Data without Provenance is like a Day without Sunshine

SYSTEMS RESEARCH AT HARVARD

Margo Seltzer

Uri Braun, Marc Chiarini, David Holland, (Kiran-Kumar Muniswamy-Reddy),
Daniel Margo, Peter Macko, Nicholas Murphy

October 25, 2011

Data & Metadata & Provenance

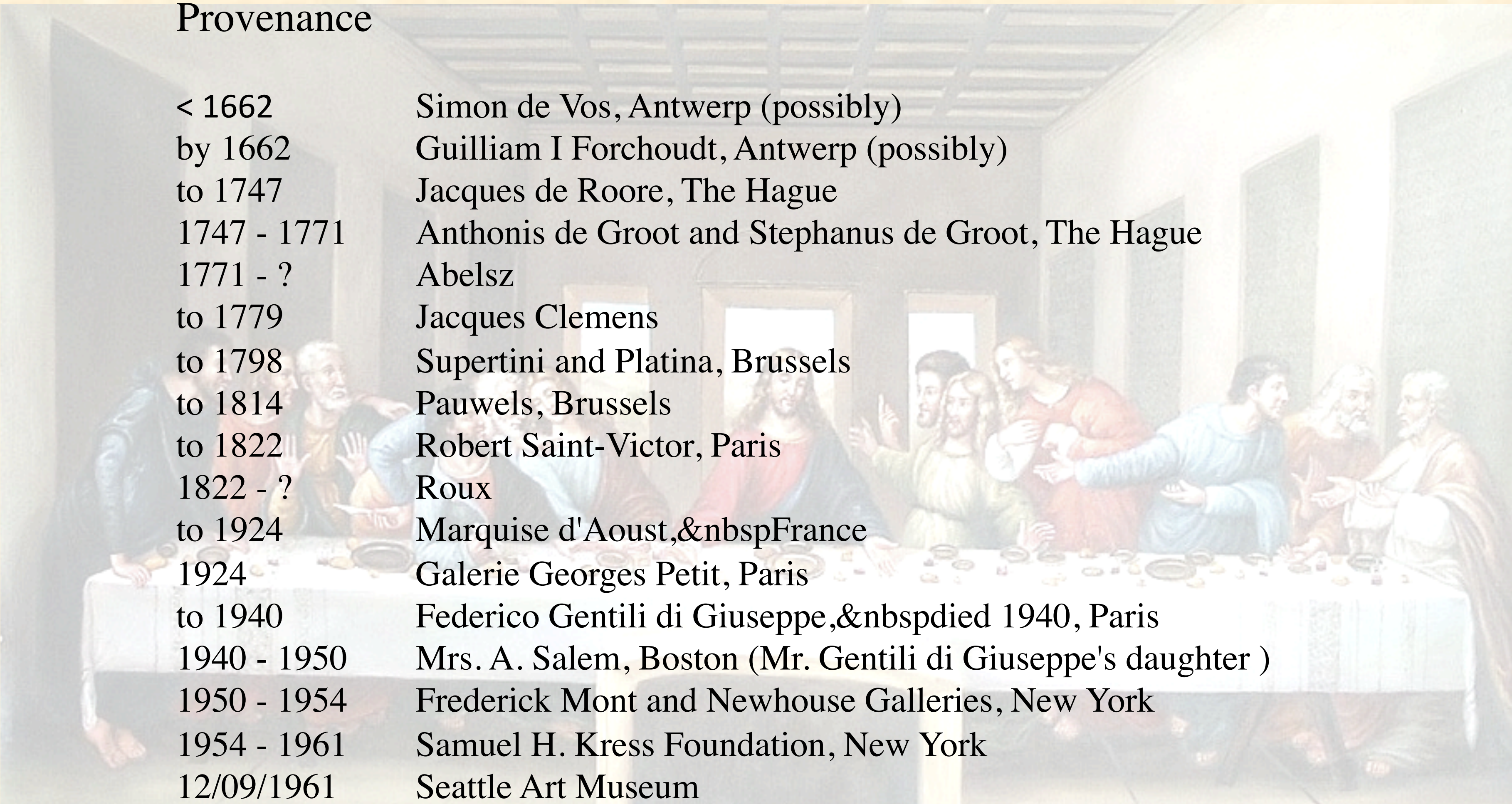
- Data: What do you know?
  - **Muammar Muhammad Abu Minyar al-Gaddafi** June 1942 – 20 October 2011)
- Metadata: How, when, why do you know it?
  - http://en.wikipedia.org/wiki/Muammar_Gaddafi
- Provenance:
  - (cur | prev) 21:11, 21 October 2011 Adriaan Joubert (talk | contribs) **m** (194,106 bytes) *(Minor edits)*
  - (cur | prev) 20:58, 21 October 2011 Luckas-bot (talk | contribs) **m** (194,107 bytes) *(r2.7.1) (Robot: Modifying da:Muammar Gaddafi)*
  - (cur | prev) 20:52, 21 October 2011 Mewulwe (talk | contribs) (194,110 bytes) *(Undid revision 456715406 by Karbuncle (talk))*
  - (cur | prev) 20:34, 21 October 2011 Sundostund (talk | contribs) (194,093 bytes)
  - (cur | prev) 20:10, 21 October 2011 Jim Michael (talk | contribs) (194,094 bytes) *(→Marriages and children: her article says they met in 71)*
  - *And 100's of other updates since 20 October 2011 …*

