VISTA DATA FLOW SYSTEM (VDFS)

---

for VISTA & WFCAM data

WSA Database Design Document

author
N.C.Hambly (WFAU Edinburgh)
Science Archive Project Scientist
number
VDF-WFA-WSA-007
issue
Issue 1.0
date
2 Apr 2003
co-authors
R.G.Mann, E.T.W.Sutorius, M.A.Read & A.C.Davenhall

Contents

1 SCOPE

The WFCAM Science Archive (WSA) database design document describes how pixel, catalogue and other data are stored within the WSA. The document is intended to be a technical reference for archive implementation, curation and also for development of the user interface layer. The design is expressed in sufficient detail to enable implementation of the V1.0 WSA system, to enable development of the required curation software, and to inform development of the user interface software. V2.0 plans will be described at a preliminary level, ie. R&D plans will be outlined, where appropriate, in each Section (the V2.0 archive will be reviewed at the end of Q1 2004 – see AD07)

The external requirements on the WSA design are expressed in the following applicable documents (which are listed in Section 12): science requirements and their consequent functional requirements are detailed in the WSA Science Requirements Analysis Document (SRAD; AD01). The SRAD itself follows on directly from the outline survey designs that form the basis of the UKIRT Infrared Deep Sky Survey proposal (AD02) which form the core usage of the WSA. The characteristics of the ingested data are detailed in the CASU/WFAU Interface Control Document (ICD; AD03). Top–level dataflows are described in the WSA Data Flow Document (DFD; AD04). Relevant details of the archive hardware design can be found in the Hardware/OS/DBMS Design Document (HDD; AD05). Finally, and most importantly from the user's point of view, likely modes of science exploitation of the WSA are illustrated in 'Usages of the WSA' (AD06). These usage examples set out the kinds of scientific questions that are expected to be asked of the archive, and directly impact the design and organisation of databases therein (eg. Section 5.1).

The relational design of the archive follows on directly from the functional requirements expressed in the SRAD, and curation 'use cases' (see Section 4). Relational models are presented that show the data entity (table) organisation along with their associated attributes (fields). These entity–relationship models are mapped onto tables, and the database schemas are described. Processing requirements are developed from the curation use cases and details of the additional processing for curation are given.

2 OVERVIEW

This document is structured as follows. In Section 3, we discuss the fundamental concepts behind the database design, including pragmatic choice of DBMS, overall archive organisation, and calibration considerations. In Section 4 we describe a set of curation 'use cases' which have been developed along with the SRAD functional requirements to enable database design. In Section 5 we introduce the relational design of the databases that will comprise the WSA, including entity–relationship models (ERMs) that provide both general, illustrative overviews of the database structure and specific detail for V1.0 data. Section 6 goes on to develop the relational design of the V1.0 archive into implementation of prototype database tables within the chosen DBMS; details of tables, column names, unique identifiers and primary and foreign keys are given. We also desribe logical table 'views' that will be created. In Section 7 we discuss issues of indexing within the database, give details of logical entities/attributes for enhanced utility, and describe defined functions that will be available for both curation and use. Finally, in Section 9 we give details of the application software that is being devloped for the purposes of data transfer, ingest and curation.

3 FUNDAMENTALS

3.1 Relational versus object–oriented design

WSA design will be based on a relational DBMS. Throughout the planning phases of the WSA, we have followed closely the development of the science archive system for the Sloan Digital Sky Survey

(SDSS) at Johns Hopkins University [2]. The pragmatic approach for WSA design has always been to use, wherever possible, the same or similar archive systems as SDSS since the latter is state–of–the–art in astronomical archives and is based on many staff years of effort that realistically are not available to the WSA. Originally, the SDSS science archive was based on the OODBMS Objectivity. However, since the Early Data Release (EDR) performance issues have arisen (eg. concerning indexing [6]) and the imminent SDSS release (DR1) will be made under an RDBMS system known as SkyServer (it should perhaps be noted that a relational design results in an archive system that is more intuitively understandable to the non–expert user).

3.2 RDBMS choice

The V1.0 WSA will be implemented using the Microsoft DBMS 'SQL Server' running under the Windows 2000 'Advanced Server' operating system. The SDSS DR1 release in 'SkyServer' uses SQL Server, and once again we are benefitting from the SDSS experience and developments in following this choice. Note however that we are not tied to SQL Server for future developments. If this choice proves not to be scalable to data volumes expected to accumulate after several years of operation, then we will use one of the other two large commercial RDBMS products currently available: IBM's DB2 or Oracle (additional information, particularly concerning licencing, is available in AD05). It should be noted that the interface to all these DBMS products is industry–standard SQL, so V1.0 application scripts and code are not wedded to SQL Server.

3.3 Arrangement of data

Pixel data will be stored as flat files rather than 'binary large objects' (BLOBs) internal within the DBMS. This is so that high data volume image usages that are not time critical will not impact datamining usages on catalogues where more 'real time' performance is required for data exploration/interaction. However, pixel files and the pixel metadata will be tracked in tables within the DBMS so that the image metadata can be browsed/queried in the same way as any other data (catalogues and ancillary data will of course be stored in tables within the DBMS).

3.4 Null values

All attributes in the WSA will be 'not null'. Inevitably, from time to time attribute values will be unavailable or not applicable in certain situations. RDBMSs provide facilities to handle this eventuality, with nullable attributes handled by the system. However, making use of this facility can complicate curation and use enormously (eg. SQL can become cumbersome). On advice from Jim Gray of Microsoft Research, and following the SkyServer example and our own experience so far, we assert that all attributes must have externally specified values and when not available or not applicable, a specific meaningful flag value will be inserted. Where null values are referenced to other tables via foreign keys, a null record will be included in the referenced table (this record consisting, in turn, of externally specified null values for all attributes).

3.5 Curation information

The WSA databases will contain tables giving details of curation information, for the purposes of curation housekeeping and also for informing users of updates and processing that have taken place. Further, the WSA will contain tables of constants and other generic information (eg. a table of tables giving details of the schemas and their provenance), again for housekeeping and informing users.

3.6 Calibration

The WSA will store astrometrically and photometrically calibrated quantities via attributes stored in calibration–independent units along with calibration coefficients, but will also store calibrated quantities based on the most recent calibration available for efficiency and ease of use. It is a specific requirements (see the SRAD) that recalibration of photometric and/or astrometric quantities must be possible within the WSA, and further that previous calibrations be available for use in addition to any current (and presumably 'best', or at least up–to–date) version.

3.6.1 Astrometry

Spherical co-ordinates will be stored internally within the WSA in equinox J2000.0 celestial RA and Dec tied to the Hipparcos reference frame via secondary astrometric standards. Astrometric calibrations will be specified via zenith–polynomial or tangent plane projections – see, for example, the ICD (AD03) and references therein – for non–resampled and mosaiced/resampled images respectively. Hence, astrometric calibration coefficients will consist of zeropoints and the 'CD' linear coefficients (for tangent plane projections) and additionally 'PV' non–linear radial distortion coefficients for zenith projections). Calibration–independent XY co-ordinates will always be associated, via relationships to images, to their calibrating coefficients. Low–level, highly non–linear distortions that may become apparent after several years of accumulation of data (analogous to the 'swirl' distortion maps present on Schmidt plates – eg. [7]) will be allowed for in the calibration entities and calibration scheme. Finally, spatial indexing attributes will be allowed for every occurence of celestial co-ordinates in the archive to enable efficient curation and usage employing positions (eg. source merging, proximity searching).

3.6.2 Photometry

The photometric calibration scheme for the WSA will follow standard practice for IR photometry as implemented for 2MASS [10] with extensions for non–linear behaviour as a function of time over one night and with airmass as follows:

$$m_{\rm INST} = m_{\rm CAL} + a + b(\Delta T) + c(\Delta T)^2 + A(\chi - 1) + B(\chi - 1)^2 - 2.5\log(t_{\rm EXP})$$

where $m_{\rm INST}$ is the instrumental magnitude; a is the time–independent zeropoint for a given nightly device/filter combination; b is the first–order time–dependent zeropoint coefficient; c is the second–order time–dependent zeropoint coefficient; A is the first–order nightly extinction per unit airmass for a given filter; and B is the second–order nightly extinction per unit airmass for a given filter (ΔT is a time offset from midnight and χ is airmass). Note that no colour dependency on extinction will be used in the WSA photometric calibration, following standard IR practice.

Furthermore, for 'list–driven' photometric measurements (see later), the SDSS 'luptitude' scale [8] will be used to allow for well–behaved calibrated magnitudes for very small or negative fluxes. This scale uses the same zeropoints as the above conventional scale and is identical for significant flux detections, but has an additional 'softening' parameter b' that is related to the waveband, typical seeing and sky brightness of a given set of images. Values of b' will be stored in the appropriate place for sets of images in the WSA.

Finally, once again 2D photometric flatness maps (see, for example, [11] for a description of analogous wide–field imaging systematic errors in the optical) will be allowed for in the calibration entities and scheme.

3.7 Reruns, repeats and duplicates

Multiple instances of a given image or source detection will inevitably occur within the WSA. There are several reasons for this:

- pixel data may be reprocessed and/or rerun through altered source extraction procedures, particularly in the early stages of commissioning of the data flow system. Newly processed image data and corresponding source detection lists, re-ingested, will result in repeats of the same data that must be uniquely identified and flagged as such.
- survey design will result in overlap areas between adjacent fields, resulting in duplicate detections of the same objects at different times – useful data that will be retained in the archive but that must be distinguishable for the purposes of duplicate–free source lists.
- stacked survey products will be recreated after some observing period, resulting in ingestion of updated image products and higher s/n detections of previously ingested sources – again, these must be distinguishable when retained in catalogues.

Where necessary, all records will be flagged as current/old where repeats are potentially present, and all potentially duplicated records will have a primary/secondary attribute specified for one primary, all other records being identified as secondary. The intention is to retain old versions of processed image products and their source detections so that users can refer back to observations previously used to make target lists for follow–up programmes. Of course, retention of large amounts of pixel data has to be subject to storage constraints – the 'best' data (ie. most recently processed) must take precedence when disk space is limited.

4 Curation use cases

A set of curation 'use cases' expressed in plain language are reproduced from [12] in Appendix 10.1. The key points to emerge from this exercise with respect to the archive relational design are:

- updates to the database occur on a variety of timescales, from daily ingest to quarterly (or possibly less frequent) 'releases';
- housekeeping (curation) entities emerge as being necessary to keep track of updates as survey data accumulate;
- new attributes and relationships between observational entities emerge that are required for the purposes of curation.

In the next Section, these points are propagated into the database design. The curation use cases are expanded into activity flow diagrams in Section 9 for the purposes of detailing the curation software design.

5 WSA relational models

5.1 Archive databases

There are several fundamental requirements that come directly from the SRAD and WSA usages (AD01 and AD06) and curation use cases:

- ingest occurs on a daily basis, and a complex set of curation tasks must be undertaken on a periodic basis to produce useful survey catalogues – hence, the concept of *offline* and *released* databases emerges;
- data having various proprietory periods will be stored within the archive and proprietorial rights must be enforced;
- survey programmes consisting of different combinations of passbands within given fields, and different observing modes, will be ingested and stored;
- externally produced survey datasets will be held in the archive;
- browsing of the *descriptive* data (ie. details of what is held in the archive) will be unrestricted, even when access to the data themselves is restricted.

