob_use stu on shx.smoking_tob_use_c = stu.smoking_tob_use_c,
    discrete_fact_constants fconst
    where shx.contact_date is not null ;
```
ETL Part I: Transform Source to I2B2

--Create a view of smoking tobacco history
create or replace view smoking_tob_use
as

select to_char(shx.pat_enc_csn_id) as encounter_ide,
     shx.pat_id as patient_ide,
     'KUMC|SMOKING_TOB_USE:'||stu.smoking_tob_use_c as concept_cd,
     shx.contact_date as start_date,
     '@' as modifier_cd,
     1 as instance_num,
     shx.contact_date as end_date,
     shx.contact_date as update_date,
     mod(shx.pat_enc_csn_id, &&heron_etl_chunks)+1 as part,
     fconst.* -- add in default values for provider_id, valtype_cd etc.
from clarity.social_hx shx
join clarity.zc_smoking_tob_use stu on shx.smoking_tob_use_c = stu.smoking_tob_use_c,
    discrete_fact_constants fconst
where shx.contact_date is not null;
--Create a view of smoking tobacco history
create or replace view smoking_tob_use
as
select to_char(shx.pat_enc_csn_id) as encounter_ide ,
       shx.pat_id as patient_ide ,
       'KUMC|SMOKING_TOB_USE:'||stu.smoking_tob_use_c as concept_cd,
       shx.contact_date as start_date ,
       '@' as modifier_cd ,
       1 as instance_num ,
       shx.contact_date as end_date ,
       shx.contact_date as update_date ,
       mod(shx.pat_enc_csn_id, &&heron_etl_chunks)+1 as part,
       fconst.* -- add in default values for provider_id, valtype_cd etc.
from clarity.social_hx shx
join clarity.zc_smoking_tob_use stu on shx.smoking_tob_use_c = stu.
smoking_tob_use_c,
   discrete_fact_constants fconst
where shx.contact_date is not null ;

* or a table, for performance
ETL Part II: Correlate Patients, Load, De-identify
ETL Part II:  (Join with I2B2 mapping tables and) Load

```sql
insert into observation_fact_upload(
    patient_num, encounter_num,
    concept_cd,
    ...)
select pmap.patient_num, emap.encounter_num,
    f.concept_cd,
    f...
from &&epic_fact_view f,
    NightHeronData.patient_mapping pmap,
    NightHeronData.encounter_mapping emap,
    NightHeronData.upload_status up
where pmap.patient_ide = f.patient_ide
    and emap.encounter_ide = f.encounter_ide
...```
When it Works in I2B2, Capture Query for Re-Play

capture_query.py

OK
Code Review Checklist

• Is the use-case clear? captured in automated tests?
  • Do the integration tests work?
  • Is logging sufficient to diagnose problems?

• Is the code secure? Is the flow of authority clear?

• Would you be happy to maintain the code? … to show it to peers?
  • Is the code DRY?
  • Do the doctests explain the design? As a story?

• Is the copyright notice clear and correct?
  • If we're using anyone else's work, is it clearly licensed and acknowledged?
35+ Consecutive Monthly Releases

2014-01-23: HERON Lovewell improves navigation and fixes Frontiers …

2013-12-30: HERON Toronto update powered by newer, faster hardware …

2013-11-15: HERON Milford simplifies the Cancer Cases folder

2013-10-15: HERON Solomon marks the 32nd consecutive release

2013-09-26: HERON Chikaskia release introduces HERON DUA to …

2013-08-20: HERON Crooked Creek release introduces the HERON …

2013-07-16: HERON Big Blue release introduces allergy and reaction data

2013-06-25: HERON Sappa release improves performance and …
QA, CI with Jenkins

Project heron-test

Runs pep8, pyflakes, and unittests (+code coverage) for the heron_load directory of Heron ETL.

todo: integrate - dry-run. Perhaps using Jenkins upstream/downstream stuff?