Provenance: Special Metadata

- From the French word for "source" or "origin"
- The complete history or lineage of a object
- In the art world, provenance documents the chain of ownership of an artifact.
- In the digital world, provenance documents the process that created an artifact.

Example: Art

Example: Art with Provenance



Provenance

< 1662	Simon de Vos, Antwerp (possibly)
by 1662	Guilliam I Forchoudt, Antwerp (possibly)
to 1747	Jacques de Roore, The Hague
1747 - 1771	Anthonis de Groot and Stephanus de Groot, The Hague
1771 - ?	Abelsz
to 1779	Jacques Clemens
to 1798	Supertini and Platina, Brussels
to 1814	Pauwels, Brussels
to 1822	Robert Saint-Victor, Paris
1822 - ?	Roux
to 1924	Marquise d'Aoust,&nbspFrance
1924	Galerie Georges Petit, Paris
to 1940	Federico Gentili di Giuseppe,&nbspdied 1940, Paris
1940 - 1950	Mrs. A. Salem, Boston (Mr. Gentili di Giuseppe's daughter)
1950 - 1954	Frederick Mont and Newhouse Galleries, New York
1954 - 1961	Samuel H. Kress Foundation, New York
12/09/1961	Seattle Art Museum

Example: Data

Example: Data with Provenance

From the camera:

DMC-FZ5
2560 * 1920 2.4 MB JPEG
ISO 80
6 mm
0 EV
f/4
1/250
October 21, 2007 10:06:19 AM

From the user:

