IVS Working Group 4: VLBI Data Structures

John Gipson

Working Group Members

Chair	John Gipson
Analysis Coordinator	Axel Nothnagel
Haystack/Correlator Representative	Roger Cappallo
GSFC/Calc/Solve	David Gordon Dan MacMillan
IAA/QUASAR	Sergey Kurdubov Elena Skurikhina
JPL/Modest	Chris Jacobs
Occam	Oleg Titov
Vienna	Johannes Boehm
Steelbreeze Formally MAO, now at GSFC	Sergei Bolotin
Observatorie de Paris/PIVEX	Anne-Marie Gontier
NICT	Thomas Hobiger Hiroshi Takiguchi

Working Group Charter

From the IVS website:

The Working Group will ***examine the data structure currently used*** in VLBI data processing and ***investigate what data structure is likely to be needed in the future***.

It will ***design a data structure that meets current and anticipated requirements*** for individual VLBI sessions including a cataloging, archiving and distribution system.

Further, it will ***prepare the transition capability*** through conversion of the current data structure as well as ***cataloging and archiving software*** to the new system.

Current Format

Mk3 database. Currently 30+ years old. Used to archive and transmit IVS sessions.

A product of its time:

- Designed to run on systems with 20k (!!) memory
- Designed before Fortran had strings

Furthermore…

- Hard to port
- Slow.→ Databases archive information. Superfiles used in analysis.
- Baseline oriented → Tremendous redundancy of some kinds of data.
- Theoretical and observation data mixed.
- Limited user community (20 users?)

Current Format

In spite of its flaws, it has served us well.

- Lasted 30 years—testament to good design.
- Self describing data format.
- Can add new datatypes

Goals

Absolute requirement:

Handle current and anticipated VLBI data needs

Goals

Absolute requirement:

Handle current and anticipated VLBI data needs

Low level goals:

1. Reduce redundancy
2. Ease of access.
3. Speed of access.
4. Different platforms, different languages

Goals

Absolute requirement:

Handle current and anticipated VLBI data needs

Low level goals:

1. Reduce redundancy
2. Ease of access.
3. Speed of access.
4. Different platforms, different languages

High level goals:

1. Flexibility
2. Easy interchange of data.
3. Separation of “observations” from “models” and “theory”
4. Ability to easily access most common parts of the data
5. Ability to access data at different levels of abstraction.
6. Completeness

Goals, Take Two

Goal	Format	Organization	Done?
Low Level Goals			
Reduce Redundancy		✓	
Ease of Access	✓		
Speed of Access	✓		
Many Languages, Platforms	✓		
High Level Goals			
Flexibility		✓	
Easy interchange of sub-sets of the data.	✓	✓	
Separate observables, models, theoreticals		✓	
Separate things that change from things that don't		✓	
Easy access to commonly used parts of the data.		✓	
Data at different levels of abstraction.		✓	
Completeness		✓	

Current MK3 Database

The current Mark3 database is a way of:

1. Storing the data. **Custom Database format.**

Current MK3 Database

The current Mark3 database is a way of:

1. Storing the data.
2. Organizing the data.

Current MK3 Database

Data is organized by Lcodes

1. Type 1 Lcodes. Session data. True for the entire session. (145 items)
   A. Station names, positions.
   B. Source names, positions.
   C. Correlator
   D. …
2. Type 2&3 Lcodes. Observation data. True for an observation. (173 items)
   A. Time
   B. Source
   C. Stations
   D. Station information (Az, El, pressure, calibrations…)
   E. EOP
   F. Observable
   G. Editing
   H. …

Redundancy of Obs-based Data

Many of the observation-dependent data (e.g., Pressure) are really station dependent. This introduces tremendous redundancy.

For an N-Station Scan:
Need to store **N** values for Pressure: One for each station.

Database stores by observation.
Number of observations= number of baselines.

N Stations → N*(N-1)/2 baselines.

For each baseline A-B, stores two values for the pressure: One for station A, one for station B. Total number of values stored:

2* Num baselines = N*(N-1) values.