For example, it might be considered possible to have a small number of database tables in a single database to simplify curation. However, on examination this scenario will not work – eg. a merged source catalogue becomes excessively complicated since at any given time some data may be open to all users while other data may be still be subject to proprietorial rights. These considerations result in a top–level archive organisation that is illustrated in Figure 1. Each box within the overall WSA illustrates a different physical database that will be stored and curated. The key points to note are:

- the 'WFAU database' is the offline, daily/periodically updated version of all archived databases, these latter databases being copied out to a publicly accessible catalogue server at release time;
- within a given database, the observations that are stored form natural groups with different characteristics; these groups are identified as 'entities' and their characteristics as attributes in the relational design detailed in the next Section.

5.2 A brief introduction to entity–relationship modelling

Entity–relationship models (ERMs) have been employed in the design of the WSA. Simply put, an *entity* is something of significance to the system data that appears many times with a fixed set of *attributes*; relationships (generally one–to–many) naturally occur between entities. The advantages of ER modelling are:

- it provides a simple, pictorial summary of the relational design for the database are made;
- ER modelling is RDBMS–independent, and results in a design that can be easily mapped into any RDBMS;
- ERMs map directly into a table design: entities become tables, attributes become columns and relationships become *foreign key constraints* between the tables.

Furthermore, in many cases the models as expressed through ERM are sufficiently general as to be applicable to VISTA data (eg. see Section 5.4).

Here we will use ER modelling in its simplest form to described the WSA relational design. To illustrate the technique, an example ERM is shown in Figure 2. This is for the SuperCOSMOS Sky Survey [13] (SSS), hereafter known as the SuperCOSMOS Science Archive (SSA); the SSA is being used to prototype many aspects of the V1.0 WSA hardware and interfaces (see Appendix 10.2). In the diagram,

- entities are shown as boxes with rounded corners ('softboxes');

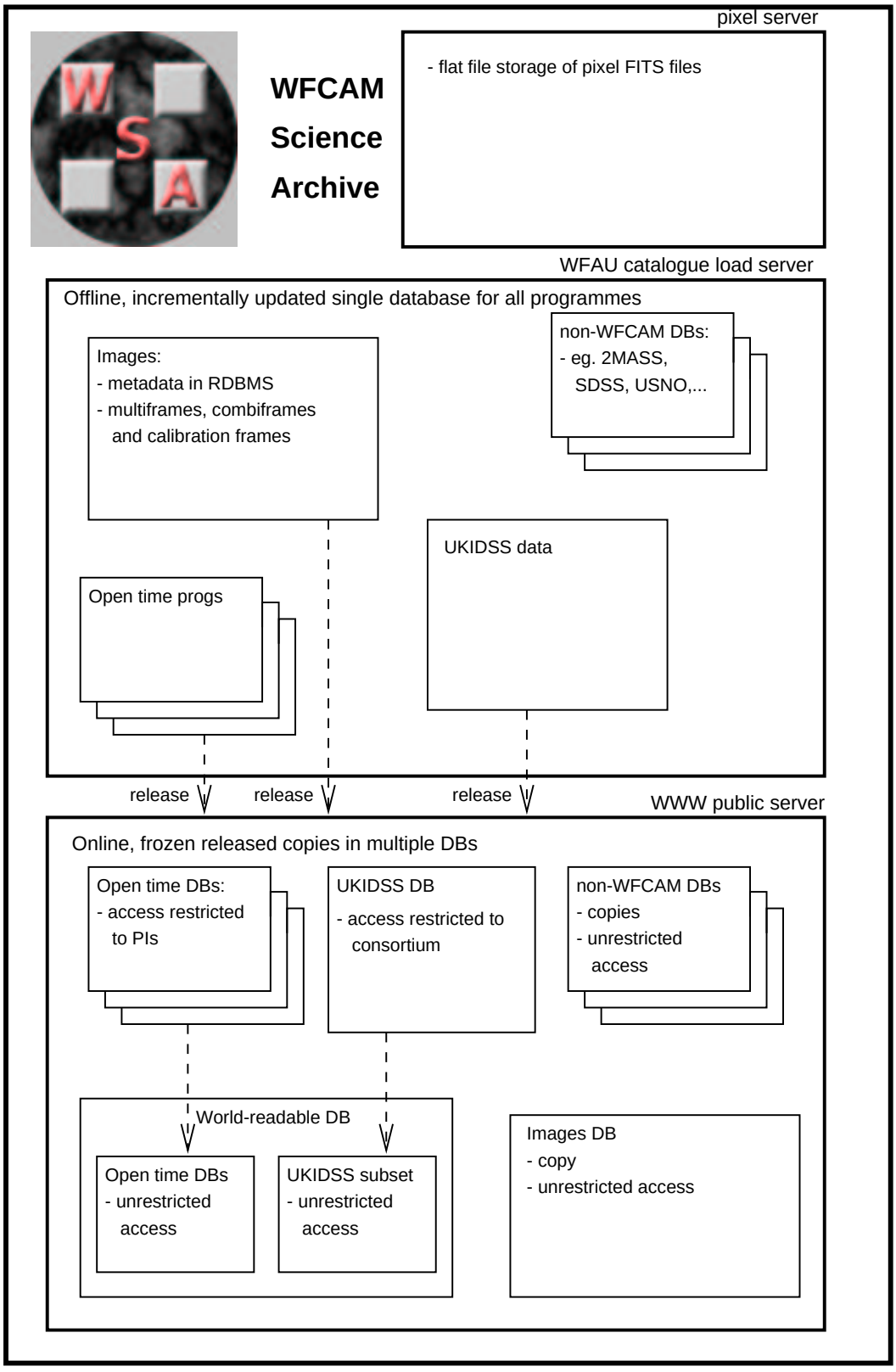

Figure 1: *Schematic overview of the WSA. Within the archive, there are three distinct data servers: pixel flat files, offline DBMS and online DBMS. The online DBMS contains several distinct databases.*

- a few attributes for each entity are listed – a unique attribute is labelled with # while * denotes any other mandatory (but not necessarily unique) attribute;
- relationships are illustrated with a single line forking out for a one–to–many relationship, dashed lines illustrate optional relationships;
- a barred relationship is used as a shorthand way of noting propagation of a unique attribute from one entity to another.

So, in Figure 2 for the SSA, we have entities *survey info*, *plate info*, *field info*, *survey detection* and *merged detection* where

- every merged detection consists of one or more survey detections;
- every survey detection belongs to one of the survey subtypes B, R1, R2 or I;
- every plate yields one or more survey detections and belongs to one survey and one field;
- every field is covered by one or more plates (actually 4 in the SSA);
- every survey consists of many plates in many fields.
- barred relationships indicate combinatorial unique identifiers (UIDs) in entities where the specified attribute is not unique – in the case of the SSA field numbers in entity *field info*, the combination of the survey UID plus the field ID results in a UID for a field (eg. field 319 exists in the ESO/SRC field system in the southern hemisphere and also in the POSS–I and POSS–II systems in the northern hemisphere, all three at different co-ordinates).

Appendix 10.2 illustrates, in summary, the resulting schema file for the SSA for implementation in SQL Server both as a standalone prototype and as one of the external surveys required to be stored for cross–identifications within the WSA.

In the following subsections, we provide and discuss the ERMs for the WSA. For clarity, these are divided into top–level data, pixel data, calibration data, catalogue data, external data and curation/housekeeping data. The final subsection discusses how these ERMs fit into the databases in the overall archive as described previously.

5.3 Top–level entities

A small set of top–level entities is illustrated in Figure 3. These track data relevant to the archive as a whole, and copies will be held with every database set (see Section 5.9 for a discussion of the arrangement of the following entities within the various databases held in the archive).

5.4 WFCAM pixel data

Image files stored within the WSA fall into two broad categories:

- *multiframes* reflecting the characteristic 2×2 device 'paw print' of WFCAM on the sky;
- *combiframes* – the result of a stack or mosaic operation (in the pipeline or in the archive itself) on many multiframes that are also individually stored in the archive;

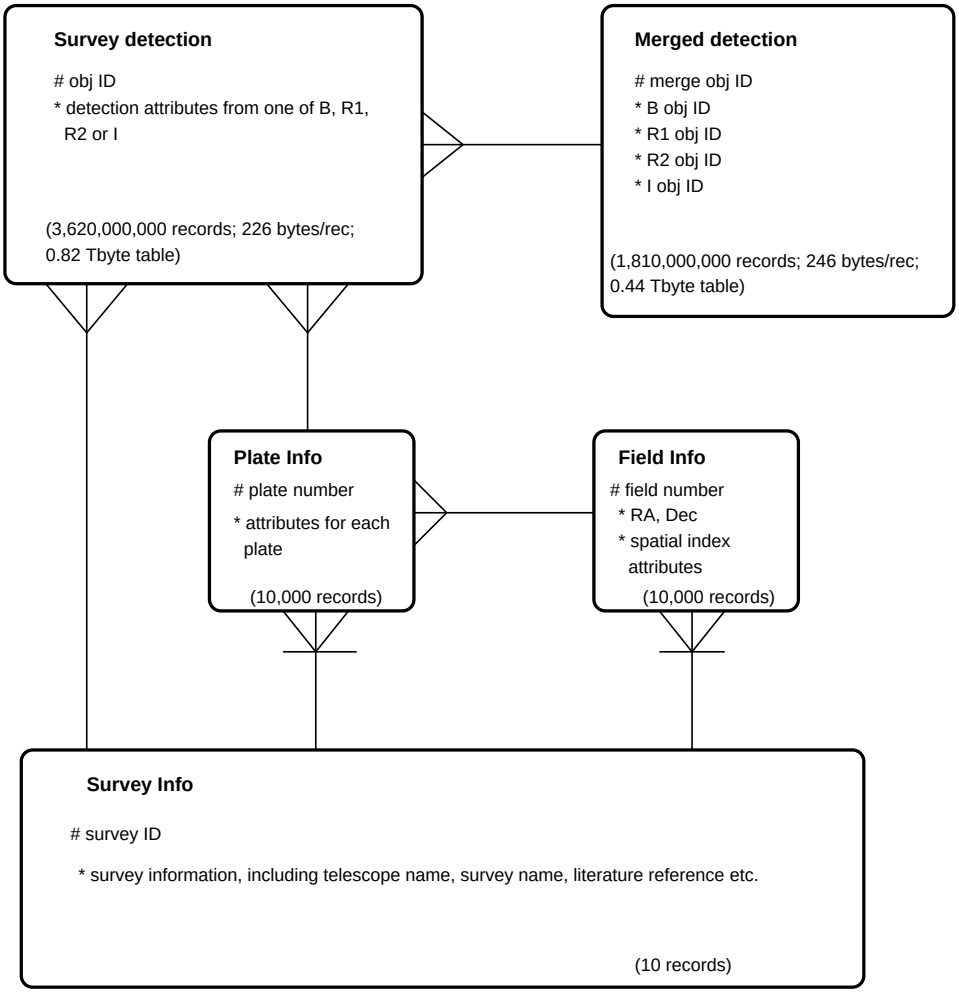

Figure 2: *Entity–relationship model for the SuperCOSMOS Science Archive, an implementation of the SSS [13] in a commercial RDBMS (relative sizes of the different entities are indicated for information only – this is not standard ERM practice).*

