LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-207256

Conceptual Design of a Prototype LSST Database

S. Nikolaev, M. E. Huber, K. H. Cook, G. Abdulla, J. Brase

October 15, 2004

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Conceptual Design of a Prototype LSST Database[1]
UCRL-TR-207256

S. Nikolaev, M. E. Huber, K. H. Cook, G. Abdulla, J. Brase

Lawrence Livermore National Laboratory, Livermore, CA 94550

ABSTRACT

This document describes a *preliminary* design for Prototype LSST Database (LSST DB). We identify key components and data structures and provide an expandable conceptual schema for the database. We discuss the potential user applications and post-processing algorithm to interact with the database, and give a set of example queries.

1. Introduction

The purpose of this document is to establish a preliminary set of functional requirements for the LSST Database. These requirements will be used to design one or more prototype systems that will explore design and implementation issues for the final LSST DB design, to be completed in a few years.

The high-level structure for the LSST data flow is displayed in Figure 1. The major constituting blocks for the structure are:

- *Image reduction pipeline(s).* LSST data reduction will be implemented in a set of image reduction pipelines. These will include an object detection pipeline which will detect and characterize point and exteneded objects in a single image and a difference image analysis pipeline that will detect changes between a new image and a reference image due to variable objects, transients, and moving objects. The pipelines will be supplemented with efficiency monitoring data (artificial objects with known characteristics) that will allow operators to monitor the pipeline for correct operations.
- *LSST DB*, the prototype of which is described in this document;

[1]Version 2.00; 01/27/05

- *Post-processing algorithms* that interact with the LSST DB by both reading and writing the attributes. We briefly discuss some specific algorithms in §4. The algorithms are designed to facilitate queries and allow analysis of science data;
- *Interactive user queries and data mining tools.* The data mining tools represent a layer between a user issuing a query and the database itself. Its primary purpose is to use some sophisticated techniques to make querying the database more efficient.

The rest of this document is organized as follows: in §2 we present an expandable schema for LSST DB that provides a basic framework for future updates. We discuss the attributes in §3, focusing on the extensibility issues. In §4 we discuss the user applications layer in order to gain insight into possible queries that may originate within this component. The example queries themselves are presented in §5. They include both the interactive queries by the DB users, and the implicit queries exercised by post-processing layer (e.g., cone search with time and magnitude constraints to be employed by the asteroid linkage algorithm). Finally, in §6 we focus on data volume, scaling and indexing issues for the DB. Throughout this paper, we use a convention that the database tables and attributes are in **boldface**. To indicate an attribute in a certain table, we use **Table::Attribute** notation.

2. LSST DB Schema

The basic structure of the database is derived from the well-defined hierarchy in the data flow: "Image → Detections → Objects → Classified Objects", which is illustrated in Figure 2. This sequence occurs naturally in time-domain astronomy (e.g. SuperMACHO Database), and lends itself as the basis for the table structure.

The extraction of Detections from the Images usually takes place at the latter stages of the image reduction pipelines, once the necessary preliminary stages (flatfielding, etc.) have been completed. The Detections are unified into Objects by clustering algorithm, which may run either concurrently with the data ingestion into LSST DB, or at specific recurring time intervals on a "frozen" database state (e.g. between the observing runs). Finally, the object classifications are deduced using various analysis codes (lightcurve, motion, etc.) based on the existing object separation.