Data is redundant by a factor of N-1!

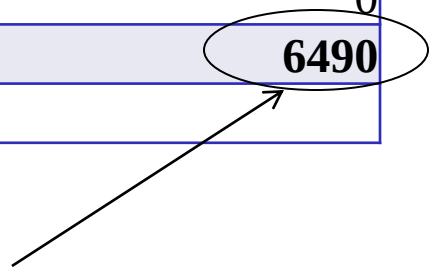
Redundancy of R1360

#Stations	#Scans	#Scans*#Stats
2	366	732
3	232	696
4	158	632
5	71	355
6	35	210
7	0	0
Total		**2625**
Redundancy		

The total amount of station dependent data we need to store is the dot product of the number of stations times the number of scans: 2*366+3*232+….

Redundancy of R1360

#Stations	#Scans	#Scans#Stats	#BL	#BL*2*#Scans
2	366	732	1	732
3	232	696	3	1392
4	158	632	6	1896
5	71	355	10	1420
6	35	210	15	1050
7	0	0	21	0
Total		**2625**		**6490**
Redundancy				



The total amount of station dependent data stored in the database is 2xnumber of observations.

Redundancy of R1360

#Stations	#Scans	#Scans*#Stats	#BL	#BL*2*#Scans
2	366	732	1	732
3	232	696	3	1392
4	158	632	6	1896
5	71	355	10	1420
6	35	210	15	1050
7	0	0	21	0
Total		**2625**		**6490**
Redundancy				**2.5**

The redundancy is found by dividing the data that we actually store by the data that we need to store.

Redundancy of RDV73

#Stations	#Scans	#Scans*#Stats	#BL	#BL*2*#Scans
2	134	268	1	268
3	149	447	3	894
4	55	220	6	660
5	42	210	10	840
6	30	180	15	900
7	24	168	21	1008
8	33	264	28	1848
9	34	306	36	2448
10	40	400	45	3600
11	60	660	55	6600
12	39	468	66	5148
13	22	286	78	3432
14	16	224	91	2912
15	11	165	105	2310
Total		**4266**		**32,868**
Redundancy				**7.7**

Redundancy of Stat32_6_2p1D0ln

#Stations	#Scans	#Scans*#Stats	#BL	#BL*2*#Scans
2	431	862	1	862
3	261	783	3	1566
4	145	580	6	1740
5	80	400	10	1600
6	49	294	15	1470
7	15	105	21	630
8	6	48	28	336
9	34	306	36	2448
10	97	970	45	8730
11	200	2200	55	22000
12	232	2784	66	30624
13	362	4706	78	56472
14	516	7224	91	93912
15	596	8940	105	125160
16	601	9616	120	144240
17	683	11611	136	185776
18	639	11502	153	195534
19	274	5206	171	93708
20	82	1640	190	31160
21	10	210	210	4200
Total		**69,987**		**1,002,168**
Redundancy				**14.3**

How to Reduce Redundancy

Introduce two new types of data:

1. Station-scan data depends only on the station and the scan.
2. Scan data depends only on the scan.

This requires modest additional bookkeeping:

1. Connects observations to scans.
2. Connects observations to stations.

This can be done, for example using the time-tags of the data.

How to Reduce Redundancy

Introduce two new types of data:

1. Station-scan data depends only on the station and the scan.
2. Scan data depends only on the scan.

This requires modest additional bookkeeping:

1. Connects observations to scans.
2. Connects observations to stations.

This can be done, for example using the time-tags of the data.

With a fair amount of work, we could do this using the *present* Mark3 database format.