Coverage Report
Workspace
Recent Changes
LATEST TEST RESULT (no failures)

Permalinks
- Last build (#224), 8 hr 11 min ago
- Last stable build (#224), 8 hr 11 min ago
- Last successful build (#224), 8 hr 11 min ago
- Last failed build (#217), 14 days ago
- Last unstable build (#215), 16 days ago
- Last unsuccessful build (#217), 14 days ago

Code Coverage

Test Result Trend

Packages 100% Files 100% Classes 100% Lines 60% Conditionals 100%

Classes — Conditionals — Files — Lines — Packages

Count

Finish reading the document.
Automated QA, CI

• Each commit/push: unit tests

• Hourly:
  • Monitoring query

• Daily:
  • ETL integration test build with small test data set
  • Performance suite: check execution plans

• Weekly:
  • ETL integration test build with a sample of production data*
  • Performance suite: measure real query times
Toward Structured Logging

• Got tired of grepping log files
• Explored alternatives to paver logging API:
  • python configurable logging API
  • Tried XUnitFormatter for Jenkins UI
• Developed NestedEvents, CSVFormatter
  • Excel (filter: connections, errors, …), R plots
  • Filtered text summary
  • HTML bigger-takes-longer outline
    • e.g. heron_beaver_etl_log_outline.html
Visualizing ETL Runtime from Log
Visualizing Data Dependencies

for a task: load_epic_labs
Static Analysis for SQL

• start with select_parser.py
  • example from PyParsing
• Expand to insert, …
• Tweak for Oracle syntax
• Extract dependencies

hierarchical_query_clause = Group(
  Optional(START + WITH + expr)
  + CONNECT + BY + Optional(PRIOR) + expr)
# Dependency Data

<table>
<thead>
<tr>
<th>code_filename</th>
<th>code_line</th>
<th>ide_column</th>
<th>src_table</th>
<th>src_schema</th>
<th>dest_table</th>
<th>dest_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>epic_labs_transform.sql</td>
<td>39</td>
<td>6</td>
<td>ORDERRESULTS</td>
<td>CLARITY</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>TABLE</td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>39</td>
<td>8</td>
<td>ORDERPROC</td>
<td>CLARITY</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>TABLE</td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>39</td>
<td>8</td>
<td>PAT_ENC</td>
<td>CLARITY</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>TABLE</td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>39</td>
<td>65</td>
<td>COMPONENT_WHITELIST</td>
<td>CLARITY</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>TABLE</td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>86</td>
<td>33</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>LAB_REGEX_SUMMARY_TBL</td>
<td>TABLE</td>
<td></td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>86</td>
<td>39</td>
<td>DUAL</td>
<td>LAB_REGEX_SUMMARY_TBL</td>
<td>TABLE</td>
<td></td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>86</td>
<td>7</td>
<td>LAB_VALUE_REGEX</td>
<td>LAB_REGEX_SUMMARY_TBL</td>
<td>TABLE</td>
<td></td>
</tr>
<tr>
<td>epic_labs_transform.sql</td>
<td>136</td>
<td>9</td>
<td>CLARITY_LAB_SUMMARY_TBL</td>
<td>OBSERVATION_FACT_LAB</td>
<td>VIEW</td>
<td></td>
</tr>
</tbody>
</table>
HERON ETL: A Workable Approach

- Embrace SQL; exploit general-purpose programming language
  - Version control, unit testing
  - No Diagram UI, but available as by-product
- Positive training, team scaling experience:
  - KUMC: from 2 to ~5 devs
  - GPC: 3 sites picking up the code
- Robust, secure:
  - 35 monthly releases
  - ~1hr down time per month
  - Provides research benefit without excessive patient, business risk
Future Directions: Interactive R statistical visualization
Refactoring With “Well-Typed” Python

- Python is dynamic and lightweight...
- Can teammates read the code with the types implicit?
- Do I have enough tests to refactor?

- “well-typed” python dialect emerged from py2scala, a hobby project
- With some limitations (e.g. no **kwargs)...
- Reliable parameter type documentation
- Never miss another rename!
- Nor latent type mismatch!
def __call__(self, env, start_response):
    '''Handle HTTP request per `WSGI`__.

    __ http://www.python.org/dev/peps/pep-0333/
    ...
    if env['REQUEST_METHOD'] != 'POST':
        ...

def __call__(self, env, start_response):
    '''Handle HTTP request per `WSGI`_.
    __ http://www.python.org/dev/peps/pep-0333/
    :param env: dictionary of HTTP request headers etc.
    :param start_response: function to start HTTP response
    :rtype: sequence of strings
    '''
    if env['REQUEST_METHOD'] != 'POST':
        ...

def __call__(self, env, start_response):
    '''Handle HTTP request per `WSGI`__.
    __ http://www.python.org/dev/peps/pep-0333/
    :param env: access to HTTP request
    :type env: Dict[String, String]
    :param start_response: access to start HTTP response
    :type start_response: (String, Seq[(String, String)]) => Unit
    :rtype: Iterable[String]
    ...

    if env['REQUEST_METHOD'] != 'POST':
        ...