The preliminary schema for the LSST DB is shown in Figure **??**. The **Image** table stores information pertaining to the observation conditions and settings, plus it also contains the links to calibrated images (it may also store pointers to template images and subtracted images). For each sky frame taken, the information stored in the **Image** table is shared by all entries in the detection tables **DIADet** and **ObjDet**. These two tables result from

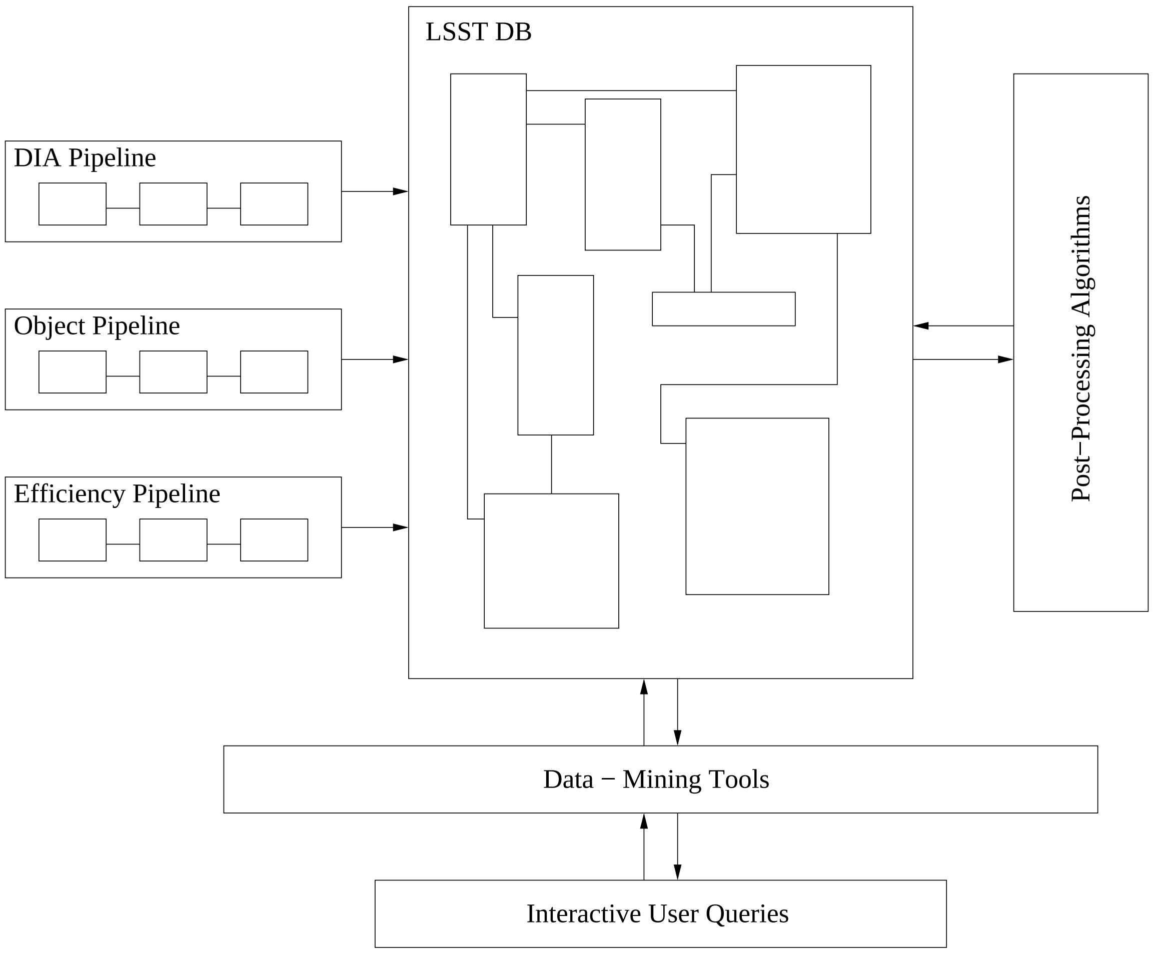

Fig. 1.— High-level schematics of the LSST data flow.