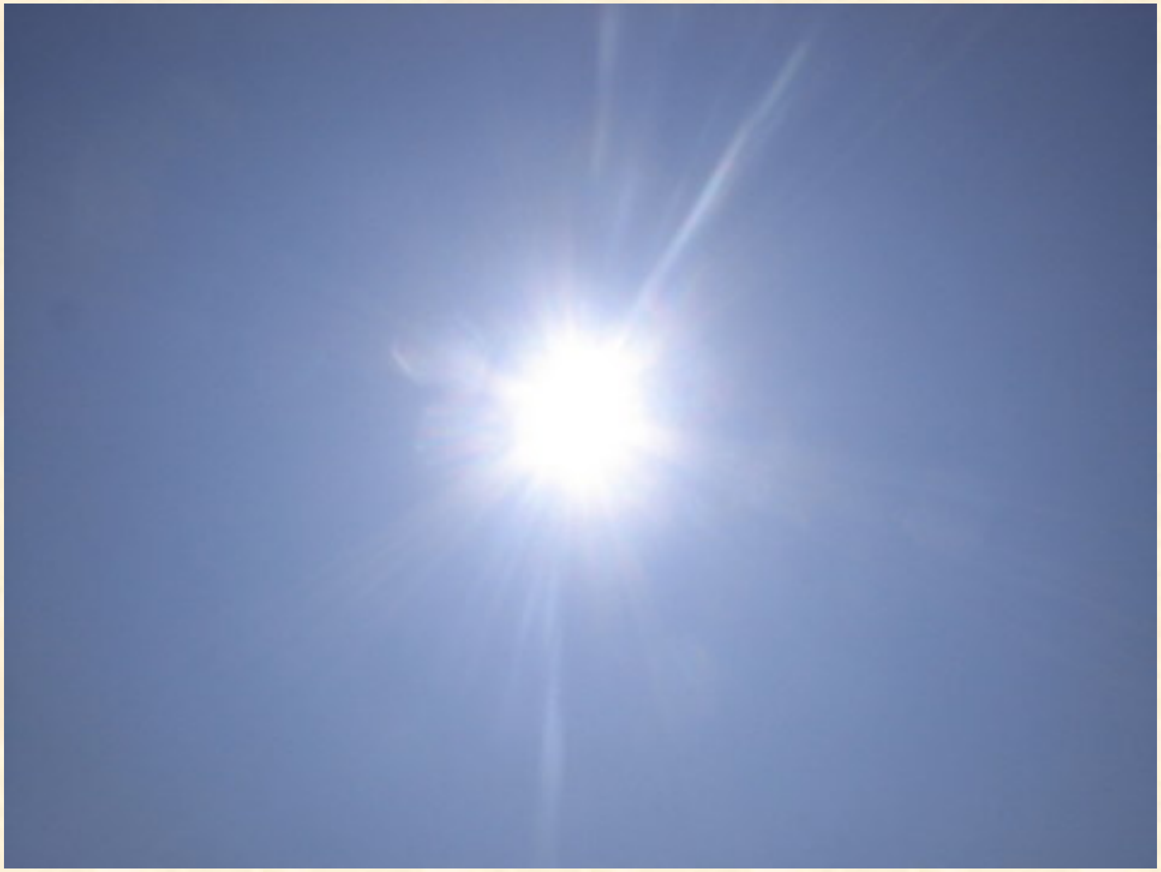
Walden Pond
Jane Beecham

From the software:

All the adjustments and processing that Professor Freeman discussed yesterday.

Example: Day w/out Sunshine

Example: Day with Sunshine



Where Does Provenance Come From?

- From instruments: thermometers, cameras, telescopes, gene sequencers, sensors
- From software: Photoshop, your database, your home-grown tools, the network
- From system software: the operating system, libraries, kernel modules
- From tools: the compiler, the interpreter, your source code control system.
- In other words: it comes from lots of places and is the result of data manipulation other than relational queries.

Why Does Provenance Matter?

- It tells you what **really** happened.
- Consider the following trivial example.
- What is the provenance of LS1.OUT?

```
% cd ~margo/talks/tapp-dir
% ls –l > ~margo/LS1.OUT
```

- Audience Participation
- Given the following:

```
% cd ~margo/talks/tapp-dir
% ls –l > ~margo/LS2.OUT
```

- Is the provenance of LS1.OUT the same as that of LS2.OUT?