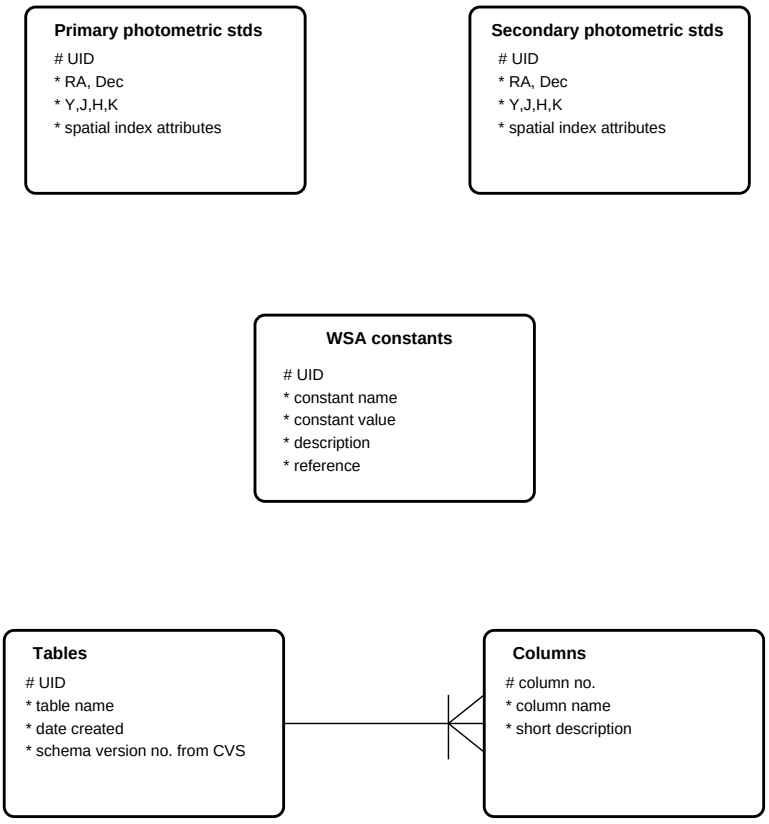

Figure 3: *Constant, photometric standard and other data entities relevant to the entire archive. Photometric standard star lists will be related to WFCAM observations via join tables (see later).*

note that the major distinguishing feature between these two is that the latter have combinatorial *provenance*, ie. a list of images used to create that image. Multiframes have individual astrometric calibrations for each extension image. A combiframe may consist of a single contiguous image (resulting from mosaicing and probably also stacking) or may itself be a non–contiguous multiframe (for example, the combination of many dithered exposures for the UKIDSS UDS), still reflecting the WFCAM paw–print and having individual astrometric calibrations for its four extension images. Combiframes do not have single dark, flat, defringing or sky frames but do have a single confidence frame resulting from the combination process. Because combiframes and multiframes have different attributes, two different ERMs are required to handle these within the WSA.

The ERM for multiframes is shown in Figure 4. Key points to note are:

- *Programme* and *Survey field* entities are included to keep track of which multiframe belongs to which programme/survey–field combination;
- a *Difference image* entity keeps track of the pairs of images that comprise default stored difference images (ie. H_2–K images for the UKIDSS GPS);
- every *Multiframe* is associated with one filter, and consists of one or more *Detector frames* (actually 4);
- every *Multiframe* has associated library calibration multiframes *Dark*, *Confidence*, *Flatfield*, *Defringe* and *Sky*, which in turn had detector frames (calibration multiframes are stored in separate entities to the science multiframes since they will have different attributes; besides, it is clearer from the user point of view to do so);

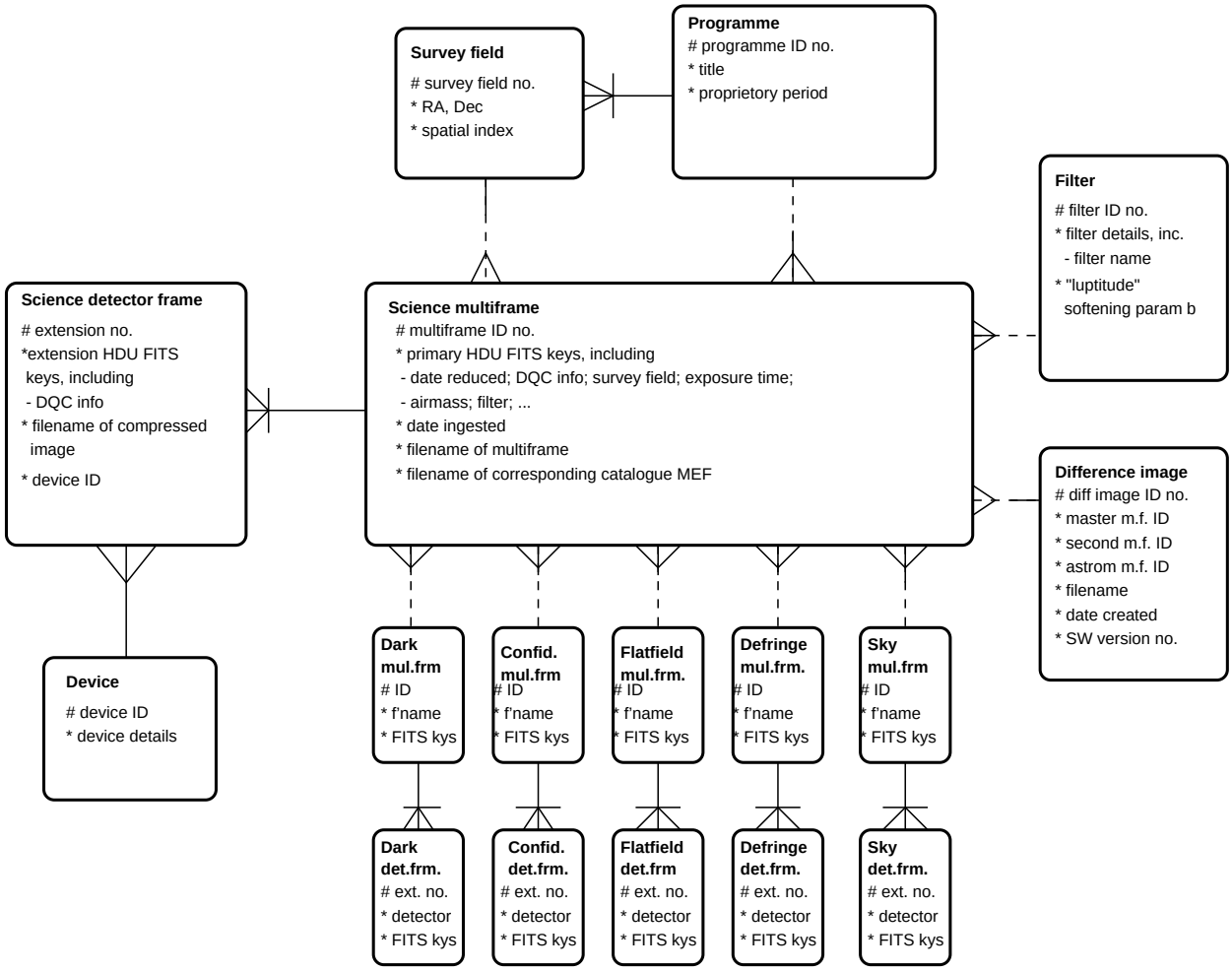

Figure 4: *Entity–relationship model for WFCAM multiframes.*

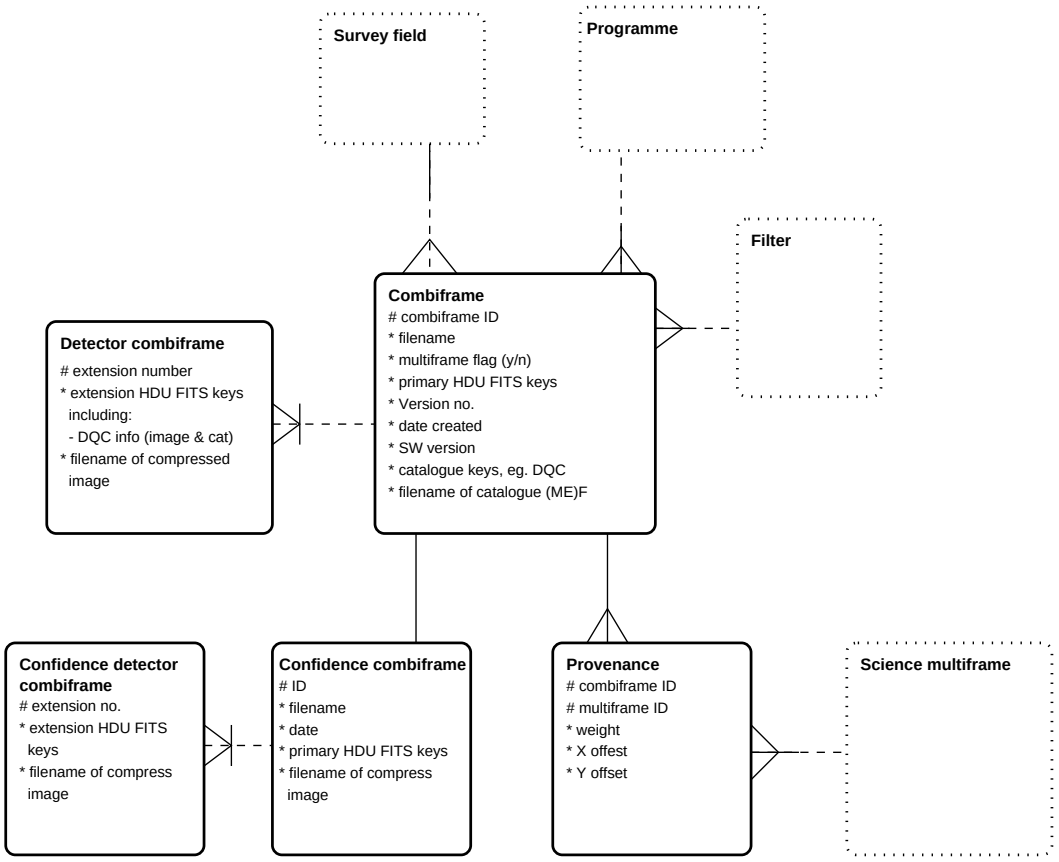

Figure 5: *Entity–relationship model for WFCAM combiframes.*

- primary HDU FITS keys appear in the *Multiframe* entity while extension HDU FITS keys appear in *Detector*; each detector frame is associated with a particular device whose details will be listed in the device entity (in case devices are swapped out over the lifetime of WFCAM).

The first point is pivotal. From the point of view of curation of the large survey programmes, the WSA needs a standardised field identification system to associate different passband observations in the same field (which may have small arbitrary positional offsets rendering co-ordinates alone insufficiently precise as an associative criterion). This is based on our experience of curating the Schmidt photographic surveys and associated data archive.