 Explicit, Statically Checked Types
Uses Existing Conventions

```python
__call__(env, start_response)
Handle HTTP request per WSGI.
```

**Parameters:**
- `env (Dict[String, String])` – access to HTTP request
- `start_response ((String, Seq[(String, String)]) => Unit)` – access to start HTTP response

**Return type:** `Iterable[String]`
Thank You

- KUMC Medical Informatics Team: Russ Waitman, Bhargav Adagarla, Sravani Chandaka, Steve Fennel, Nathan Graham, Brandon Hamlin, Matthew Hoag, Vincent Leonardo, Tamara McMahon, Suman Suman, Brittany Zschoche
- UTHSCSA DEB Clinical Informatics Research Division: Angela Bos, Margie Guevara, Viresh Thusu, Alfredo Tirado-Ramos
- UTHSCSA DEB IS Division: Santhi Dandamudi, John Hough, Shobhana Krishnakumar
- The rest of our colleagues in the GPC
- The nurses and physicians who are busting their asses in a way that makes our all-nighters seem like a joke.
- The patients. Hang in there guys! We’re working as fast as we can.
Links and References

informatics.gpcnetwork.org
informatics.kumc.edu

i2b2.org

cx_Oracle
paver by Kevin Dangoor of SitePen.
PyParsing by Paul McGuire
Glad you asked...
i2b2 observation_fact table: EAV

<table>
<thead>
<tr>
<th>Patient</th>
<th>Sex</th>
<th>Pulse</th>
<th>Diagnosis</th>
<th>... x400K+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>M</td>
<td>75</td>
<td>Bronchitis</td>
<td></td>
</tr>
<tr>
<td>Sue</td>
<td>F</td>
<td>68</td>
<td>Diabetes</td>
<td></td>
</tr>
</tbody>
</table>

- $E \approx \text{Patient\_Num}$
- $A \approx \text{Concept\_CD}$
- $V \approx \text{TVal} / \text{NVal}$
- wrinkles
  - start/end date
  - modifier
  - observer
  - ...

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Sex</td>
<td>M</td>
</tr>
<tr>
<td>Bob</td>
<td>Pulse</td>
<td>75</td>
</tr>
<tr>
<td>Bob</td>
<td>Diagnosis</td>
<td>Bronchitis</td>
</tr>
<tr>
<td>Sue</td>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Sue</td>
<td>Pulse</td>
<td>68</td>
</tr>
<tr>
<td>Sue</td>
<td>Diagnosis</td>
<td>Diabetes</td>
</tr>
<tr>
<td>Person #1774</td>
<td>f</td>
<td>Non-hispanic</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person #1809</th>
<th>f</th>
<th>Non-hispanic</th>
<th>005-301070 BMI (Calculated) [268,216 facts; 80,385 patients]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Diabetes mellitus [376,016 facts; 31,783 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frontiers Research Participant Registry [7,654 facts; 7,654 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLUCOSE (#2011) [1,087,668 facts; 119,524 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HEMOGLLOBIN A1C (#2034) [85,793 facts; 35,269 patients]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person #21253</th>
<th>f</th>
<th>Non-hispanic</th>
<th>005-301070 BMI (Calculated) [268,216 facts; 80,385 patients]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Diabetes mellitus [376,016 facts; 31,783 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frontiers Research Participant Registry [7,654 facts; 7,654 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLUCOSE (#2011) [1,087,668 facts; 119,524 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HEMOGLLOBIN A1C (#2034) [85,793 facts; 35,269 patients]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person #13473</th>
<th>f</th>
<th>Non-hispanic</th>
<th>005-301070 BMI (Calculated) [268,216 facts; 80,385 patients]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Diabetes mellitus [376,016 facts; 31,783 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frontiers Research Participant Registry [7,654 facts; 7,654 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLUCOSE (#2011) [1,087,668 facts; 119,524 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HEMOGLLOBIN A1C (#2034) [85,793 facts; 35,269 patients]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person #3138</th>
<th>m</th>
<th>Non-hispanic</th>
<th>005-301070 BMI (Calculated) [268,216 facts; 80,385 patients]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Diabetes mellitus [376,016 facts; 31,783 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frontiers Research Participant Registry [7,654 facts; 7,654 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLUCOSE (#2011) [1,087,668 facts; 119,524 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HEMOGLLOBIN A1C (#2034) [85,793 facts; 35,269 patients]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person #25806</th>
<th>f</th>
<th>Non-hispanic</th>
<th>005-301070 BMI (Calculated) [268,216 facts; 80,385 patients]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Diabetes mellitus [376,016 facts; 31,783 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frontiers Research Participant Registry [7,654 facts; 7,654 patients]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GLUCOSE (#2011) [1,087,668 facts; 119,524 patients]</td>
</tr>
</tbody>
</table>
## Python Decorators wrap functions