Goals

Goal	Format	Organization	Done?
Low Level Goals			
Reduce Redundancy		✓	✓
Ease of Access	✓		
Speed of Access	✓		
Many Languages, Platforms	✓		
High Level Goals			
Flexibility		✓	
Easy interchange of sub-sets of the data.	✓	✓	
Separate observables, models, theoreticals		✓	
Separate things that change from things that don't		✓	
Easy access to commonly used parts of the data.		✓	
Data at different levels of abstraction.		✓	
Completeness		✓	

Ease of Access

- Ability to easily access data on different platforms.
- Ability to use different languages, platforms.
- Speed

Ease of Access

- Ability to easily access data on different platforms.
- Ability to use different languages, platforms.
- Speed

There are many data storage formats that meet these goals: NetCDF, CDF, HCDF.

Ease of Access

- Ability to easily access data on different platforms.
- Ability to use different languages, platforms.
- Speed

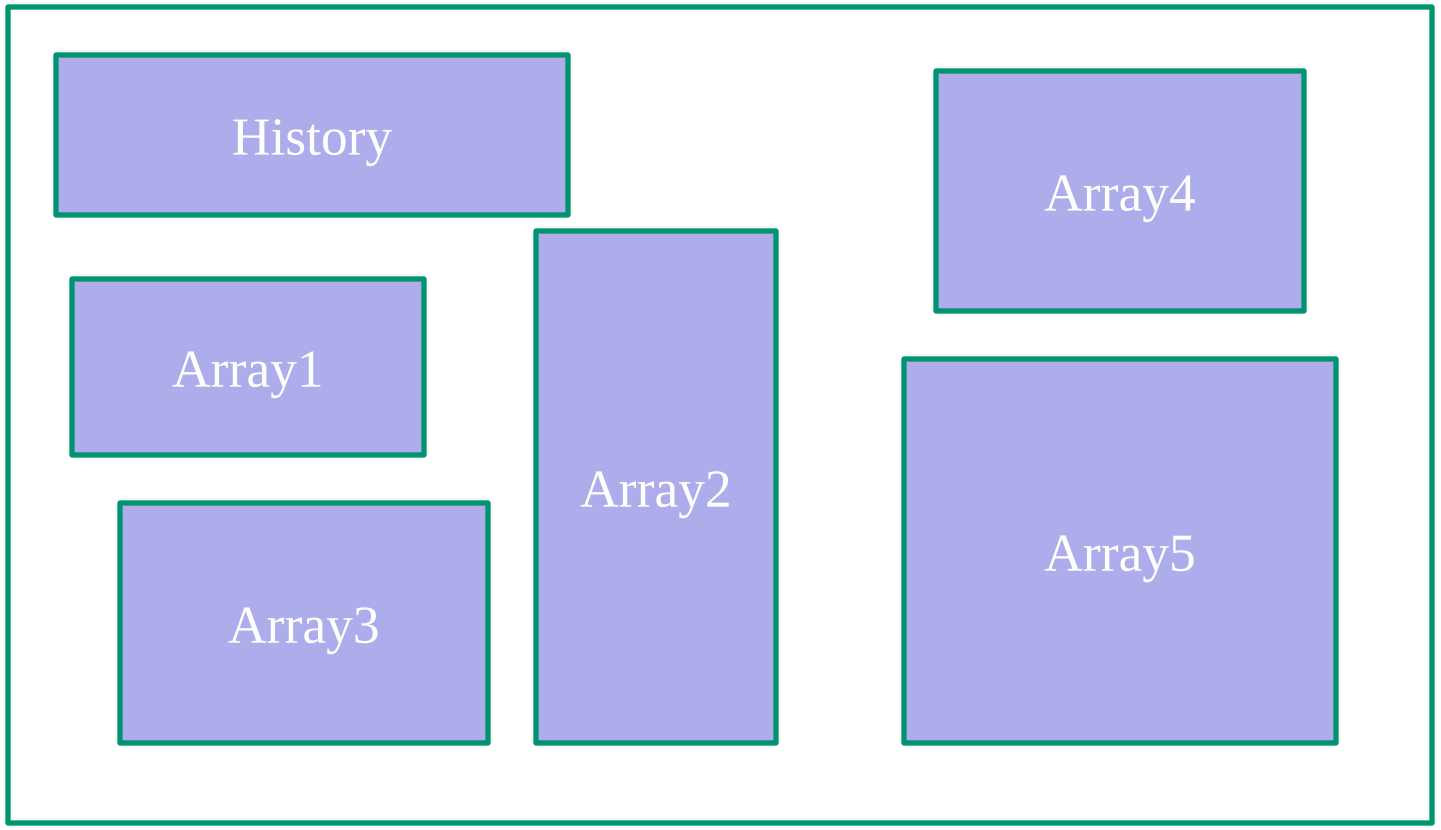
There are many data storage formats that meet these goals: NetCDF, CDF, HCDF.