The ERM for combiframes is shown in Figure 5, where entities shown as dotted softboxes are in the previous ERM. The principal differences between Figures 4 and 5 are:

- every *Combiframe* has *Provenance* – a list of images that links back to individual multiframes;
- every *Combiframe* may have one or more (in practice, 4) *Detector combiframes* or may be an image in itself (in which case it has no *Detector combiframes*);
- every *Combiframe* has an associated (single or multiframe set) confidence image.

Examples of combiframes will be the UKIDSS UDS and DXS surveys, and open time surveys that observe deeply in restricted areas. As in the multiframe ERM, every combiframe belongs to one programme and filter, and may belong to a particular standard survey field.

In both pixel ERMs, filename attributes associate all the metadata with i) the image FITS file ii) for science frames, the catalogue file ingested from CASU with the image or generated using the

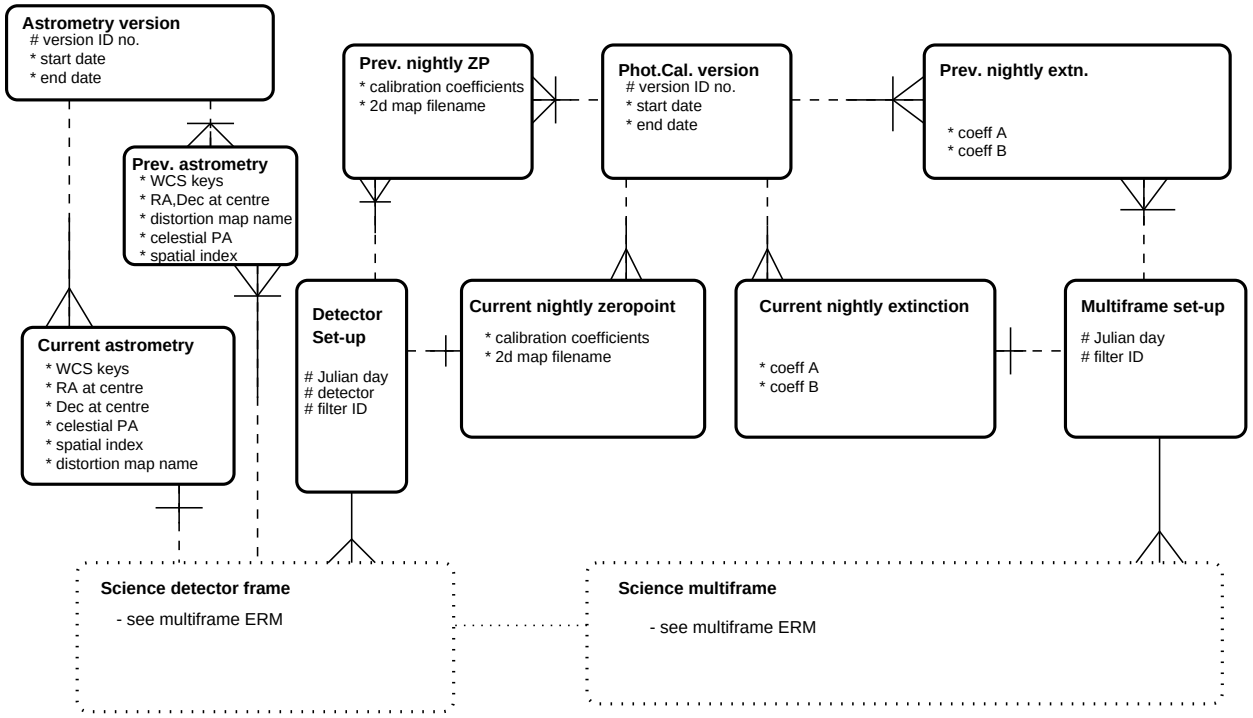

Figure 6: *Entity–relationship model for calibration of WFCAM multiframes.*

same standard CASU source detection software; iii) the filename of a compressed image for speedy quick–look finder charts and or browsing. All these files will be stored externally to the DBMS in the mass–storage flat file Linux file system (AD05).

These ERMs are applicable to VISTA data. For example, there is nothing in Figure 4 that specifies 4 devices per multiframe – 16 (cf. VISTA) will also fit the model.

5.5 WFCAM photometric/astrometric calibration data

Both astrometric and photometric calibrations data will be associated with images stored in the WSA, and then that same calibration information will be associated with catalogued sources through the later's association with their respective images. Calibration data present an additional complication in that versioning over time is required, and all previous calibration versions need to be retained (a requirement from the SRAD). Version entities are implied for this purpose, these entities have the following attributes: unique version number, start time and end time.

Figure 6 illustrates multiframe calibration entities; again dotted softboxes are entities introduced previously. Photometric calibration coefficients are associated via 'set–up' entities that list every combination of photometric night and filter, and in the case of detector–based calibration, every combination of detector/night/filter. Note that the zeropoint and extinction coefficients are separate since the latter are the same for all detectors in a multiframe. On recalibration, the 'current' coefficients will be moved to the 'previous' store and new values calculated and inserted in their place, with a new version number and start time created for the new current calibration. One final point to note about both calibrations is that they both allow for low–level systematic error mapping in 2D over each detector area via an attribute that points to a correction map file.

Figure 7 illustrates analogous calibration entities for image combiframes. Photometric calibration of a combiframe may be fundamentally different to that of individual multiframes; the coefficient and version entities are distinct from previous illustrated entities because

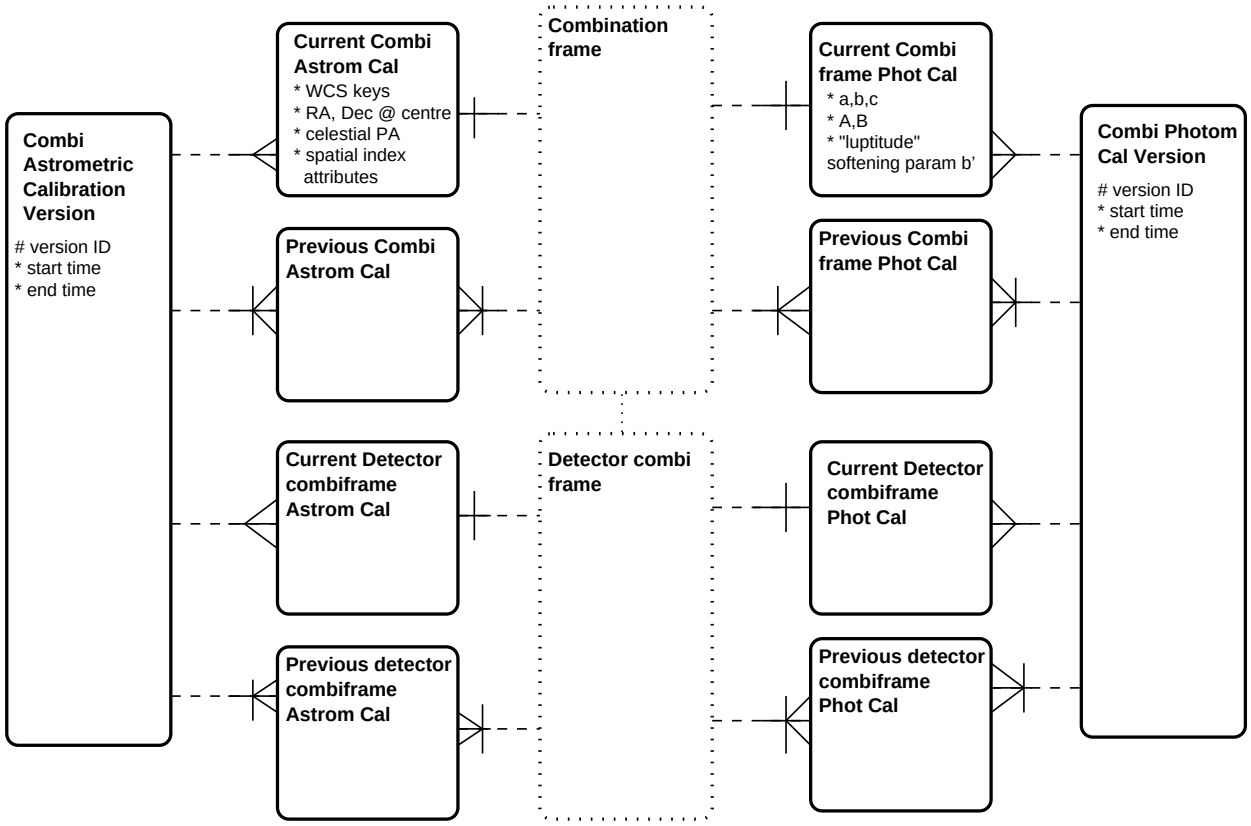

Figure 7: *Entity–relationship model for WFCAM combiframe calibration data.*

- combination frames consist of observations taken at varying airmass so any airmass dependence is either undefined or based on an effective airmass;
- single combination frames have directly associated astrometric and photometric calibrations;
- seeing and sky noise characteristics will vary from combiframe to combiframe, requiring individual values of the 'luptitude' softening parameter b'.

5.6 WFCAM catalogue data

5.6.1 Generalised ERM for WFCAM catalogues within the WSA

Figure 8 illustrates a generalised ERM for WFCAM catalogues that is applicable to all archived programmes. The following points are worthy of note:

- CASU standard processing results in a set of detections on every science image (multi- or combiframe). Every detection is associated with an image and is uniquely identified via a sequence number within that image's detection list, and by the UID of the image. In this way, source detections are intimately linked to their progenitor image and all associated metadata. Uncalibrated instrumental quantities will be stored in the detection list; a 'current/old' flag attribute will distinguish repeat measurements (eg. from re-ingested frames or updated stacks etc.);
- a merged source entity will be created using a merging procedure (one of 20 curation tasks – see Section 9) to create a single multi–colour, multi–epoch record to the prescription available for a given programme (these prescriptions are stored as one of several curation entities – see

