

# The LSST Data Model

# Our Charter

## Introduction

Do we need an LSST datamodel?

What are moving object's especial problems?

What organisation will enable our science?

Will a single catalogue suffice?

How do we control reprocessing?

What query language should we adopt?

What DB architecture should we adopt?

What synergies does LSST have with other projects?

We have often spoken loosely of “LSST data”. But what is it, exactly? It is clear that there are many different levels, including raw camera data; processed camera data (eg crosstalk removal, debiasing, etc); calibrated images (eg corrected for photometric zeropoint; de-fringed; WCS added, etc); combined images (added; subtracted); and catalogs of “objects”, some of which may be complex (eg an asteroid track or a resolved galaxy).

This list includes a variety of different things:

- Telescope engineering data and raw images, both with metadata (e.g. telescope pointing, TAI, image sizes)
- Processed imaging data and derived metadata
- Derived *detections* and their derived parameters
- Catalogs in a database
- Information extracted from a database and processed further

These are increasingly removed (abstracted?) from the detectors. It is probable that we will want to treat different levels in this hierarchy differently, with the levels closer to the camera being more strongly typed than the ones closer to the end user.

We will allow for additional parameters associated with data and properties associated with objects, but we shall not necessarily allow extensions to the way in which the properties are defined, represented, or calculated.

We need to define in-memory representations for at least some datatypes as they pass through the pipeline APIs; this “standardization” needs to occur before we define the APIs (e.g. we may know what our representation of an image is before we are sure what the interface is to the debias-flatfield-linearise module).

It seems intuitively appealing to have some kind of base data model for LSST that will underly any science pipeline, with the provision that it can be readily extended by different groups as needed to encompass their science. Is this sensible? If so, can we come up with a rough sketch of the base model, and a couple of examples of extending it?

We agree that we need a single LSST datamodel that we shall design and build to. This datamodel needs to cover three reasonably distinct parts of the project: the pipeline processing that populates the science database; the schema used in that database; and the analysis codes that are run on information extracted from that database. The word *database* should not be taken to imply that we believe that the LSST data must be stored in a classic RDBMS.

The architecture of the LSST data systems assumes that we have a modular system, and this *requires* that we define what the data looks like. The fundamental unit of modularity is set by where the data representation is changed (e.g. from a raw image to a trimmed flattened one; or when an image is projected onto a tangent plane; or where an object finder generates a list of detections; or when data is inserted into or extracted from a database).

We need to think about how to define a data-model that is sufficiently flexible to support our science, but not foolishly flexible. For example, we certainly need to be able to measure additional parameters of objects, and we don't need to support time-tagged X-ray data. We should avoid generality for the sake of generality, unmotivated by compelling science requirements.

We should allow for additional parameters associated with data and properties associated with objects, but we need not necessarily permit extensions to the way in which the properties are defined or calculated. There also need to be semantic limits on how much change an data-object can undergo before it is no longer just the original object.

Science concerned with moving objects (solar system objects, stellar proper motions) will be a particular challenge in creating a flexible and efficient data model. What's known about how to do this? What needs research?

This is solidly in the database schema part of our task. The two examples (solar system [SS] and stellar velocities) are fundamentally different — once we've seen a star (even Barnard's star) we know which other (non-SS) objects it'll be associated with, whereas SS objects wander all over the sky.

We need to think about how ambitious we want to be in supporting SS discovery in the database. It's one thing to link together observations of an object with an externally defined orbit; but it's quite another to associate detections via a SQL query. We should probably expect to write a moving object pipeline to *find* SS objects, but the database must support their study and retrieval

Some kinds of science will be practical only if data is organized in particular ways. Calculating correlation functions for galaxies is perhaps one example. What other challenges can we foresee? What are the implications for our data organization?

Again, how ambitious should we be within the database, and how many of these calculations should we expect to support via post-database pipelines? At what point should we throw up our hands and invoke the VO? At least some of the following classes of queries should probably be supported directly by the database schema.

- The properties of an **object's** individual **detections**
- The mean properties of an **object**
- The mean/individual properties of an **object's** neighbours
- The higher-order moments of an **object's** properties
- The mean properties of the mean/individual properties of a set of **objects** grouped by some criterion (e.g.  $l/b$ )

- The higher-order moments of the mean/individual properties of a set of **objects** grouped by some criterion (e.g. ra/dec; l/b)
- The assembly of individual **detections** into **objects**

Can we imagine creating only a single LSST object catalog, encompassing all the properties that people are likely to care about, or are we certain to need multiple catalogs?

One question that Tim raises is whether a single set of measured properties of objects will satisfy all of our customers. RHL thinks that the answer is, yes, but that we may have to educate them.

We need to decide whether we'll want to re-import derived catalogues (e.g. of RR Lyraes, SNe, or QSOs) into our database. The construction of some of these catalogues (e.g. via archival searches for newly detected TNOs) should probably be left to the VO.