Recommend using NetCDF4.

1. Wide user community.
2. Many libraries and utilities.
3. Compatible with HCDF.
4. On the fly data-compression.

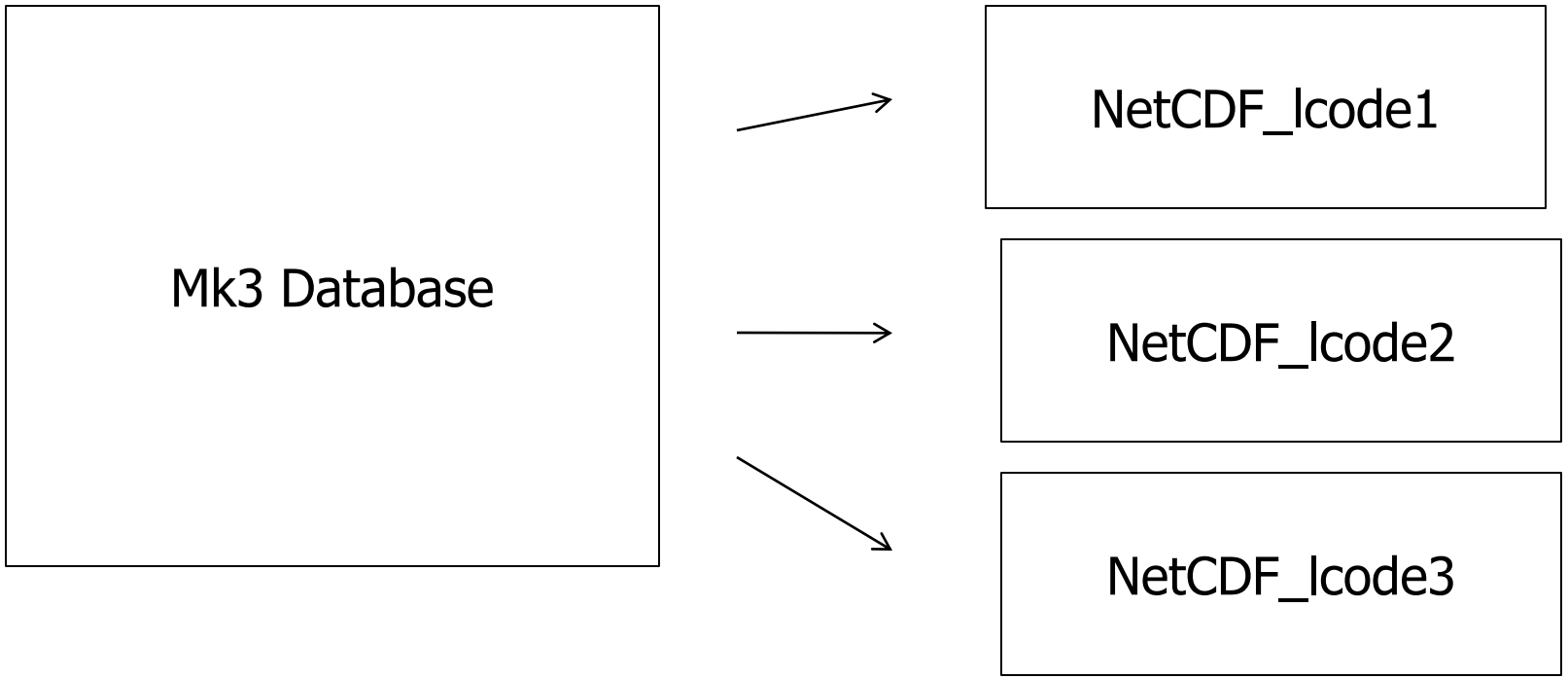
This also makes it possible to access sub-sets of the data.