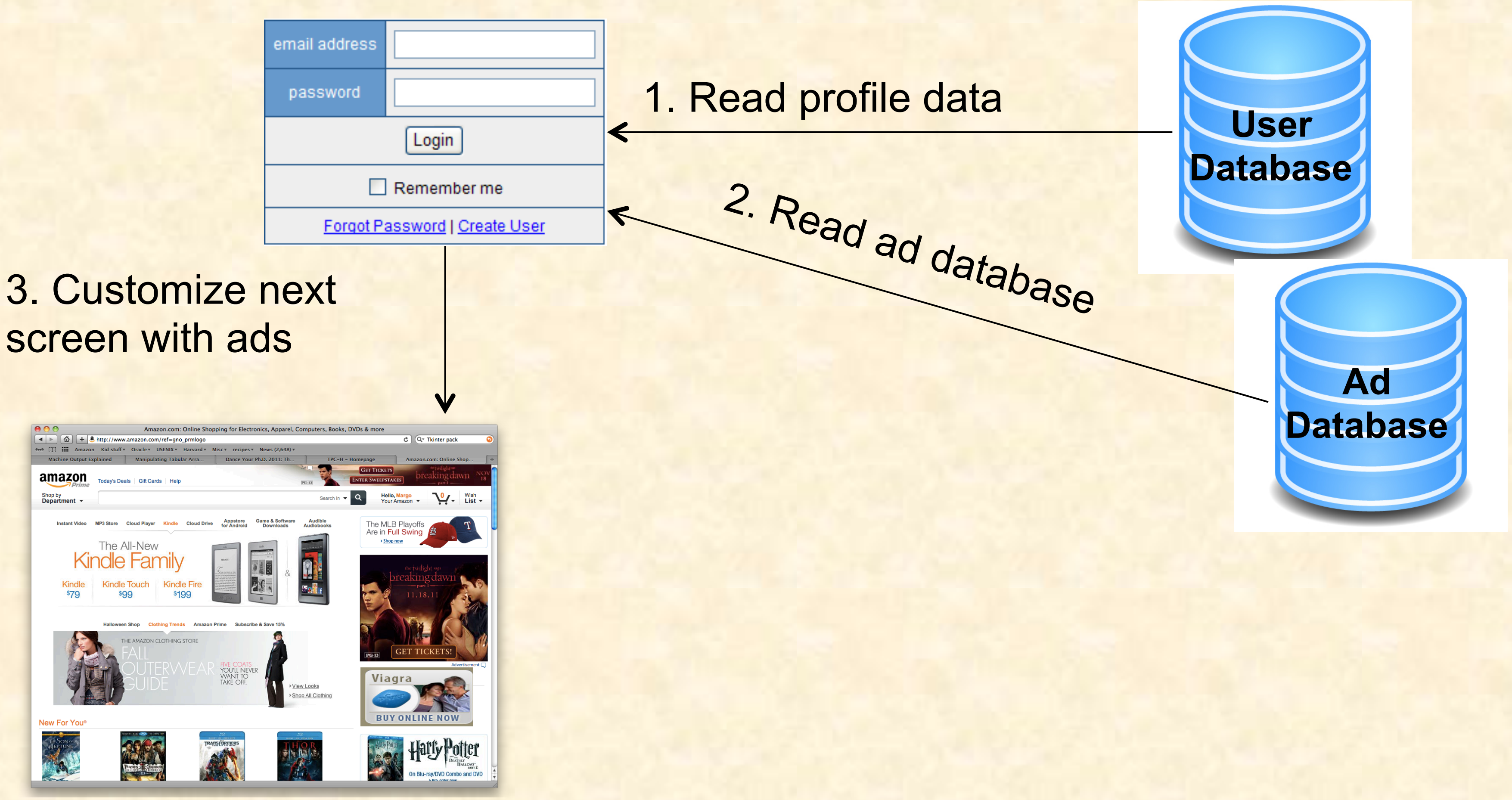
How is Provenance Managed Today?

- Largely manually
  - Embedded in file/directory names
  - Maintained in a lab notebook
  - Entered into a separate provenance database (e.g., MDS)
- Implied
  - "Oh if that data came from that device on this date, then it was running this version of the software."
- Embedded
  - Part of the file format (e.g., XML, FITS, JPEG).
- In a workflow system
  - Interactions expressed as part of the workflow captured in the workflow system.

The Vision: Provenance Everywhere

- **All** data has provenance.
- **Applications** generate provenance.
- **Systems** generate provenance.
- **Users** generate provenance.
- Provenance is:
  - Secure
  - Queryable
  - Globally searchable*
- There are provenance-aware algorithms.