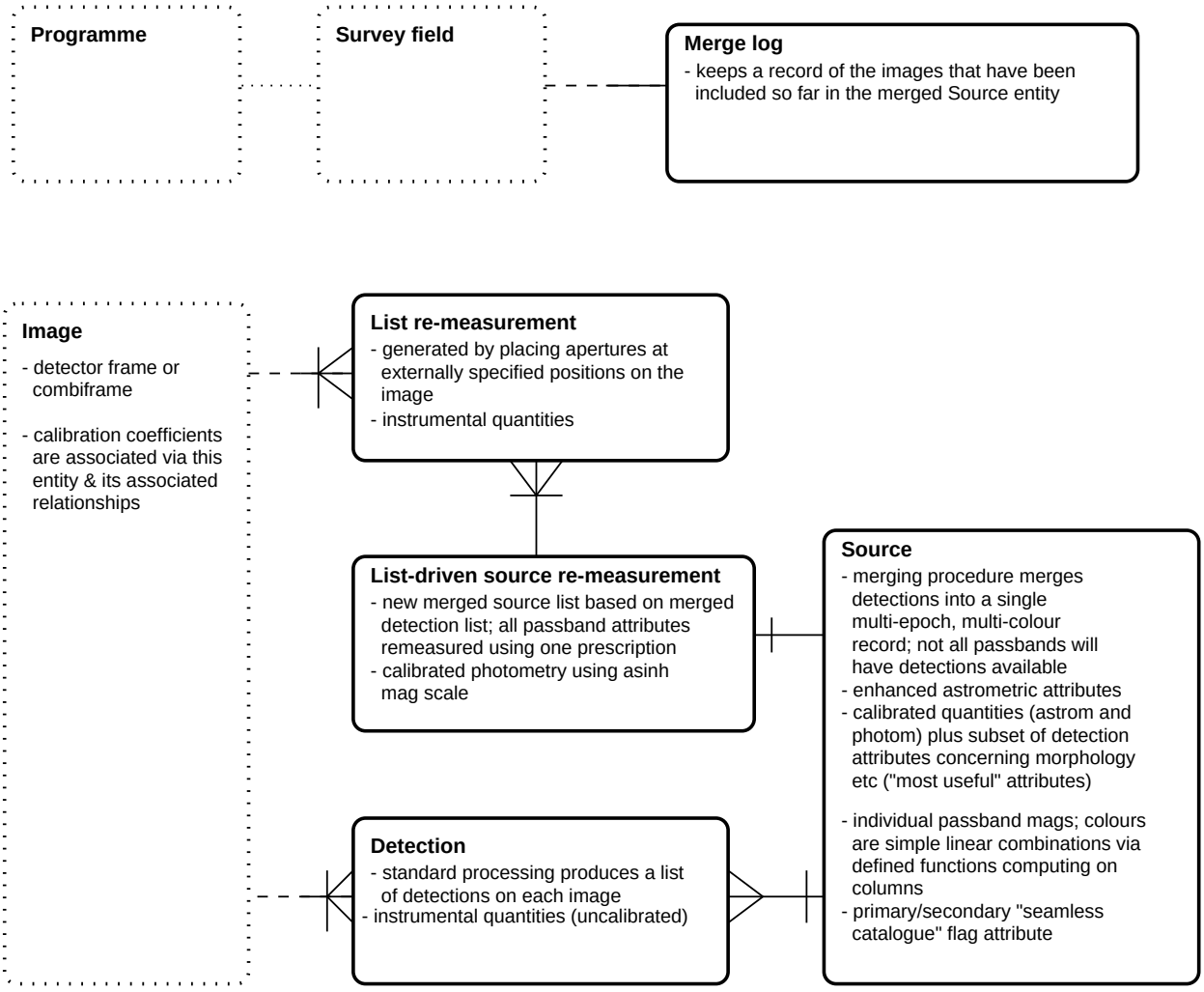

Figure 8: *Generalised ERM for WFCAM catalogue data of all archived programmes.*

Section 5.8). The merged source list will include a primary/secondary attribute to identify duplicates (eg. from overlap regions between survey fields). Enhanced, calibrated astrometric solution attributes will be included based on all available position measures at individual epochs. Additional attributes for a full–blown astrometric solution will also be included: α_0, δ_0, σ_{α_0}, σ_{δ_0}, μ_α, μ_δ and errors, epoch, χ^2 of astrometric fit, and number of position measurements. Calibrated photometric attributes will be included along with the subset of the 80 standard detection attributes (see ICD Section 5.2.2) considered most useful for science exploitation, ie. excluding attributes 1–6, 10–19, 56, 57 and 65–68, leaving 58 attributes per passband. Calibrated fluxes will be stored in the merged source list, the calibration being based on the most recently derived photometric calibration; colours (eg. J–K) will not be explicitly stored as columns but will be specified via defined functions computing on the individually stored passband magnitudes.

- for every archive programme, a merge log will be kept in an entity linking to the programme and survey field entities. This log will have attributes programme ID, survey field ID, and image UID for each image whose detections have been merged into the merged source list along with a new/old flag (for the subsequent list–driven re-measurement curation task – see later). The final attribute will be a merge software version, to keep track of any necessary re-merging when the software has been updated.

- the other two entities in this generalised model contain list re-measurements for a particular programme's frame set. The idea here is that the merged source list will contain gaps (ie. some passbands will not have detections corresponding to those in other passbands), and furthermore merged detections will be based on attributes produced from analysis of individual passband data in isolation. The concept of list–driven source analysis is to take the merged source list, and to define a master list of positions, apertures, profiles and deblending criteria and then apply them uniformly to all available passbands. This procedure results in a set of list–driven re-measurements in each passband which will be written to a 'list re-measurements' entity analogous to the detection entity (and therefore relating back to the source image in a similar way) but with a different set of attributes [TBD] and with fluxes that *may be negative* (due to random errors). Similarly, the list–driven re-measurements, which are generated across all appropriate passbands in one go, will be written into a source catalogue analogous to the merged source detection entity, but again with a different set of attributes [TBD] and most significantly, calibrated fluxes based on the SDSS 'luptitude' scale that can handle negative fluxes [8]. The relationship between the merged source detection entity and list–driven remeasurement source entity is one–to–one since every merged source detection will give rise to a list remeasurement across the passband set for a particular programme.

5.6.2 ERMs for the UKIDSS LAS, GPS and GCS programmes

Figure 9 shows ERMs for the wide–angle, shallow (unstacked) UKIDSS LAS, GPS and GCS subsurvey programmes. The main point to note here is that a single detection list will be maintained for these three surveys since they share common attributes for single passband detections, while three separate merged source catalogues will be maintained for the three subsurveys since they have different combinations of passbands and, therefore, different attribute sets. Similarly, three separate merge logs and three separate list–driven source remeasurement entities will be maintained for these programmes.

5.6.3 The UKIDSS DXS and UDS programmes

Figure 10 shows the relational model for the two deep, small area programmes in UKIDSS: the DXS and UDS; both will have a separate set of entities as illustrated. The main difference between the DXS/UDS