What does a NetCDF File Look Like?



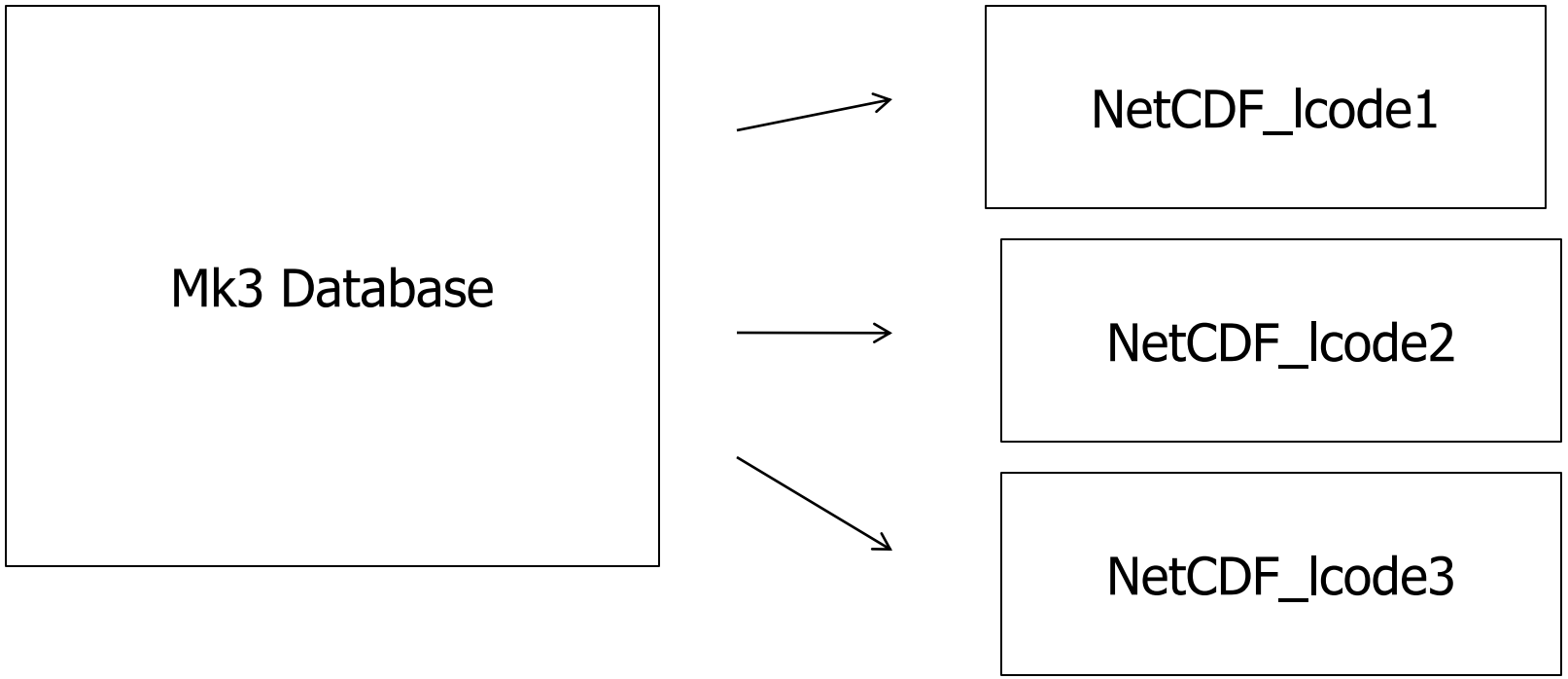
A NetCDF file can contain an arbitrary number of arrays. The arrays can differ in dimensions and type (byte, short, integer, real, double). The arrays can have attributes like name, unit, long-name, description associated with them.