Example: Personalized Ads



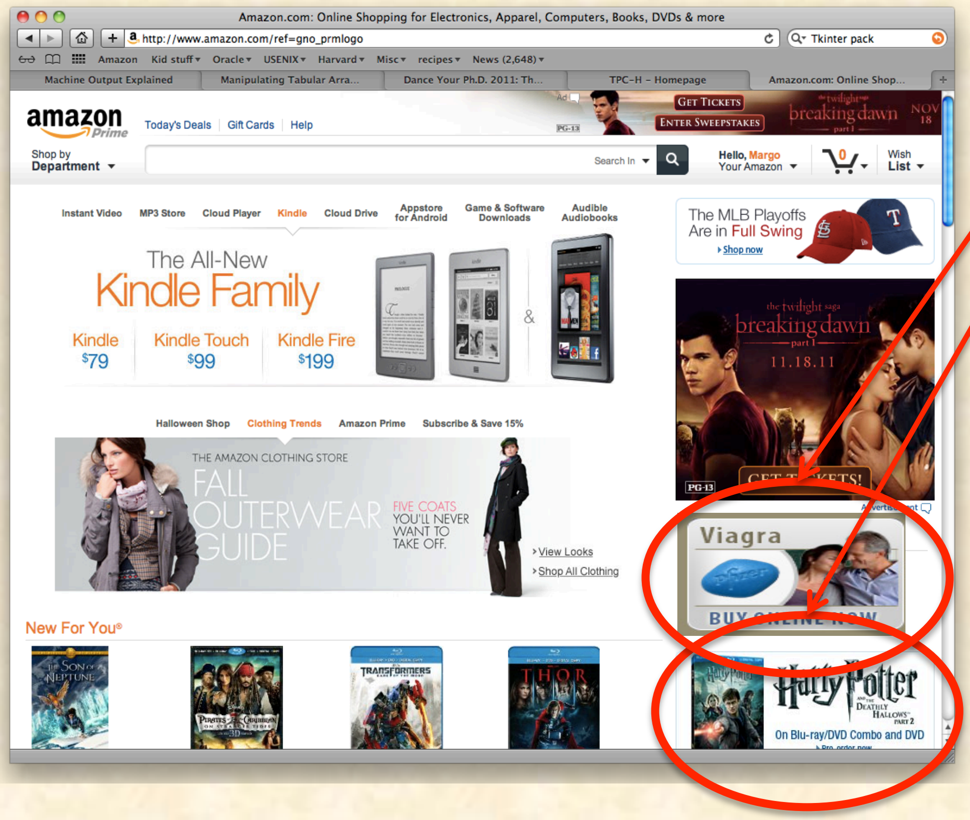
1. Read profile data

2. Read ad database

3. Customize next screen with ads

Questions you Might Ask

- Why did Margo get an ad for Harry Potter?



Why did Margo get an ad for .. Viagra???

How do you Answer?

- What is Margo’s browsing history?
  - Good luck: it has several hundred recent entries (non of which show you anything obvious).
- What do you want to know?
  - What keywords were produced from Margo’s profile?
  - How did those keywords map to ads?
- These are provenance queries!

Provenance across the Software Stack

- What do you need in to answer queries like this?
  - Browser provenance – click on the browser window and ask, "Where did this come from?"
    - The result of a database query
  - Database provenance -- given a query, why is this ad in the result?
    - Formally called "Why provenance"
  - Language provenance – what query was made to the profile database and how did that result get transformed into the query on the Ad database?
    - Data and control flow

Other Provenance Queries

- Why am I getting different results since yesterday?
- Where did this file come from?
  - Assuming it's a virus – "Did I get anything else from the same place I got this?"
- Where did I send this file?
- Show we all my test results produced before I fixed this bug.
- This piece of mail got marked as SPAM, why?

Queries I want to be able to Answer

- You just sent me something – I wonder if it was actually derived from something I sent to someone else.
- Where did this web page come from?
- Where did this piece of spam really come from?
- My machine got compromised – tell me everything that happened.
- A customer is reporting a bug – I'd like to see exactly everything they did before this bug occurred.

First Things First

- Build provenance into your systems.
- Any system – plan for it from day one and

  JUST DO IT
- Now
- Don't wait for some kind of standardization
  - Provenance is a graph
  - Identify objects
  - Store relationships among objects.
  - We'll work out the details later.

State of Provenance

- Most systems don't record provenance, but you're all going to go fix that. Now.
- But ... different layers in your software stack deal with different sets of abstractions and native objects:
  - Operating system: files
  - Database systems: tuples
  - Workflow engines: objects
  - Applications:
    - Variables (from an interpreter)
    - Links or sessions (from a browser)
    - Pieces of text (from a word processor)
- Today, each system is myopic
  - Each system knows about its native objects.
  - Lacks understanding of what happens in black boxes.
  - Lacks connections with things that happen outside of It.