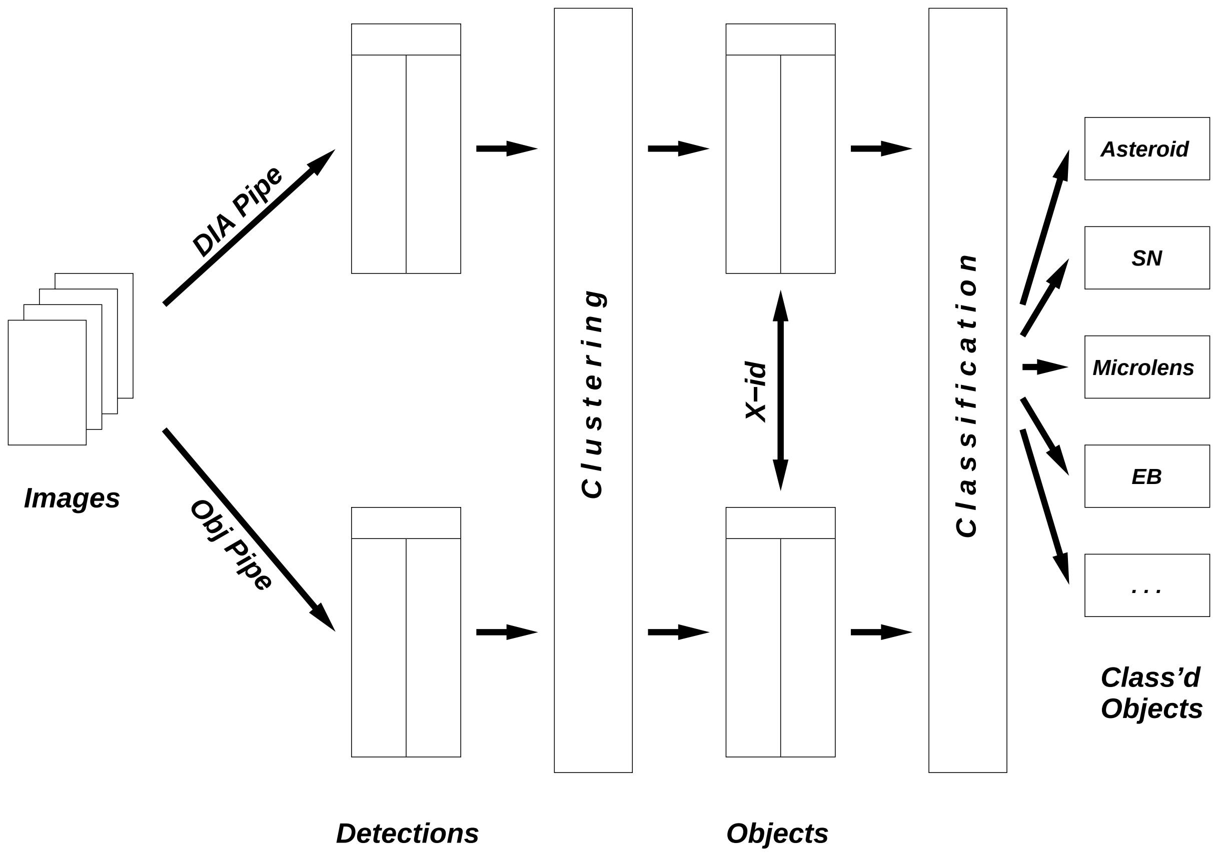

Fig. 2.— Data hierarchy in the LSST DB. Shown are principal data entities: Images, Detections, Objects and Classified Objects, as well as the intermediate algorithmic steps: pipelines, clustering and classification. Note the two separate paths (DIA pipeline and Object-based pipeline) which result in separate Detections and Objects tables. The Objects tables need to be cross-linked, as they will contain common objects.

separate branches of the image reduction pipelines, DIA and PSF. The last component of the triad, the object tables **VarObj** and **Obj**, result from a clustering algorithm running on a corresponding detection table. Clustering will introduce tags (object IDs) for detections to indicate their belonging to a unique object. In this arrangement, all detections related to a specific object can be extracted by running a query on the object ID attribute (**VarObjID** or **ObjID**) in the respective detection table. Finally, the Figure **??** shows an example table for a specific Classified Object (**Asteroid**), which contains orbit elements derived by analyzing moving objects in the sky frames. Note that **Asteroid** is just one possible Classified Object type; there will be many more tables for specific classified objects types, e.g. SN, eclipsing binaries, microlensing events, etc.

