

---

## 'Slurp' Doesn't Suck

Bryce B. Baril  
SCHARP



## What Isn't Slurp?

---

- Slurp isn't DFsqlload  
... but it is related.
- Slurp isn't the database of record for DataFax Data.
- Slurp isn't intended to replace edit checks.
- Slurp isn't a good thing to search for on Google Images when looking for pictures for a presentation.



## What Is Slurp?

---

- A customizable DataFax mirror in a Relational Database
- Enables incremental updates to your database (Hence the name)
  - Only new or newly changed data is added to the database versus a complete re-copy.
- Written in OO-Perl and built to be subclassed
  - This is how most of the customization is done.
- Stores additional study metadata
  - DFschema (required)
  - DFcenters, DFvisit\_map, etc. (optional)
- Contains active QC processes



## Why Slurp?

---

- SQL access to DataFax Data
  - Ok... so you can get that with DFsqlload...
- So why not DFsqlload?
  - Admittedly some of the DFsqlload shortcomings that led to Slurp's creation have been addressed since the first version of Slurp.
- Incremental updates allow neat-o triggers and near real-time access to live data
- Not limited to an all-or-nothing copy of the data
- Study metadata
- Completely customizable
- Study data can be intrinsically linked to other data sources, including other studies being Slurped



## Why Slurp? (continued...)

---

- Multiple mirrors– test while live!
- Study setup for Slurp ENTIRELY based on study metadata
- Even if you don't use incremental updates, you still can pull dumps from updated plates only
- It has a catchy name!



## The Gory Details (Nap Time)

---

- All studies stored in the same database, using Postgres schemas
- Object-oriented methods for all steps
- Does not start the study servers to check for new data when doing incremental updates
  - A good thing, since this is the most frequent case
- Study permissions can be based on DataFax permissions, or even more granular if you like
- Database layer is fairly abstracted
  - Could be fit to other back-ends with only changing the database layer.



## Case Use: CQC

---

- CQC (Clinical Quality Control) was the main reason for building the Slurp system
- Centered around a lengthy process using multiple databases and calculation systems to provide a report
- Rather than users clicking-and-waiting, installs its own triggers on the DataFax plates using Slurp and calculates results as the data is validated in DataFax
- Triggers fire for each data row slurped, running calculations for only that data
- Report is now constantly available and never more than 10 minutes out of date
- Started testing about a year ago, in production for about 8 months.



## Case Use: CQC cont.

---

- Implementation:
- Subclass of Slurp that contains its own methods for:
  - Setup  
Generates the trigger code based on the study schema and installs triggers and creates additional required tables and views.
  - Slurp  
Has an additional cleanup step in the incremental update
  - Study Removal  
A little additional cleanup is required to remove the extra stuff we added.
- With a fairly minor change to the Slurp system (with a lot of added work outside it) we've radically changed its behavior.



## Caveats

---

- No default handling of DFmissing\_map yet
- Partial dates currently imputed by default
- May fail and raise errors on bad data
  - Duplicate records (Auto-culls true duplicates)
  - Invalid data types
- Incremental copying is actually slower than a full copy (per plate) unless you are using triggers
- SAS ODBC isn't without peril
- Relies on file system access to plate files



## Questions, eh?

---



---

## Looking Ahead



---

## The Future of CDSI?