We can probably come up with a single catalogue for detections (although this is complicated by issues of object detection algorithms, techniques for handling multi-band data, and deblenders). We will also have to come up with a catalogue of **objects** (with associated parameters, derived from the constituent detections, e.g. a mean magnitude), although the definition of the members of this catalogue is not uniquely defined.

We may also wish to provide extra information (e.g. spatio-temporal indices) to allow users to construct their own object catalogues.

This question of identity also impacts the design of the pipelines; we shall probably need to feed known **objects** back into the measure-properties module(s) of the main pipeline.

It has been the experience of every large survey that data needs to be processed multiple times. This tends to cause a rapid increase in the complexity of the data, due to the need to be able to trace the origins (provenance) of any entry in the data structure. What options do we have for making this problem manageable?

The act of reprocessing doesn't make the data more complex, what makes it complex is simultaneously supporting the old version of the catalogues (which have generated Nobel-prize-winning research) and the new one (which is more accurate, or has added valuable features).

LSST does have a responsibility for being able to prove the ancestry of any piece of information in the LSST databases, and our final datamodel must include sufficient metadata (code versions; file CRCs; tunable parameters) to support this. This metadata will probably be primarily associated with exposures not individual detections, and that it will therefore not significantly alter our storage requirements.

What sort of API will we need to query the data? Should we continue to think SQL?

The LSST community need to decide what sort of queries we'll be supporting, and how efficiently we support them. SQL (possibly with some extensions as regards group by clauses) can *express* a large fraction of these queries, but the question as to whether a sufficiently powerful *implementation* will exist is out of our domain of expertise.

As discussed above, there may be scientific enquiries that we would like to support but which will in fact be carried out by post-database pipelines (e.g. linking solar-system detections into orbits).

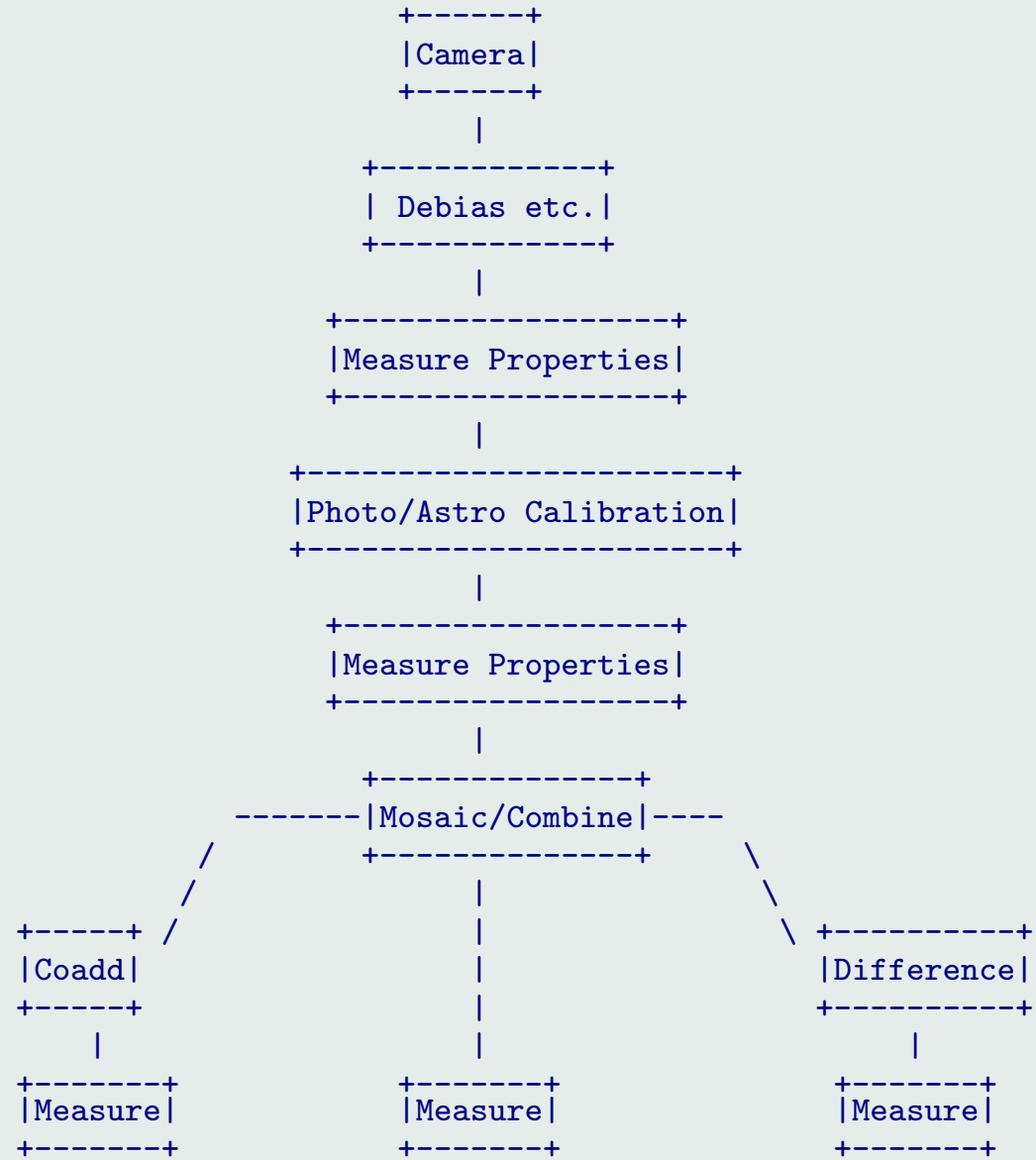
Existing surveys have used a variety of technology for implementing their data structures, including commercial relational DB's; object oriented DB's; and "roll-your-own" solutions. Looking ahead to 2011, how do we expect the available technologies to stack up compared with our needs for LSST?