We emphasize that this database structure is very preliminary. There can (and will) be many other tables added to the schema in Figure **??**, with numerous additional attributes. We merely intended to provide a framework for addition of new tables and attributes. For example, one can create a separate table (linked to **Image**) for storing efficiency statistics, resulting from efficiency pipeline (see above). We also note that the tables in the LSST DB structure contain the attributes both derived directly from the image processing pipelines (*primary* attributes) and written by external (post-processing) algorithms (*derived* attributes). The following section discusses some specific attributes, by table.

3. LSST DB Attributes

Disclaimer: the contents of this section is subject to change. In fact, it may change significantly.

3.1. Image

The table contains attributes pertaining to the general parameters of the image and observation settings/conditions. The attributes in this table are shared by all detections, on a per image basis.

1. **Image::name**. This is the primary key for the table, and will likely be a combination of parameters (date, time, observing sequence, run, filter, etc.) that uniquely identifies the image in the database.

2. **Image::Date** and **Image::Time** are standard tags to keep the temporal information (as contained in FITS header). In a time-domain survey, such as LSST, this is a crucial

component of the database, linked to almost all potential use cases for LSST DB.

3. **Image::FitsFile** contains pointer to the raw (or calibrated) FITS image of a sky frame. There can be also pointers to the template image and the difference image (for DIA pipeline).

4. **Image::Airmass**, X. This could be one or two values, related to a mean time through the integration or the observation start/stop (important for long exposures). Regardless of the implementation, a functionality to recover airmass X as a function of chip coordinates (x, y) is essential for possible calibration issues (ADC?).

5. **Image::PointingRA** and **Image::PointingDecl** give the telescope pointing, (α_p, δ_p)

6. **Image::WCSinfo**. This structure stores information related to sky-to-CCD mapping.

7. **Image::FWHMStats** structure is related to seeing (average PSF size, FWHMa, FWHMb, and orientation Θ). Besides these mean quantities characterizing the PSF shape for the whole image, a functionality is needed to track changes in PSF as a function of chip position (x, y). This may be implemented through polynomial correction terms of the form $x^a y^b$, with the overall constraint $a + b < c$. Also, these statistics can apply both to the individual CCD and to the entire focal plane.

8. **Image::SkyShape** is a structure to store the polynomial coefficients of the sky representation. The basic idea is to provide a mapping of the sky value to the chip position (and CCD ID).

9. **Image::Gain** and **Image::RDNoise** are data structures rather than simple data types (e.g. floats), since they will store information varying depending on a CCD or amplifier number.

10. **Image::TelescopeFlag** is a data structure to contain observing conditions. These may include wavefront sensor data, wind, transparency/clouds, power issues, temperature, etc.

3.2. DIADet

1. **DIADet::RA** and **DIADet::Decl**. A combination of these two fields serves as the primary key for the table. This is done because most of the queries on this table will likely be cone searches, and have to be optimized. The same is also true for the **ObjDet** table below.

2. **DIADet::ImageName** provides a link to the **Image** table, and to the information about observing conditions and telescope settings.

3. **DIADet::VarObjID** is a foreign key to the **VarObj** table. A clustering algorithm (see §4) will assign a unique VarObjID to each cluster, and this foreign key will be used to identify detections belonging to a given object.

4. **DIADet::ObjID** is a foreign key to the **Obj** table, to identify common objects between DIA and PSF branches of the image reduction pipeline. Note that every detection in the **DIADet** should ideally have a counterpart in the **ObjDet** table, but not vice versa. In practice, this may not be true due to crowding effects.

5. **DIADet::AsteroidID** is a foreign key to the **Asteroid** table.

6. **DIADet::Shape**. This is a complex data structure containing various parameters of the detection shape (e.g., second moments C_{xx}, C_{xy}, C_{yy}; parameters characterizing galactic profiles, such as radial power-law index, etc).