<table>
<thead>
<tr>
<th>Decorator Syntax</th>
<th>Elaboration/Results</th>
</tr>
</thead>
</table>
| `@dec2`  
`@dec1`  
def func(arg1, arg2, ...):  
    pass | def func(arg1, arg2, ...):  
    pass  
    func = dec2(dec1(func)) |
| def _memoize(f):  
    cache = {}  
    def f_memoized(*args):  
        try:  
            return cache[[args]  
        except KeyError  
            out = cache[args] = f(*args)  
            return out  
        return f_memoized  

def memoize():  
    return _memoize(f) | >>> crunch_n_munch(12345)  
# ... lots of time passes  
9273983  

>>> crunch_n_munch(12345)  
# no time!  
9273983 |
DVCS with ~1 branch per ticket

Emerging: MultiSiteDev
GPC: initially characterizing three cohorts

- Use honest brokers for preliminary counts and data extraction
- Focus on Data Sharing Agreement to support observational studies with de-identified data

- GPC IRB Initial Consensus: IRB protocol not required for fully de-identified data
- Limited datasets to monitor prospective Comparative Effectiveness trials that have IRB approval
Efficiency, Scalability: R Engine Cell Data Path

CRC Cell sends back to the plug-in an XML response containing the requested data (extracted from the i2b2 datawarehouse).
Efficiency, Scalability: 
rgate connects R to Oracle directly

Like the CRC cell, 
rgate calls the PM cell to validate authorization.
Object Capability Discipline supports the Principle of Least Authority

Memory safety and encapsulation¹
+ Effects only by using held references²
+ No powerful references by default³

Reference graph ≡ Access graph
Only connectivity begets connectivity
Natural Least Authority
OO expressiveness for security patterns

acct = db_config()

in a: b.m(c)

M. Miller, C. Morningstar, B. Frantz; "Capability-based Financial Instruments"; Proceedings of Financial Cryptography (Springer-Verlag); 2000
Reproducible Research: Modifiers, Concept Paths and sharing ETL code

• Clear attribution likely as important as data standards
  • Billing, EMR, problem list, administration versus dispense?
• Load with the local terminology first; don’t miss unmapped observations.
  • Map to standards secondarily in the concept space.
• Works around mapping challenges with contributing organizations; support multiple organizing hierarchies
# Self-Service, Ad-hoc Queries

## Query Volume

<table>
<thead>
<tr>
<th>Year-Month</th>
<th>Queries</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-04</td>
<td>488</td>
<td>46</td>
</tr>
<tr>
<td>2014-03</td>
<td>917</td>
<td>42</td>
</tr>
<tr>
<td>2014-02</td>
<td>858</td>
<td>35</td>
</tr>
<tr>
<td>2014-01</td>
<td>567</td>
<td>35</td>
</tr>
<tr>
<td>2013-12</td>
<td>405</td>
<td>33</td>
</tr>
<tr>
<td>2013-11</td>
<td>621</td>
<td>43</td>
</tr>
<tr>
<td>2013-10</td>
<td>1164</td>
<td>42</td>
</tr>
<tr>
<td>2013-09</td>
<td>1008</td>
<td>35</td>
</tr>
<tr>
<td>2013-08</td>
<td>1157</td>
<td>52</td>
</tr>
<tr>
<td>2013-07</td>
<td>641</td>
<td>36</td>
</tr>
<tr>
<td>2013-06</td>
<td>299</td>
<td>21</td>
</tr>
<tr>
<td>2013-05</td>
<td>372</td>
<td>29</td>
</tr>
</tbody>
</table>
Performance Challenge: Interactive Time Frame

- 2 sec: great!
- 20 sec: good
- 2 min: acceptable

Performance of Recent Queries

Current release: HERON Kirwan (data through March 2014)