Hobiger's NetCDF database implementation



There is a 1-1 mapping between lcodes and NetCDF arrays.

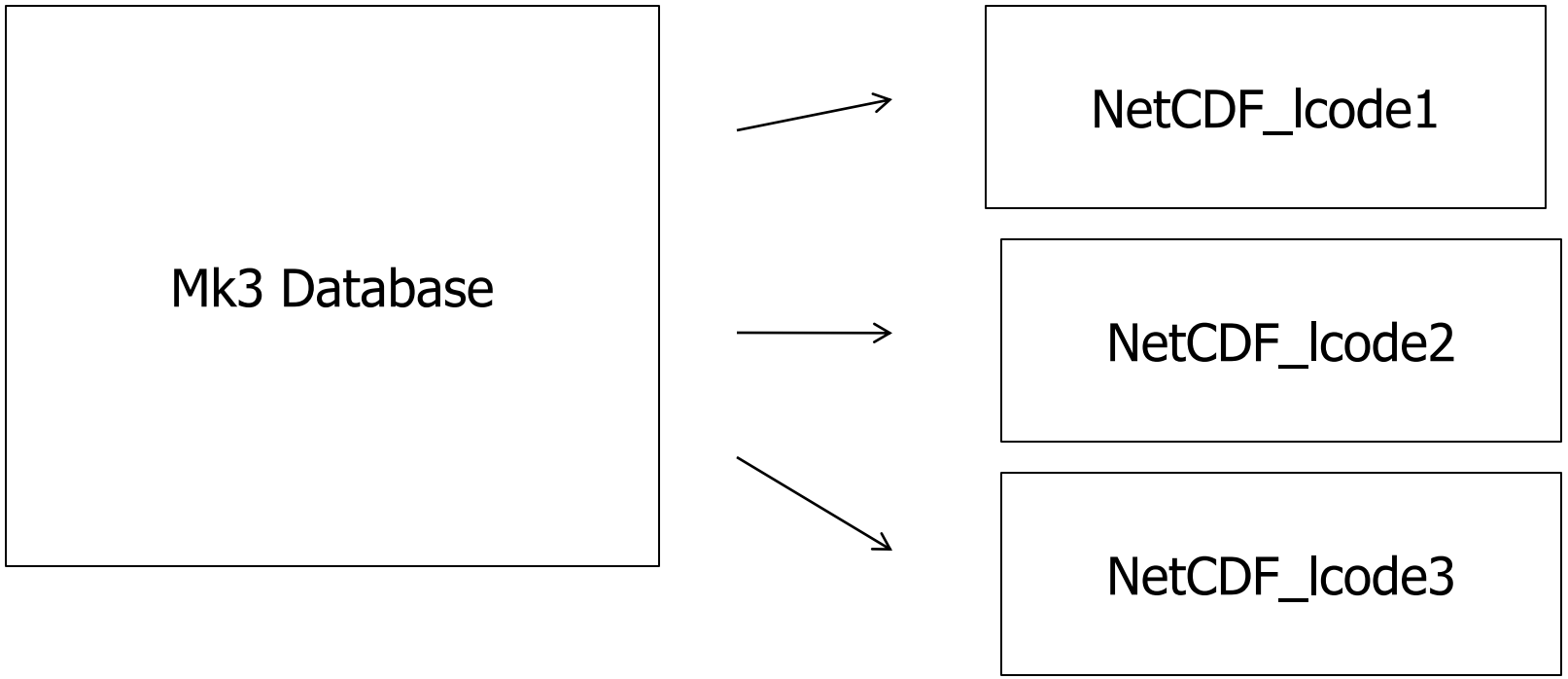
Hobiger's NetCDF database implementation



There is a 1-1 mapping between lcodes and NetCDF arrays.

Can also go from NetCDF files to Mk3 database.

Hobiger's NetCDF database implementation



There is a 1-1 mapping between lcodes and NetCDF arrays.