7. **DIADet::HTMZoneID** is provided by the HTM zoning algorithm (see §4).

The rest of the attributes are related to the flux measurements and various flags to indicate a special condition for the detection. One attribute of particular importance is **DIADet::FlagBlend**. This should be a "backward-updateable" field, since many detections originally classified as single detections in moderate seeing, may resolve into blends when a good seeing image becomes available. Therefore, a special "smart" algorithms for blend separation/analysis are needed, see §4.

3.3. ObjDet

The attributes for this table are very similar to the ones for **DIADet** table, discussed above. As above, the primary key is a combination of the sky position attributes, to facilitate cone searches. Here, we concentrate on the fields that differ from the **DIADet** table only (which, in fact, warrant keeping separate tables for DIA and PSF detections).

- **ObjDet::Mag** and **ObjDet::MagErr** fields are different from the corresponding information in the **DIADet** table (there, the difference fluxes are stored). It is important to realize that PSF pipeline will provide the actual magnitude of the object, rather than the difference flux.

- **ObjDet::Sky** and **ObjDet::SkyErr** are fields that do not have analogs in the **DIADet** table, since the DIA pipeline matches the sky intensity between the image and the template.

3.4. VarObj

This table will be populated by a clustering algorithm and by light curve analyzer software.

1. **VarObj::VarObjID** is the primary key for this table. It is a unique identifier which links a given object to the detection table.
2. **VarObj::RA0** and **VarObj::Dec0** are centroid coordinates. These are weighted averages of the detection coordinates for all detections that make up the object.
3. **VarObj::VarType** will be assigned by a light curve analyzer, to distinguish between variability types. This field should be "backward-updateable", since there may be cases when the original variability classification is incorrect for some reason (uncertain photometry, too few points on the lightcurve, etc.), so there should be a way of updating the classification on the objects. In addition, there should be a probability flag which indicates how "certain" is the current variability classification.
4. **VarObj::AlertID** field stores a unique identifier to indicate that the object has been given an alert status. In future, this field may be a foreign key to the separate **Alert** table.

The rest of the attributes in this table are descriptors of the light curve shape (e.g. period, amplitude, timescale, etc).

3.5. Obj

This table is quite different from the **VarObj** table. The term **Obj** is really a misnomer, since the table will also hold variable stars in it. The name resulted from the fact that the table is created from detections coming from the PSF pipeline, which may be considered "static".

1. **Obj::StatObjID** is the primary key for the table, linking it to the individual detections for a given object.

2. **Obj::Parallax**. Will become available for objects in the Solar neighborhood after sufficiently long survey baseline. This field is only available from object-based photometry on normal (non-subtracted) survey images, unless a star is variable.

3. **Obj::PropMotion**. This data structure will hold the proper motion estimates, $\mu_\alpha \cos\delta$ and μ_δ obtained from analyzing the time drift of the object centroid.

Most of the other attributes in this table are related to brightness measurements of the objects, e.g. average magnitudes in *g,r,i,z,y* filters and their respective RMS errors.

3.6. Asteroid

This table holds information for a specific object type, an asteroid. Asteroids will be identified among "movers" detections through a complicated linkage algorithm (see §4).

1. **Asteroid::AsteroidID** is the primary key on the table, to uniquely identify the asteroid.

2. **Asteroid::OrbitElements** is a data structure that contains the asteroid orbital elements $(a, i, e, \Omega, \omega, t_0)$.