<table>
<thead>
<tr>
<th>ID</th>
<th>Created</th>
<th>Status</th>
<th>Name</th>
<th>User</th>
<th>Groups</th>
<th>Terms</th>
<th>Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>5018</td>
<td>April 28 09:</td>
<td>COMPLETED</td>
<td>HERON MONITORING</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0:00:01</td>
</tr>
<tr>
<td></td>
<td>03:31pm</td>
<td></td>
<td>QUERY</td>
<td></td>
<td></td>
<td></td>
<td>**</td>
</tr>
<tr>
<td>5018</td>
<td>April 28 08:</td>
<td>COMPLETED</td>
<td>[AM11-Delir-Acute@20:43:34]</td>
<td>...</td>
<td>3</td>
<td>12</td>
<td>0:00:18</td>
</tr>
<tr>
<td></td>
<td>43:38pm</td>
<td></td>
<td>Query</td>
<td></td>
<td></td>
<td></td>
<td>****</td>
</tr>
<tr>
<td>8131</td>
<td>April 28 06:</td>
<td>COMPLETED</td>
<td>Patient Set for &quot;Breas-004- -withi@18:02:20&quot;</td>
<td>...</td>
<td>0</td>
<td>4</td>
<td>0:00:00</td>
</tr>
<tr>
<td></td>
<td>02:44pm</td>
<td></td>
<td>Timeline</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>5018</td>
<td>April 28 06:</td>
<td>COMPLETED</td>
<td>Breas-004- -withi@18:02:20</td>
<td>...</td>
<td>3</td>
<td>4</td>
<td>0:00:15</td>
</tr>
<tr>
<td></td>
<td>02:26pm</td>
<td></td>
<td>Patient list</td>
<td></td>
<td></td>
<td></td>
<td>****</td>
</tr>
<tr>
<td>5018</td>
<td>April 28 06:</td>
<td>COMPLETED</td>
<td>Breas-004- -withi@18:02:20</td>
<td>...</td>
<td>3</td>
<td>4</td>
<td>0:00:15</td>
</tr>
<tr>
<td></td>
<td>02:26pm</td>
<td></td>
<td>Patient list</td>
<td></td>
<td></td>
<td></td>
<td>****</td>
</tr>
</tbody>
</table>
Novel Terminologies: Flowsheets, ...
Richness: 400K+ Terms
Security: Protected Health Information (PHI)

Privacy, Security
De-identification

giving access to de-identification means the researcher doesn’t have to jump thru IRB hoops to start using it

* And login disclaimer
HERON Product: Query and Analysis with I2B2

Recovering from ETL failures

- drop bad chunk(s) of data
  a. using e.g. `paver drop_uploads upload_ids=7,8,9`
- fix problem (code, Oracle temp space, ...)
- `paver failed_task remaining_tasks1`

...  

A bit fragile, but can save dozens of hours.
Standard Terminology: ICD9 Diagnoses
Correlating patients across sources: i2b2 patient_mapping table

<table>
<thead>
<tr>
<th>patient_mapping</th>
<th>patient_ide</th>
<th>patient_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient_ide_source</td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="mailto:Epic@kumed.com">Epic@kumed.com</a></td>
<td>68</td>
<td>100</td>
</tr>
<tr>
<td><a href="mailto:Epic@kumed.com">Epic@kumed.com</a></td>
<td>81</td>
<td>150</td>
</tr>
<tr>
<td><a href="mailto:SMS@kumed.com">SMS@kumed.com</a></td>
<td>676</td>
<td>150</td>
</tr>
<tr>
<td><a href="mailto:SMS@kumed.com">SMS@kumed.com</a></td>
<td>921</td>
<td>100</td>
</tr>
<tr>
<td><a href="mailto:SSN@ssa.gov">SSN@ssa.gov</a></td>
<td>123-45-6789</td>
<td>100</td>
</tr>
<tr>
<td><a href="mailto:SSN@ssa.gov">SSN@ssa.gov</a></td>
<td>987-65-4321</td>
<td>150</td>
</tr>
</tbody>
</table>

Bob

Pete
Sidestep: Lift script calls to decorators

```python
from paver.easy import task, needs, might_call, no_help
...
from heron_build import single_loader, multi_loader, view_loader

@task
@no_help
@needs('load_labs_regex', 'load_component_whitelist')
@view_loader('epic_labs_transform.sql', **epic_view_kwds)
def make_epic_lab_views(options):
    pass

@task
@needs('make_epic_lab_views')
@multi_loader([(observation_fact_lab, 'Epic Lab observation facts')],
              **epic_facts_loader_kwds)
def load_epic_labs(options):
    '''Load, de-id Lab observations from Epic.
    ...'''
```

Decorate all the things!
Sidestep: Lift script calls to decorators

Oops:
- Scripts are parameterized
  - &&epic_facts_view
- Data Dependencies are sensitive to parameter values.

Consolation Prize:
- DRY
  - Factored out redundancy in @task implementations

Boobie prize:
- obscure
  - vs. function calls