Can also go from NetCDF files to Mk3 database.

We could store Mark3 databases in NetCDF format.

Goals

Goal	Format	Organization	Done?
Low Level Goals			✓
Reduce Redundancy		✓	✓
Ease of Access	✓		✓
Speed of Access	✓		✓
Many Languages, Platforms	✓		✓
High Level Goals			
Flexibility		✓	
Easy interchange of sub-sets of the data.	✓	✓	
Separate observables, models, theoreticals		✓	
Separate things that change from things that don't		✓	
Easy access to commonly used parts of the data.		✓	
Data at different levels of abstraction.		✓	
Completeness		✓	

Splitting the VLBI Session data

The Mark3 database format was designed so that *all* data pertaining to a session resides in one file.

Advantage: "one-stop-shopping".

Disadvantages:

1. Database contains data of interest only to calc/solve users.
2. Anytime anything changes—calibrations, ambiguities, models—you need a new version of the database.
3. Anytime something is added to the database, you need a new version of the database.
4. Databases contains lots of obsolete information that is no longer used.

Splitting the VLBI session data

Proposal: Gather data that is similar in

- Scope
- Origin
- Physical effect
- Frequency of change.

Store in its own file.

Splitting the VLBI session data

Proposal: Gather data that is similar in scope, origin, physical effect, frequency of change. Store in its own file.

1. Experiment info: everything known about experiment beforehand.
2. Atmospheric delay
3. Met data
4. Calibrations
5. Physical and geophysical effects calculable beforehand: relativity, tidal ocean loading, etc.
6. Physical and geophysical effects calculable afterwards: atmosphere loading, hydrological loading, etc.
7. Observables and commonly used observation related data.
8. Editing and Ambiguity
9. Less commonly used observation related data

Splitting the VLBI session data

Advantages:

1. Easy to add new data types.
2. Items that are not expected to change are separated from items that may change.
3. Data is separated from models.
4. This approach lends itself to building up the session piece by piece.
5. We delay discussion of what the VLBI2010 observable format should look like.
6. Commonly used data is separated from less commonly used data.
7. This enables easy testing of new models.
8. As models improve, they can be easily incorporated.

Organizing the Data With Wrappers

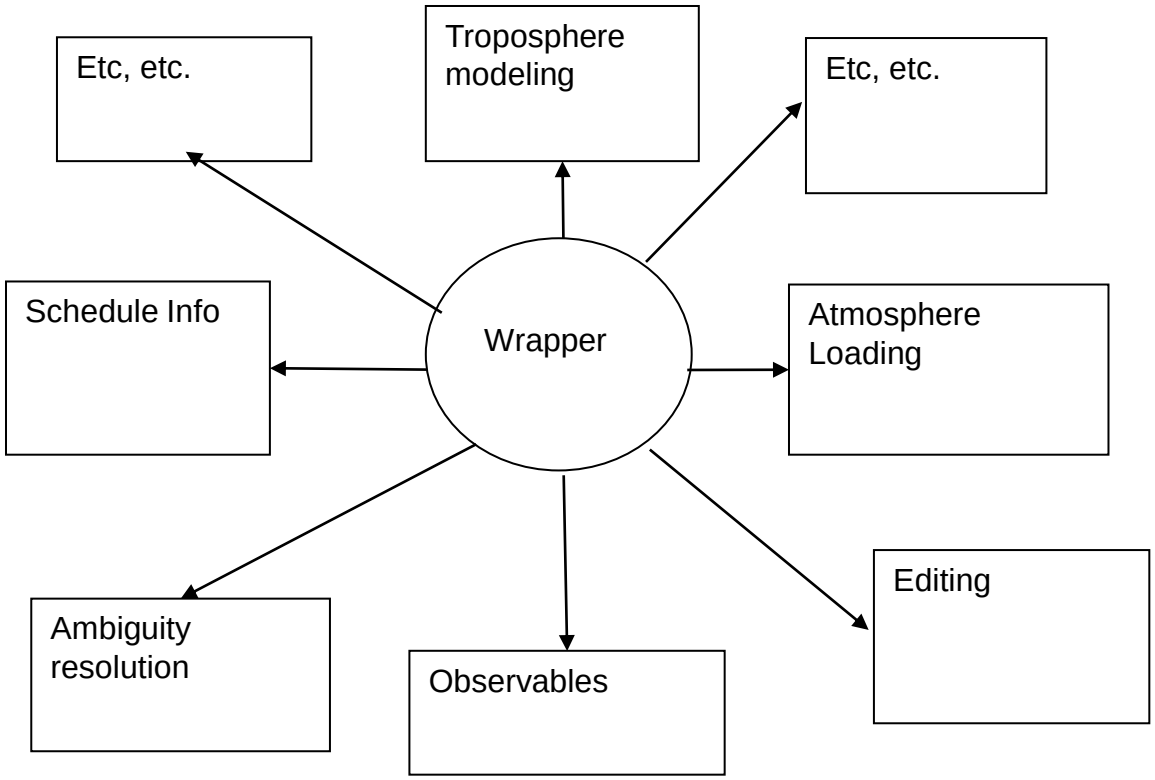
Now that we have split the data, how do we gather it up?