We are not sure that this question needs to be answered at this time, or by us. When one designs a conceptual database schema it doesn't matter if the data is in **SQL** or **FITS** tables, and even with an **OODB** model the underlying design is similar.

How can we identify and exploit synergies with other projects?

There are many projects with similar data and requirements to LSST (the closest is probably Pan-STARRS), and we should certainly be able to share algorithms with them. If we wish to also share *code*, we will have to share a concrete realisation of a joint datamodel (i.e. libraries and header files). It is an interesting question as to whether it is really possible to define a sufficiently general data model to make this possible, and whether it would be worth the effort involved.

# LSST Dataflow



# A Strawbear LSST Model

## Data Types during Pipeline Processing

Here's a rough classification of all the data that we're going to need to create and track.

**Raw Imaging data** CCD images and associated metadata, including e.g. camera geometry

**Site/Telescope metadata** Pointing; Servo loop errors; Camera temperatures; Atmospheric pressure

**Image Calibration Data** Flats; Biases; Linearity curves; Fringe Frames; ...

**Object catalogues** Astro/Photometric catalogues (which we'll bootstrap eventually), and a known-objects catalogue (e.g. from the coadded images). The parameters associated with these objects will differ from catalogue to catalogue.

**Imaging data with instrumental signature removed** Debiased  
Flat fielded, ... and associated metadata

**Focal Plane Model** A model of the current focal plane distortion, and of the PSF variation across the camera

**Calibrated data** Astro/Photometric calibrated imaging data (maybe just the flattened data with additional metadata appended)

## **Coadded data from multiple near-simultaneous exposures**

LSST may not need this (Pan-STARRS does), but depending on how the moving/transient object detection is done it might.

This may follow the same data model as the preceding, with added metadata.

## **Coadded data from many non-coeval exposures**

This may follow the same data model as the previous two types. Note that we need to define a description of the projection used.

## **Object detection information**

I.e. the pixel-footprint for all significant detections in some of the preceding types.

Depending on the deblending algorithms adopted, this information will probably also have to include information about pixel *intensities*. I suspect that we'll need to carry

information about the individual peaks found within each object.

This information is needed as part of the audit trail, and also as an input into the code that measures object properties

**Object Parameters** I don't expect that there is one set of parameters that we'll all agree on, but it isn't hard to support their union.

We additionally will probably wish to store other items which *could* be recalculated, and which are not used outside a single module; an example would be difference imaging kernels. We can finesse this question by calling these items part of the metadata.

## Data Types in the Database

We have four types of information to track:

**Metadata** Tracking everything that we've done, including exposure footprints, calibration information, and the history of our processing.

**Measurement-Sets** Properties measured at a point, either corresponding to a detection, or to a catalogued-object

**Objects** The collection of a set of **measurement-sets** into a what we believe to be by a single physical entity (e.g. star, galaxy, asteroid, Exciting Transient)

**Truth** We are going to need to run simulations, both end-to-end and injection of fake objects into real data. The **truth** tells us what these false inputs were.

There are (at least) two approaches to handling `objects`; we can statically associate `measurement-sets`, or we can augment the metadata to allow us to build `objects` dynamically. In fact, we may wish to define `objects` as collections of either `measurement-sets` or `objects`.

These two approaches are not orthogonal; for example, the SDSS pipeline associates a set of ugriz magnitudes into a single entity, and the SDSS database then provides a spatial index to allow the users to associate these measurement-sets with each other to e.g. study variability.

## Post-Database Data Types

We shall need to export data in VO-approved formats; we consider this to be an external interface, rather than a data-model, issue.

We have imagined exporting data from the database and allowing our user community to run analysis codes on the exported data. We need to define a data model for this activity; it is not obvious that the VO data model will be appropriate for this activity, but if possible we should adopt it.

For any of these pipelines whose results we wish to re-import into the database (e.g. the moving object pipeline), we shall have to define their outputs within the datamodel.