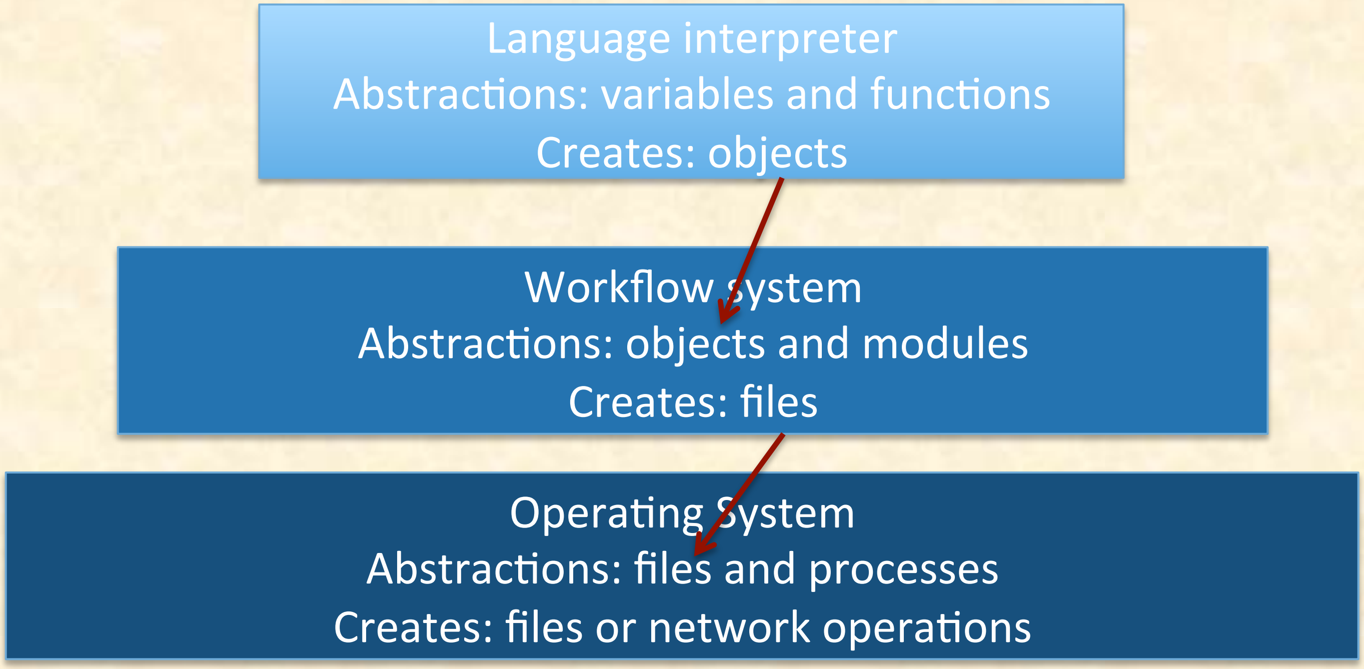
Good news: Objects at different layers are related

- Tuples live in files.
- Files comprise data sets.
- Browsers write files.
- Variables relate to each other.
- Objects may be files, tuples, or data sets.

Relationships between data from different agents are as important as relationships within the provenance of a single agent.

The Solution: Layering & Integration

- Key concept:
  - Each layer collects provenance.
  - Each layer associates its objects with objects in its adjacent layers.



Language interpreter
Abstractions: variables and functions
Creates: objects

Workflow system
Abstractions: objects and modules
Creates: files

Operating System
Abstractions: files and processes
Creates: files or network operations

Making Layering Work

- Can't we just place all provenance in a central repository?
  - NO – that would give you an excuse to delay adding provenance.
  - It wouldn't work anyway
    - All participants would need to agree on naming conventions.
    - Participants would need to be able to generate references to objects created by other participants.
    - What happens when you add a new participant with a new naming mechanism?
- In layering, a participant discloses the relationship between its objects and those in the layer below; that layer then becomes responsible for further transmission.

Layering provides a natural way to transmit and integrate provenance and facilitates query across the layers.

Examples of Layered Systems

- We've built a provenance-aware storage system (PASS).
  - Layers on NFS and/or a cloud storage service.
  - Enables Kepler workflow engine to layer on top of it.
- We prototyped simple database provenance in PostGRES
  - Layered on top of PASS
  - (Did the third provenance challenge with it.)
- We have a provenance-aware python workflow engine (Starflow).
  - Layers on PASS
  - Provides auto-update capabilities
  - Integrates with StarCluster
- Other possibilities:
  - Provenance-aware R
  - Provenance-aware browsing

Provenance Everywhere

- Provenance is useful at all levels of the system:
  - Capture semantics of applications.
  - Capture execution mode of interpreter.
  - Capture system dependencies.
  - Capture source of networked information.
- Provenance lets you make statements about computation.
- Provenance makes research reproducible.
- Provenance is useful for debugging.
- (Secure) provenance is great for auditing.
- Provenance lets you prove things about your computation (maybe).
- Querying provenance reveals sensitive information
  - Oops – that's a bug, not a feature – stay tuned, we're working on it.
- Provenance makes all days sunny.

Just Do It

Margo Seltzer

margo@eecs.harvard.edu

margo.seltzer@oracle.com

http://www.eecs.harvard.edu/~syrah/pass