3.7. Extended Object

TBD

3.8. Orphan Object

TBD

3.9. Moving Object

TBD

3.10. Variable Star

TBD

3.11. Non-variable Star

TBD

4. Post-Processing Algorithms

There will be a number of user applications to use the database once some of the data have been pushed through the pipeline and ingested into the database. Some of them are presented below:

- **Asteroid linkage**. This algorithm will rely on cone searches in the database, with time and (weaker) brightness constraints to find the orbital solution for moving detections. As a result, the algorithm will provide a unique identifier for each solved asteroid, along with the orbital elements, to be stored in the **Asteroid** table.
- **HTM zoning**. In a quest of speeding up search algorithms for the DB, LSST may use Hierarchical Triangular Mesh (HTM) formalism employed by SDSS. The algorithm provides mapping between sky coordinates (α, δ) and a tiled hierarchical set of HTM zones with unique identifiers. The numbering system for HTM makes it easy to search for close objects. There will be no DB query per se for this post-processing step, rather it is a simple reading of all the DB entries, and assigning HTM Zone to each row.
- **Clustering algorithm**. This is an essential part of all time-domain astronomy. The goal is to associate distinct detections with unique objects, taking great care of removing outlier/flagged data points. This algorithm will involve a cone search, with potential magnitude (flux) constraints and culling due to artifacts.
- **Lightcurve analyzer/Classifier**. This module will analyze a time-ordered sequence of flux measurements of each object in the transient database and attempt to label it as a certain astrophysical object (e.g. Cepheid, RR Lyr, microlensing event, Supernova, etc.) The decision will be made with some kind of template fitting or Fourier frequency analysis and will also produce numerous light curve characteristics (see **VarObj** table). A machine learning algorithm employing neural networks is also possible.

- **Photometric calibration**. While surveys can be calibrated on a nightly basis by observing photometric standard stars, one usually get a lower systematic error by calibrating a set of photometric measurements made over extended time period (e.g. 2MASS global photometric calibration). The algorithm will operate on the object-based DB, and will utilize cone or object ID search, while culling known variable stars. It will produce a set of corrections to photometric zeropoints based on CCD number and (possibly) chip coordinates, and atmospheric extinction terms.

- **Astrometric calibration**. Will be related to the clustering algorithm, using the centroid of the sky positions to improve the astrometric accuracy of the detections.

- **Shear calculation for weak lensing**. This is one of the science goals of the LSST, to quantify the distribution of the dark mass in the universe. Will involve a search for a specific object type (galaxy) and processing the shape information.

- **Proper motion calculator**. An algorithm to compute proper motion of an object based on the time drift of the detection centroid.

- **Alert system for microlensing and other interesting phenomena**. This module will involve looking for "new" objects in the sky (cone search + time constraints + photometry (SNR) constraints). Output is not written out to the database, but rather is distributed among the community (alerts).

5. Example Queries

This section lists some typical queries for the database. The queries are organized into broad classes, which describe the primary purpose of the query: *scientific*, i.e. queries that will be issued by researchers interested in some astronomical phenomenon or object(s); and *auxiliary*, i.e. queries that will be utilized by algorithms (usually hidden from ordinary users) that interact with the database with the purpose of derive new attributes or updating the existing ones. For all queries, we give an approximate SQL expression which indicates the attributes relevant for the query.

In all queries below, we use a notation that all items typed in `typewriter font` are treated as parameters (i.e. substituted with certain values, not interpreted verbatim).

5.1. Scientific Queries

1. Extract the time series for a given object (specified through `ObjectID`), in a given `Filter`:

   ```
   SELECT   Image.MJD, ObjDet.Mag, ObjDet.MagErr
   FROM     ObjDet o, Image i
   WHERE    o.ObjID = ObjectID AND
            o.ImageName = i.name AND
            o.Filter = Filter;
   ```

   Note that in the current schema, the query necessary implies a join between the Detection table and the Image table, since the time information is kept only in the Image table. This is done to conserve space, since time is the same for all detections in a given image. Filter information is also shared by all detections in an image, so perhaps it too should be stored in the Image table? (TBD)

2. Select a list of objects in a given sky area, without extracting their time histories. This type of query is called a "cone" search, since it retrieves objects in a pseudo-cone solid angle. There are many varieties of the cone search, depending on how the shape is defined (e.g., rectangle or ellipse in RA, DEC; HTM zone number, etc). Here's an example for a rectangle shape, defined by `Ra1, Ra2, Dec1, Dec2`:

   ```
   SELECT   *
   FROM     Obj
   WHERE    RA0 BETWEEN Ra1 AND Ra2 AND
            Dec0 BETWEEN Dec1 AND Dec2;
   ```

Note: Due to cyclic nature of the RA coordinate (0^h is the same as 24^h), the query above may fail when the search region includes the zeroth meridian. In general, this type of query usually relies on R-tree or B-tree algorithms, for fast 2-d indexing. Also, the query can be modified by using **Obj.Flag** attribute.

Same type of search can be used for extracting data for input into cluster-finding algorithm (note that this is different from DB "clustering", since we are already querying the Object table; this "clustering" means finding stellar clusters, or galaxy clusters). In this case, there might be an additional constraint on the Obj::Type attribute, depending on the nature of the clusters.

3. Select all variable objects of a specific type (`Type`) in a given area:

   ```
   SELECT   *
   FROM     VarObj
   WHERE    RA0 BETWEEN Ra1 AND Ra2 AND
            Dec0 BETWEEN Dec1 AND Dec2 AND
            VarType = Type;
   ```

   The case for such a query is when one wants to find just this type of objects, which could be, for example SN, elliptical galaxies, eclipsing binaries, etc. This assumes two things: 1) the light curve classifier has already assigned proper classification to each variable object, and 2) the VarType attribute has a potential to distinguish between these different classes (in other words, it is more than just 'stars', 'galaxies' and 'moving objects').

4. Select all objects with variability timescale less than specified threshold (`TimeScale`), or with the period in a certain range `p1, p2`, or amplitude greater than some value (`Amp`):

   ```
   SELECT   *
   FROM     VarObj
   WHERE    TimeScale < TimeScale OR
            Period BETWEEN p1 AND p2 OR
            Amplitude > Amp;
   ```

   A search like this can be useful in locating objects (regardless of their classification) that possess certain variability characteristics indicative of specific phenomena being studied.

5. Select all variable objects in a given sky area:

```
SELECT   *
FROM     VarObj
WHERE    RA0 BETWEEN Ra1 AND Ra2 AND
         Dec0 BETWEEN Dec1 AND Dec2;
```

This is a straightforward query of object database (clustered detections), resulting from the DIA pipeline.

6. Select all galaxies (Obj.Type = `Type`) in a given area:

```
SELECT   *
FROM     Obj
WHERE    RA0 BETWEEN Ra1 AND Ra2 AND
         Dec0 BETWEEN Dec1 AND Dec2 AND
         Type = Type;
```

The query will return extragalactic objects which will be used for calculating the shear for weak lensing applications.

7. A generic search of type "cone-magnitude-color" will useful for finding particular astronomical objects (e.g., quasars, white dwarfs, binaries, etc.) in a given sky area. The type of objects to mine is determined by its position in the color-magnitude space. Expanding on the simple cone search above, we add additional constraints for magnitude (`z1, z2`) and color (`gr1, gr2, iz1, iz2`):

```
SELECT   *
FROM     Obj
WHERE    RA0 BETWEEN Ra1 AND Ra2 AND
         Dec0 BETWEEN Dec1 AND Dec2 AND
         zMag BETWEEN z1 AND z2 AND
         gMag-rMag BETWEEN gr1 AND gr2 AND
         iMag-zMag BETWEEN iz1 AND iz2;
```

The query results can be filtered by using various flags, stored in the Detections table. Additional constraints may include a high value of the proper motion, or measured parallax.

8. Select objects with statistically significant (at the level `Lev`) proper motion in the sky, for example to study streamers in the Galactic halo:

```
SELECT   *
FROM     Obj
WHERE    abs(muRA) > Lev*muRAErr OR
         abs(muDec) > Lev*muDecErr;
```

9. Select objects with a measured parallax in a given range (`p1, p2`):

```
SELECT   *
FROM     Obj
WHERE    parallax BETWEEN p1 AND p2;
```

   This query will select nearby stars at a given range of distances, and can be useful for Galactic structure work.