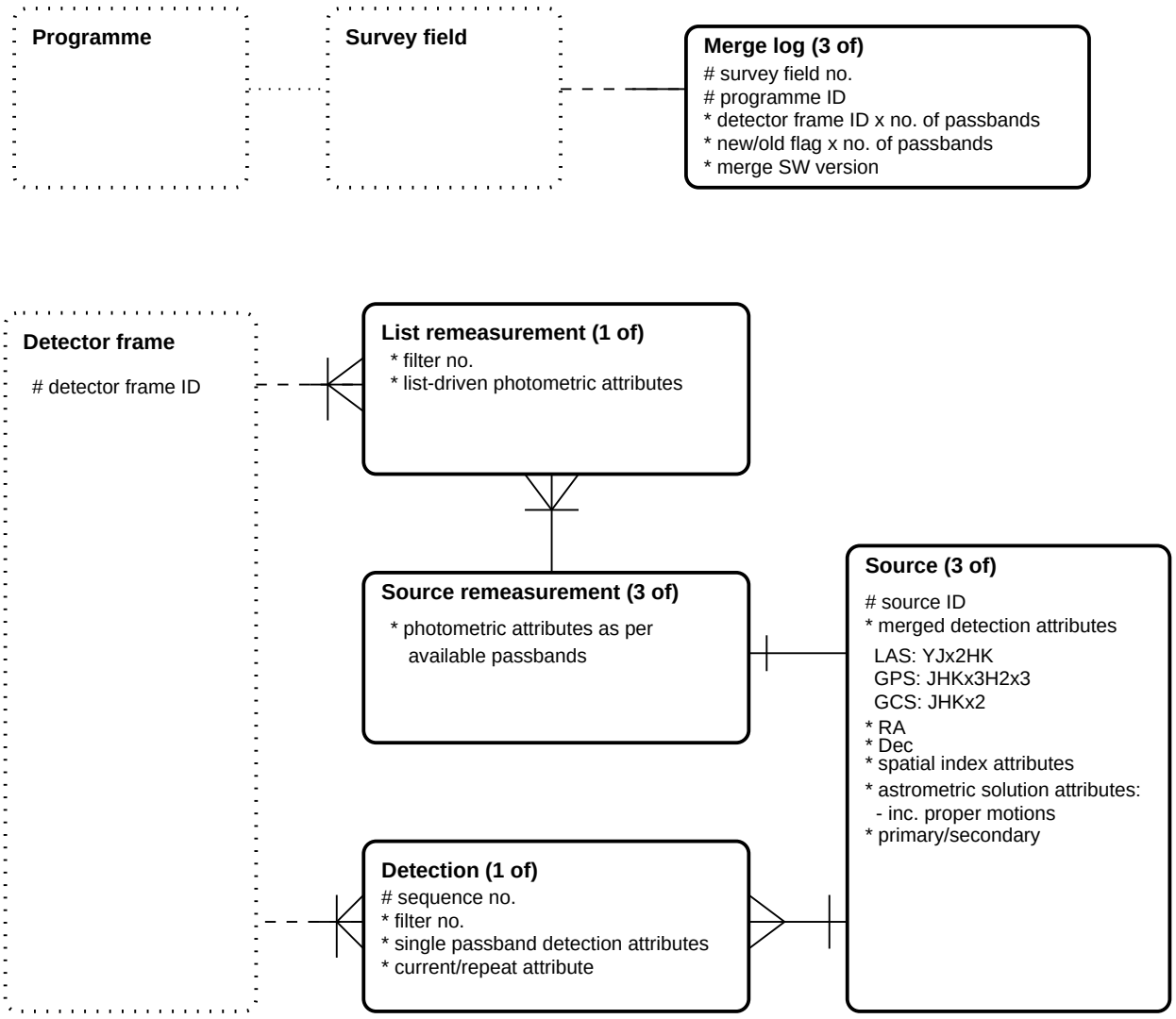

Figure 9: *Entity–relationship model for the UKIDSS LAS, GPS and GCS.*

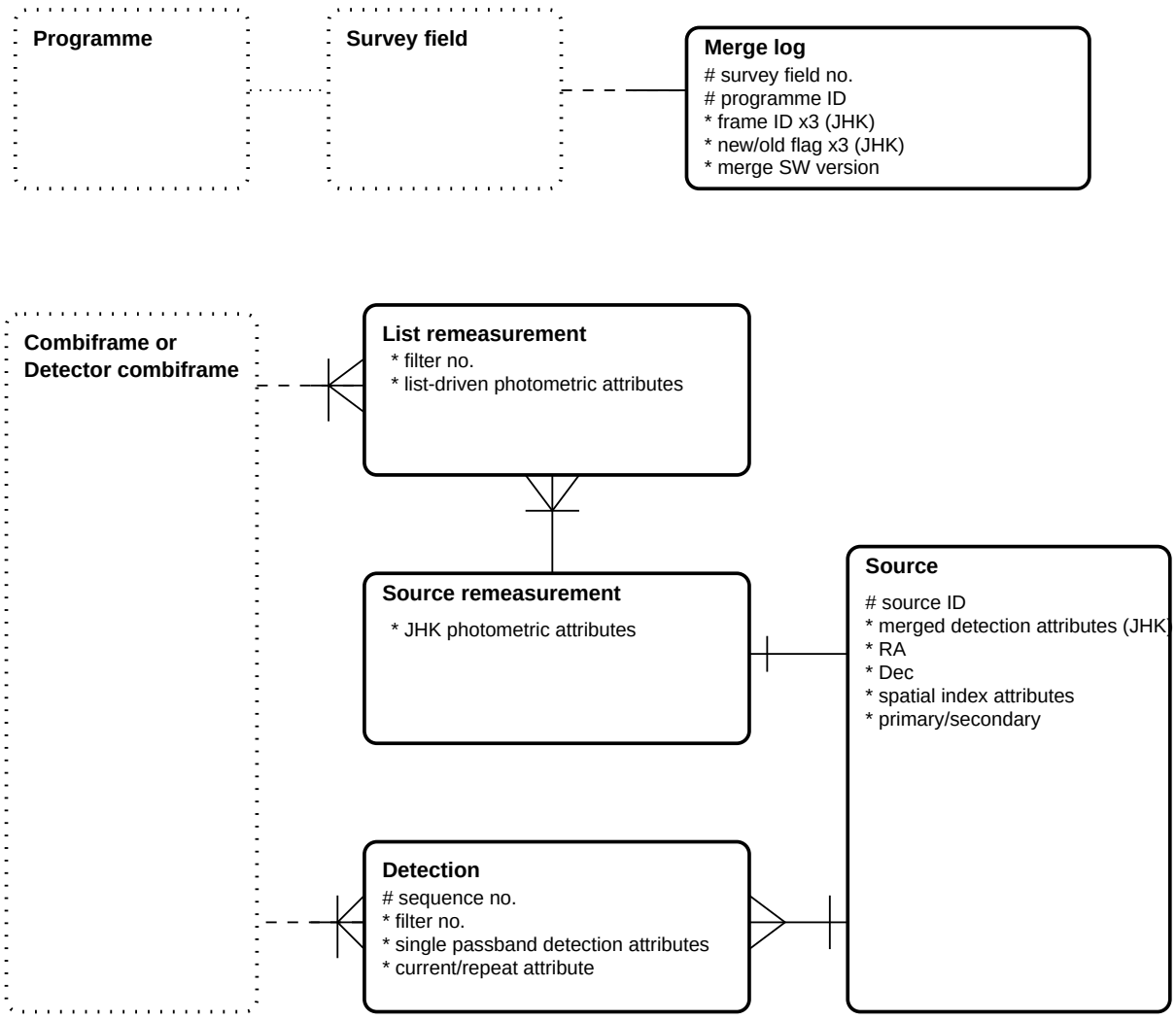

Figure 10: *Entity–relationship model for the UKIDSS DXS and UDS.*

and the previous three UKIDSS programmes is that the former will have JHK passbands only. The DXS links back to the combiframe entity while the UDS links back to the detector combiframe entity. List–driven source re-measurements and merged source entities will be curated, as before.

5.7 Non–WFCAM data (catalogues and images)

Figure 11 illustrates the general relational model for non–WFCAM data that will be held in the WSA for the purposes of joint querying. Examples of non–WFCAM data that are required to be held (see the SRAD; AD01) are legacy photographic all–sky surveys (eg. SSS, USNO); current optical and infrared surveys (eg. SDSS, 2MASS); complementary optical imaging surveys for the UKIDSS programmes; and finally survey catalogues from more distant and diverse wavelengths (eg. FIRST; ROSAT–ASS). The entities in Figure 11 are described and related as follows:

- for each non–WFCAM survey, a catalogue will be stored (eg. SkyServer.PhotoObj from the SDSS or the SSA merged source catalogue);
- a table of 'cross–neighbours' will be created for every source in the non–WFCAM list that is near to a given WFCAM source in the merged source detection list for the appropriate programme(s). The maximum angular radius for neighbours will be set according to appropriate

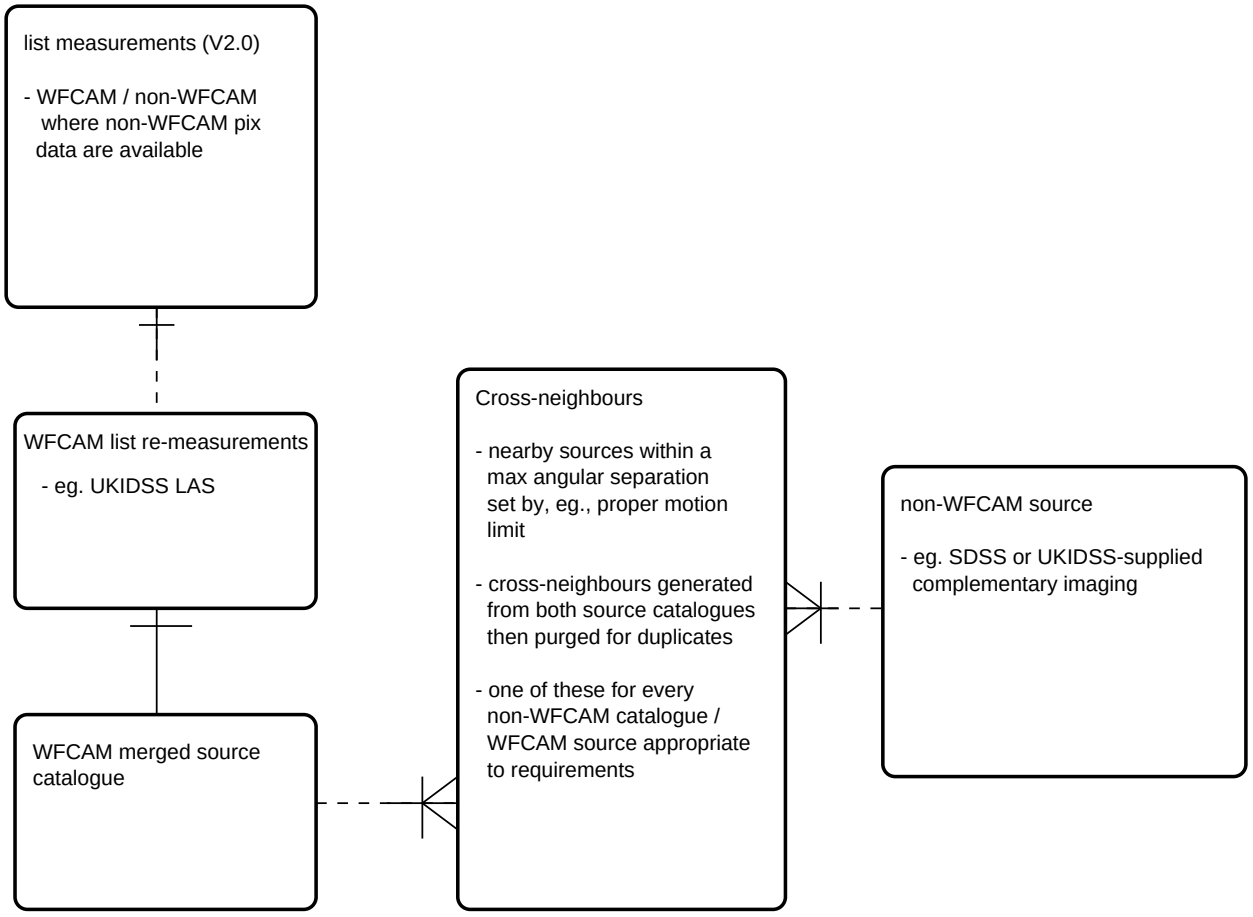

Figure 11: *Generalised entity–relationship model for non–WFCAM survey data that will be held in the WSA.*

science drivers, eg. expected proper motion displacement between the catalogues, or in the case of large error boxes on the non–WFCAM data, a 5σ positional tolerance. Every WFCAM source may have one or more non–WFCAM cross–neighbours; likewise every non–WFCAM source may be cross–identified with one or more WFCAM sources; the UID of the cross–neighbour pair is the combination of UIDs in the WFCAM and non–WFCAM source lists;

- for the V2.0 WSA, a set of list–driven re-measurements in the non–WFCAM pixel data (where available; eg. SDSS) will be made based on positions, apertures, profiles and deblending specified from the WFCAM merged source. Each list re-measurement has a one–to–one relationship via the appropriate WFCAM list re-measurement entity.

5.7.1 Example: the UKIDSS LAS and its relationship to the SDSS

The relationship between the UKIDSS LAS and the SDSS is illustrated in Figure 12 as an example of the connection between a programme of WFCAM data and a non–WFCAM external survey dataset. Note that the LAS/SDSS list remeasurements are a V2.0 requirement while intra–WFCAM passband list re-measurements are a V1.0 requirement. Every LAS list remeasurement *may* have a corresponding SDSS list remeasurement, depending on availability of the SDSS ugriz corrected frames that are part of the SDSS incremental data releases.

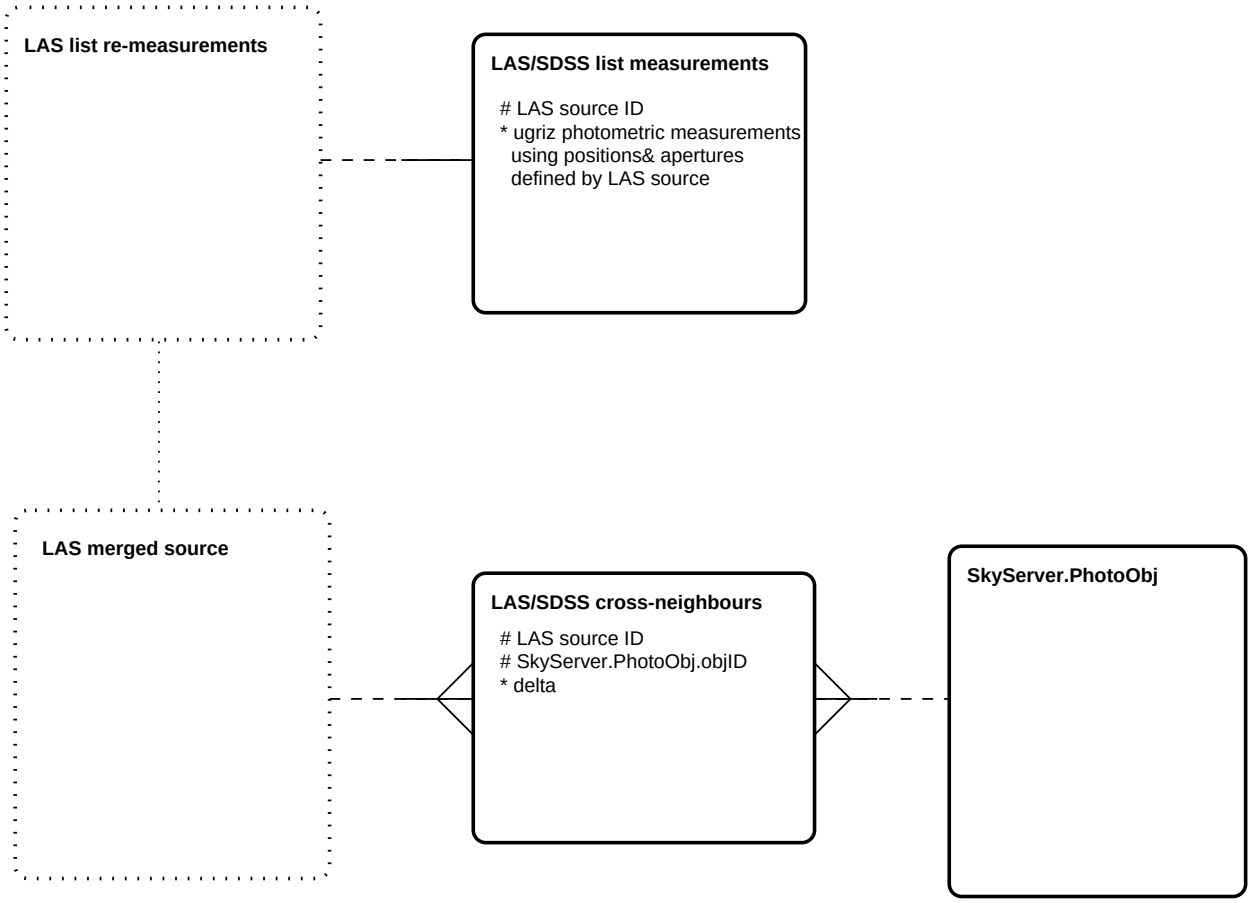

Figure 12: *Entity–relationship model for the UKIDSS–LAS/SDSS surveys.*

5.8 Curation ERMs

Figure 13 details the curation entities that will be kept (in addition to the merge logs associated with each programme, discussed previously). These form a 'star schema' around the central archive entity that describes each archived programme. The entities are as follows:

- a set of *Curation tasks* (described in the next Section) is defined for the archive;
- each curation task has one or more timestamps in a *Curation info* entity recording when it was last executed for a given programme, where each curation timestamp is uniquely identified with the task and programme;
- each programme may have a required *Final curation task* defined for it — at release time, the time stamp of that task will be verified to be the most recent of the set of tasks time–stamped for that programme;
- every programme may have one or more time–stamped release dates/descriptions associated with it in entity *Release dates*;

The other entities in Figure 13 define the action of several of the curation tasks:

- *Required joins* and *Required list driven products* are defined, where appropriate, for each programme;
- the required combination of filters for each programme is defined in an entity which references filter descriptions.

More details of the curation procedures and software are given in Section 9.

5.9 Arrangement of entities within the archive

The previous Sections describe the various entities that will be used to store and track information in the WSA. However, in Section 5.1 we described the requirement for several independent databases to archive open–time observations, to ensure proprietorial rights for all data are protected, and yet to allow unfettered 'world' access to the international astronomical community once proprietorial periods have expired. The various databases needed to meet these requirements will be arranged as follows:

5.9.1 Internal, incremental database

WFAU will curate an internal access–only database using a 'load server' (see HDD; AD05). Copies of all entities discussed previously will be held in this database, which will ingest new data on a daily basis.

5.9.2 Open time observations databases

Open time (eg. PATT) programmes will be curated in the same way as the large–scale UKIDSS surveys within the offline database. Merged source catalogues will be produced where possible and where required, as can list–driven remeasurements between the WFCAM passbands if so desired by the PIs. Once all the data for a particular programme are ingested and merged into the relevant programme entities within the archive, they will be copied into an individual database on the public server with access restricted to the PI via password protection.

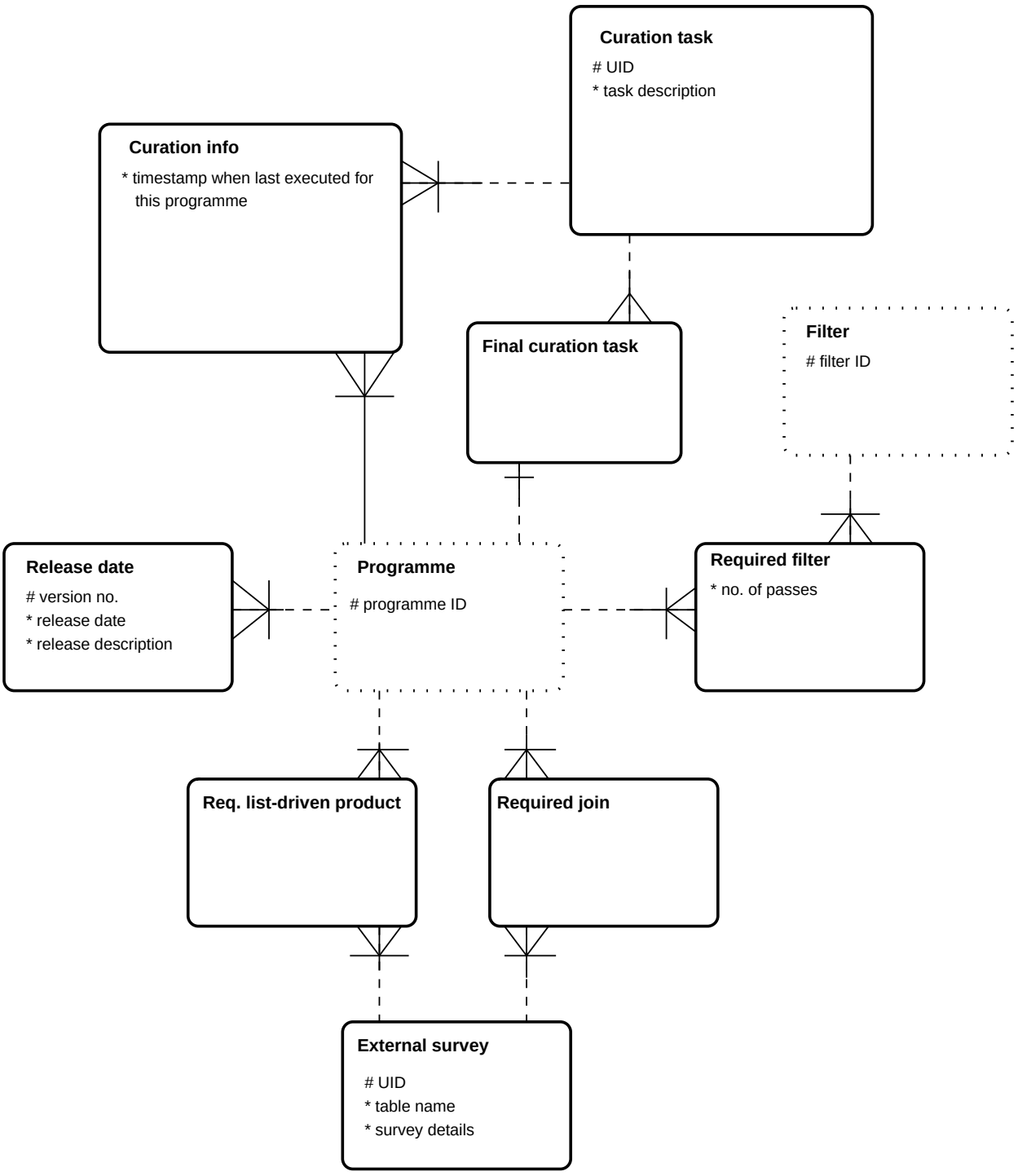

Figure 13: *Entity–relationship model for curation information within the WSA.*

5.9.3 UKIDSS release database

All internally curated UKIDSS entities will be periodically (after each WFCAM observing block) frozen, backed up and released by copying into a UKIDSS release database on the public–access catalogue server.

5.9.4 World readable database

All WFCAM data will become ultimately world–readable as proprietorial periods (where they exist) expire. When a significant amount of data have become world readable, a 'world' release will be made into a database with unrestricted access. This will involve creating world readable subsets of all currently released UKIDSS database catalogues in the world database based on the present release date, the observations date and the proprietorial period and copying in open–time catalogues that have passed their proprietorial expiration. Table 'views' will be created (see the next Section) to create a logical overview catalogue that can be queried for general interest usages not aimed specifically at a particular programme.

6 Tables, fields, keys and views

The V1.0 WSA will be implemented in SQL Server. This Section describes the details of the database schema files via examples, and illustrates the prototype WSA databases. The following conventions have been adhered to in generating schema files:

- SQL scripts are used to create the database tables, and these scripts are placed under CVS version control;
- meaningful table names that relate directlry to the ERM descriptions in the previous Section are employed, but excessively long table names are avoided;
- attribute names are kept meaningful, and correspond as closely as possible to FITS key names in the progenitor FITS files where possible and appropriate;
- attribute units are 'natural' so as to be clear to users – eg. arbitrary scaling to fit attributes into smaller datatypes (for example storing magnitudes in millimag units in 2–byte integers instead of normal magnitude units in 4–byte reals) is not be done;
- where possible, meaningful unique identifiers are used to track the provenance of a record – eg. WFCAM images will have compound UIDs made up from original UKIRT run numbers, date observed and date ingested.

6.1 SQL Server schema SQL scripts

An example extract from the V1.0 SQL Server WSA schema SQL scripts is as follows:

```
--
-- MultiframeSchema.sql
--
-- Database schema file for multiframe images, programme and filter
-- data in the WFCAM Science Archive. Contains sql scripts to create
-- the relevant tables.
```

```
--
-- Original author: Nigel Hambly, WFAU, IfA, University of Edinburgh
--
-- Revision history:
-- Login name of the user who checked in this revision:  $Author$
-- Date and time (UTC) when the revision was checked in: $Date$
-- Login name of the user who locked the revision:       $Locker$
-- CVS revision number:                                  $Revision$
--

CREATE TABLE Programme(
--
--  This table contains details of the observation programmes undertaken
--  with WFCAM and for which observations are stored and curated in the WSA.
--
--        The unique ID number (assigned internally within the WSA)
--        identifying all WFCAM observation programmes is as follows:
--
--          programmeID = 0 : not a programme: used for calibration/confidence
--                            frames common to many programmes
--                      1 : Commissioning data
--                    101 : UKIDSS Large Area Survey (LAS)
--                    102 :   "    Galactic Plane Survey (GPS)
--                    103 :   "    Galactic Clusters Survey (GCS)
--                    104 :   "    Deep Extragalactic Survey (DXS)
--                    105 :   "    Ultra-Deep Survey (UDS)
--                   1000+: Open Time programmes (PI)
--                  10000+: Service programmes;
--
--  Required constraints: primary key is (programmeID)
--
programmeID   int not null, -- UID of the archived programme coded as above
title         varchar(32) not null, -- a short title for the programme,
--            eg. "UKIDSS Large Area Survey"
description   varchar(256) not null, -- a concise description of the programme
reference     varchar(256) not null, -- a reference for the programme,
--            eg. "http://www.ukidss.org/surveys/surveys.html"
propPeriod    real not null -- the proprietory period for any data taken
--            for this programme in years, eg. 1.0 for open time.
       )

CREATE TABLE Filter(
--
-- This table stores details of the WFCAM filters.
--
--        The unique ID number (assigned internally within the WSA)
--        identifying all the WFCAM filters is as follows:
--
--          filterID  =  1 : Y filter
--                       2 : J   "
--                       3 : H   "
```