We *wrap* it up using wrappers.

A wrapper is an ASCII file that contains pointers to files that contain data about a session.

A pointer is an instruction about where to find the data. Simplest case is a location on the disk.

Wrappers Organize the Session Data



Goals

Goal	Format	Organization	Done?
Low Level Goals			✓
Reduce Redundancy		✓	✓
Ease of Access	✓		✓
Speed of Access	✓		✓
Many Languages, Platforms	✓		✓
High Level Goals			✓
Flexibility		✓	✓
Easy interchange of sub-sets of the data.	✓	✓	✓
Separate observables, models, theoreticals		✓	✓
Separate things that change from things that don't		✓	✓
Easy access to commonly used parts of the data.		✓	✓
Data at different levels of abstraction.		✓	✓
Completeness		✓	✓

Kinds of Wrappers

1. Bare bones wrapper. Essentially points to information contained in NGS cards.
2. Complete session wrapper. Contains information in NGS cards, also information about where to find correlator output file. Useful for experts.
3. Private wrappers. Used by researchers to test different models.

Advantages and Uses of Wrappers

1. Can specify “IVS-standard” session.
2. Can incorporate alternative models by changing a pointer:
   met_xR1345_kNMF_cGSFC_v01.nc →met_xR1345_kVMF_cVIEN_v01.nc
3. Researchers can use their own “private” wrappers to test alternative models.
4. Groups can swap editing and ambiguity information.
5. Can easily add new data types.
6. Can use this to preserve history of processing.

Where Are We and Next Steps

Draft proposal circulated in July 2009.

Positive feedback.

Conversion of Mark3 database to New Format.

1. Partial utility written in July 2009.
2. Converts ~30% of the lcodes. Everything in NGS cards and more.
3. Complete conversion utility July 2010

Where Are We and Next Steps

Conversion of Software to use New Format.

1. **steelbreeze**
   A. Partial conversion Sep 2009. Uses NetCDF as storage format.
   B. Timing penalty of 40 microseconds/obs. No optimization.
      6 million obs * 40 microseconds=240 seconds penalty=6 minutes.
2. **calc/solve**
   A. Replacement of superfiles by new format: July 2010.
   B. Replacement of databases by new format: January 2011.
3. **VieVS** . (Vienna VLBI Group)
   A. Ability to read new format: March-June 2010.
4. **Occam, Modest, other software?**

Where Are We and Next Steps

Format Finalized

Finalize format July 2010

Expect to learn a lot in **calc/solve, VieVS** software development.

May lead to some changes in the design.

Distribution in New Format

1. July 2010. Begin distribution in new format on trial basis.
2. January 2011. Switch from database to new format.

Further Feedback

Meeting this Thursday—encourage participation of everyone.