10. Select all "new" objects in the database. This type of query is the basis of the alert system, whereby only new detections, which were not observed before are accessed. The idea for the SQL query below is to extract rows from Detection table which do not have VarObjID attribute set (that is, these detections do not belong to known clusters, or objects):

```
SELECT   *
FROM     DIADet
WHERE    VarObjID = NULL;
```

   This of course implicitly assumes that clustering algorithm is run first, so that known objects will have the VarObjID attribute propagated into the DIADet table.

11. Select transient near a known galaxy (for supernova search).

```
SELECT   VarObj.RA0, VarObj.Dec0
FROM     VarObj v, Obj o
WHERE    abs(v.RA0-o.RA0) < DeltaRA AND
         abs(v.Dec0-o.Dec0) < DeltaDec AND
         o.Type = 'galaxy';
```

5.2. Auxiliary Queries

1. Select all detections in a given filter in a given sky area, for the purpose of clustering them into objects.

```
SELECT   *
FROM     ObjDet
WHERE    RA BETWEEN Ra1 AND Ra2 AND
         Dec BETWEEN Dec1 AND Dec2;
```

   This query may rely on additional filtering of results by their flag attributes.

2. Select time series data for all objects in a given area of the sky (`Ra1, Ra2, Dec1, Dec2`), in photometric band `Filter`, with the additional constraint of low variability (`VarThresh`):

```
SELECT  Obj.ObjID, Image.MJD, Image.Airmass, Image.ExpTime,
        ObjDet.Mag, ObjDet.MagErr, ObjDet.AmpID, ObjDet.CCDID,
        ObjDet.XPos, ObjDet.YPos
FROM    Obj o, ObjDet d, Image i
WHERE   o.RA0 BETWEEN Ra1 AND Ra2 AND
        o.Dec0 BETWEEN Dec1 AND Dec2 AND
        d.Filter = Filter AND
        o.MagErr < VarThresh AND
        d.ObjID = o.ObjID AND
        d.ImageName = i.name;
```

This type of query is especially useful for selecting photometric “standards”, i.e. stars on very nearly constant brightness. The extracted light curves of standards are used in photometric calibration work. Note that there is no Obj::MagErr attribute as such, but there are five MagErr attributes for different photometric bands. Hence, for this query to be implemented, the correct MagErr attribute for an appropriate band must be *computed* based on the `filter` value! For example, if `Filter` is ’g’, then o.gMagErr should be substituted. Not sure how and whether this can be implemented; perhaps through use of DB functions?

3. Select the time series of the object centroids and their uncertaintines, for calculating parallax and proper motion of the object. The query uses `ObjectID` to identify the object and `Filter` to choose the filter with the best seeing:

```
SELECT  Image.MJD, ObjDet.RA, ObjDet.Dec, ObjDet.RAErr, ObjDet.DecErr
FROM    ObjDet o, Image i
WHERE   o.ObjID = ObjectID AND
        o.ImageName = i.name AND
        o.Filter = Filter;
```

4. (S) N-point correlation function, for large-scale structure studies

5. (S) Color gradients/profiles on extended sources, i.e. blue core galaxies.

6. (S) Select all objects which vary, and were not in the template image. These will be “movers” (asteroids and KBOs) and outburst objects.

7. (S) Select transients with no orbital solution. This is a subset of the previous query, including mostly outburst objects (there could still be asteroids returned by this query, if this is first few observations of the field and no orbital solution has yet been found).

8. (S) Find NEAs. Query could involve both searching for "streaked" detections (elongated PSF); and the orbital elements attributes for specific signature corresponding to NEAs.

6. Performance and Scaling Analysis

TBD

This work was performed under the auspices of the U.S. Department of Energy, National Nuclear Security Administration by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.