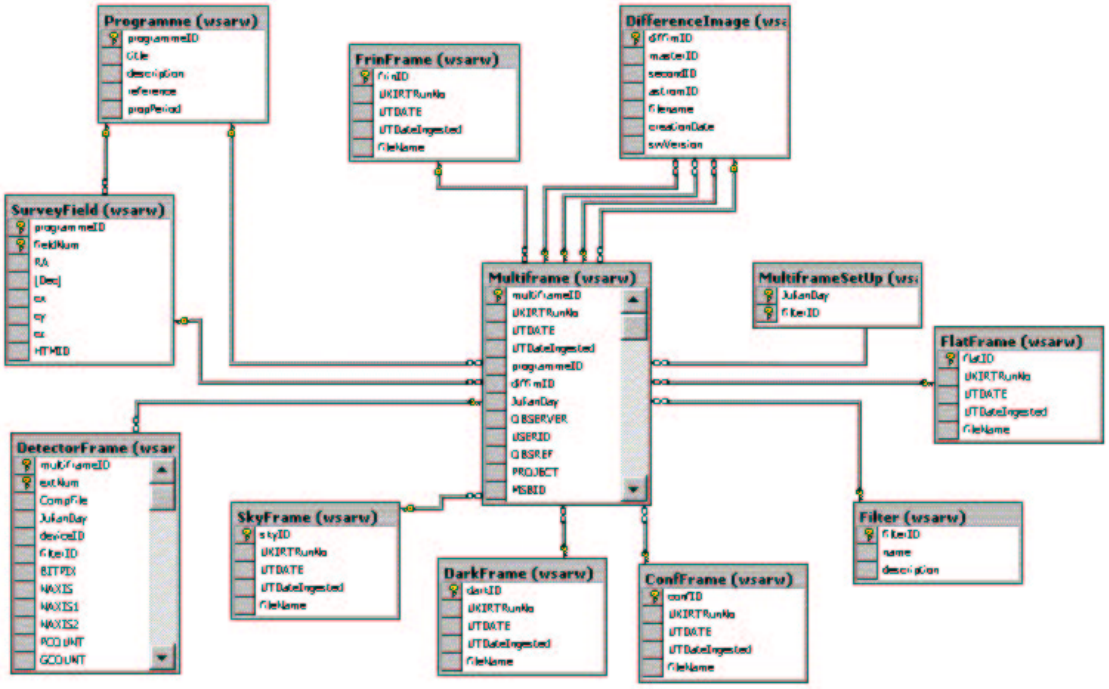

Figure 14: *Multiframe tables loaded into SQL Server and ready for data ingest.*

```
--                      4 : K    "
--                      5 : H2   "
--
--  Required constraints: primary key is (filterID)
--
filterID     tinyint not null, -- UID of filter, defined as above
name         varchar(16) not null, -- the name of the filter,
--           eg. "MKO J", "UKIDSS Y", "Narrow band H2" etc;
description  varchar(256) not null -- a concise description of the filter
--
.
.
.
```

All tables and attributes will be defined in this way. Attributes have datatypes as specified; a short comment in the script gives additional description, comments and/or units as appropriate. Appendix 10.3 gives more detailed examples for the V1.0 WSA multiframe, corresponding calibration tables, and UKIDSS LAS tables which will reside in the database.

6.2 Constraints

Constraints (in the form of primary keys and foreign keys) will again be specified via SQL scripts. An example is given in Appendix 10.4 which details the constraints on the tables created using the scripts in Appendix 10.3. The resulting 'star schema' picture of the tables and constraints as implemented in SQL Server, and ready for data ingest, is shown in Figure 14, where for clarity only the multiframe table and all the nearest associated tables are shown.

6.3 Views

Views are *logical* tables (rather than physical tables on disk) expressed via an SQL script that is executed when a query is requested on the view. Views will be implemented in the WSA as follows:

- A 'world' view will be created over all the released tables (UKIDSS and open time programmes) for general use. This table will contain the subset of common attributes from the covered tables, ie. astrometric information (celestial co-ordinates, proper motion, astrometric errors) and photometric information (JHK magnitudes). Where a given programme table does not contain this minimal information, it will not appear in the view;
- A UKIDSS view will be created for UKIDSS users to query when not specifically interested in a particular subsurvey. Again, the view will consist of the subset of attributes common to all individual UKIDSS tables: JHK photometry and celestial co-ordinates.

6.4 V2.0 considerations

As stated previously, the V1.0 WSA will be implemented in SQL Server, while the V2.0 DBMS choice is left open until sufficient experience has been gained with V1.0 to enable an informed decision as to its scalability. The other obvious potential DBMS choices for V2.0 are IBM's DB2, and Oracle. In parallel with our V2.0 implementation work, we are undertaking a comparative study of all three DBMS products with experts from the respective companies to test out scalability issues for Tbyte–scale applications.

All the database design presented so far is applicable to any RDBMS product with, if any, minor changes. Indeed, some facilities exist within SQL Server, Oracle and DB2 to import databases created in the competitors DBMS. To this point in the database design, there will be only very minor differences in the V1.0 and V2.0 systems if the latter needs a different RDBMS choice to that already chosen for the former.

As stated previously and in AD07, the V2.0 hardware/OS/DBMS choice must be specified by the end of Q1 2004 at which point a review milestone occurs.

7 Indexing and other implementation details

7.1 Spatial indexing attributes

One of the main types of query that any astronomical database will encounter is one making use of spherical co-ordinates to locate a small sky region of interest – for example, several usages in the SRAD require position/proximity searches. A primary requirement on database organisation is therefore sorting or some other form of spatial indexing. Many potential schemes are possible, but the V1.0 WSA will employ the Hierarchical Triangular Mesh (HTM) indexing scheme [1] employed in the SDSS and other current surveys.

Briefly, HTM employs a hierarchical quad–tree indexing scheme to code up the positions of nested equilateral triangular patches of sky of decreasing size, starting with 8 triangles covering the whole sky, 4 octants in each hemisphere. The WSA will employ an 8–byte integer index for up to 20 decimal digits which will uniquely specify a position to a resolution of ~ 10 milliarcsec. Where any celestial co-ordinate attribute pair occurs in a table, a corresponding HTM index will also be present (see, for example, table `SurveyField` in Appendix 10.3).

8 Table indexing

To greatly enhance query performance on commonly queried attributes, (eg. position, magnitude, colour etc.) table indices will be created. These indices are employed by the query optimiser at query run time to avoid the DBMS having to search entire tables to execute the query – this is of course standard practice in RDBMS design.

Table indexing is not an exact science, and a certain amount of experimentation will have to be undertaken with real–world WSA data and queries to ellucidate and optimise the nature, and number of the indices created for each table. However there are some general rules that will be adhered to in creating indices.

SQL Server uses B–tree (B≡balanced) indexing, and there are two basic types of index: clustered and non–clustered. A clustered index results in the table data being reordered (ie. sorted) on disk, so only one clustered index can be made per table; otherwise any number of non–clustered indices can be made (subject to disk space limitations). Both types of index can be constructed on multiple attribute sets. In creating the WSA indices, we will adhere to the following general principles:

- when building an index on more than one attribute, the attributes will be put in order of decreasing selectivity;
- the default–generated clustered index will be taken as that based on each table's primary key – SDSS experience [3] is that this is the optimal set–up (for example, the primary key of detections is an UID based on a progenitor frame ID, and hence detections on one frame, which will tend to be accessed together during both curation and use, will be stored together on disk);
- for performance, indices should ideally be held on a different physical disk volume to the data themselves so that the index read is as close to 100% sequential as possible;

The HTM attribute will be included as one of the most selective of each table's attribute set in at least one index, where appropriate.

In order to progress experimentation of indexing questions on large–scale tables in lieu of large amounts of real WFCAM catalogue data, we are experimenting with the SSA. We have defined 20 typical astronomical science queries of the data [4] and are investigating performance optimisation of these via index creation within the SSA database.

8.1 Defined functions

Positional (eg. celestial co-ordinate system transformation; positional associating functions using HTM) have been obtained from [15] as implemented for SkyServer. These have been modified slightly for use in the SSA, and the same functions will be defined within the WSA for use by curation software and users.

8.2 V2.0 considerations

8.2.1 Spatial indexing

The only major difference between the V1.0 and V2.0 implementation is possibly in spatial indexing attributes and functions. Depending on the relative ease of implementation and performance of HTM compared with other potential options (eg. HEALPix [9], or spatial indexing options available as part of a given DBMS like Oracle), a decision will be made on whether to change the spatial indexing in V2.0 from that used in V1.0.

8.2.2 Table subsets for performance gain

A question that is likely to arise in scalability to multi–Tbyte sized tables is one of optimal performance for the most common kinds of queries. This question was addressed in the SDSS–EDR via the concept of *tag tables* [6], where a subset of the most astronomically useful attributes from the master source catalogues where copied into a table of greatly reduced size. Note that *views* are logical tables not physcial, so despite presenting a row subset to the user they are in fact nothing more than a query defined for the full table; indices can only limit row reads, not column reads. We will investigate the most common queries that are executed for the archive, and if it becomes sensible to do so, we will create *materialised views* (analogous to the EDR tag tables) to greatly enhance query performance. An example of such a materialised view might be `LASstars`, where a subset of the main LAS merged source table is created that includes only astrometric, point–source photometric and a limited number of morphological attributes. In this way, the large number of extended–source photometric attribues is excluded, the number of columns in the table is greatly reduced, and query performance will be greatly enhanced.

9 Details of curation procedures and software

For each of the curation tasks identified in the use cases presented in Appendix 10.1, we have followed standard practice in the rational unified process and identified the primary scenarios associated with each case. These are presented on a case–by–case basis in the following Sections in the form of a numbered–steps pseudo–code description along with an activity diagram illustrating the flow of events. Figure 15 shows a key for the diagrams, while Figure 16 illustrates the overall logical flow of the curation tasks (which, within any given curation period, run sequentially). For some of the more complicated cases, algorithmic details are given; in all cases we have identified the archive version number that each task must be implemented for (mainly V1.0) and a relative priority within that version in order to prioritise work over the coding and implementation phase. Implementation details are given for all tasks.

Some general notes on the curation tasks are as follows:

- we plan to implement all curation tasks except CU6, CU15, CU16, and CU18 to CU20 using Perl scripts to 'glue' C/C++/Java applications (existing, supplied or in some cases specially written);
- a few tasks will execute on the Linux server side (those involving to data transfer/ingest and pixel manipulation) while the rest will execute on the DBMS load–server side. For details of the hardware configuration, see the Hardware/OS/DBMS design document (AD05);
- several of the curation tasks (eg. CU3) involve the transfer of bulk data (anywhere from Mbytes to a few Gbytes in size) from the Linux server that receives the data from CASU and the Windows/SQL Server 'load' server. This requires communication between processes running on the two machines, which can be achieved using sockets – the Perl 'sockets' module will be used to achieve inter–process communication;
- curation tasks involving database updates (eg. CU4) require foreign key referencing to be updated also. Some experimentation will be needed to determine the optimum way to do this. It may be that foreign key constraints will be 'dropped' before updating tables and then recreated from scratch, or it may be better to allow the DBMS to update references during the loading process. The latter scenario will likely be used initially (V1.0) and then experimentation and results will inform the final, scalable solution (V2.0 and beyond).

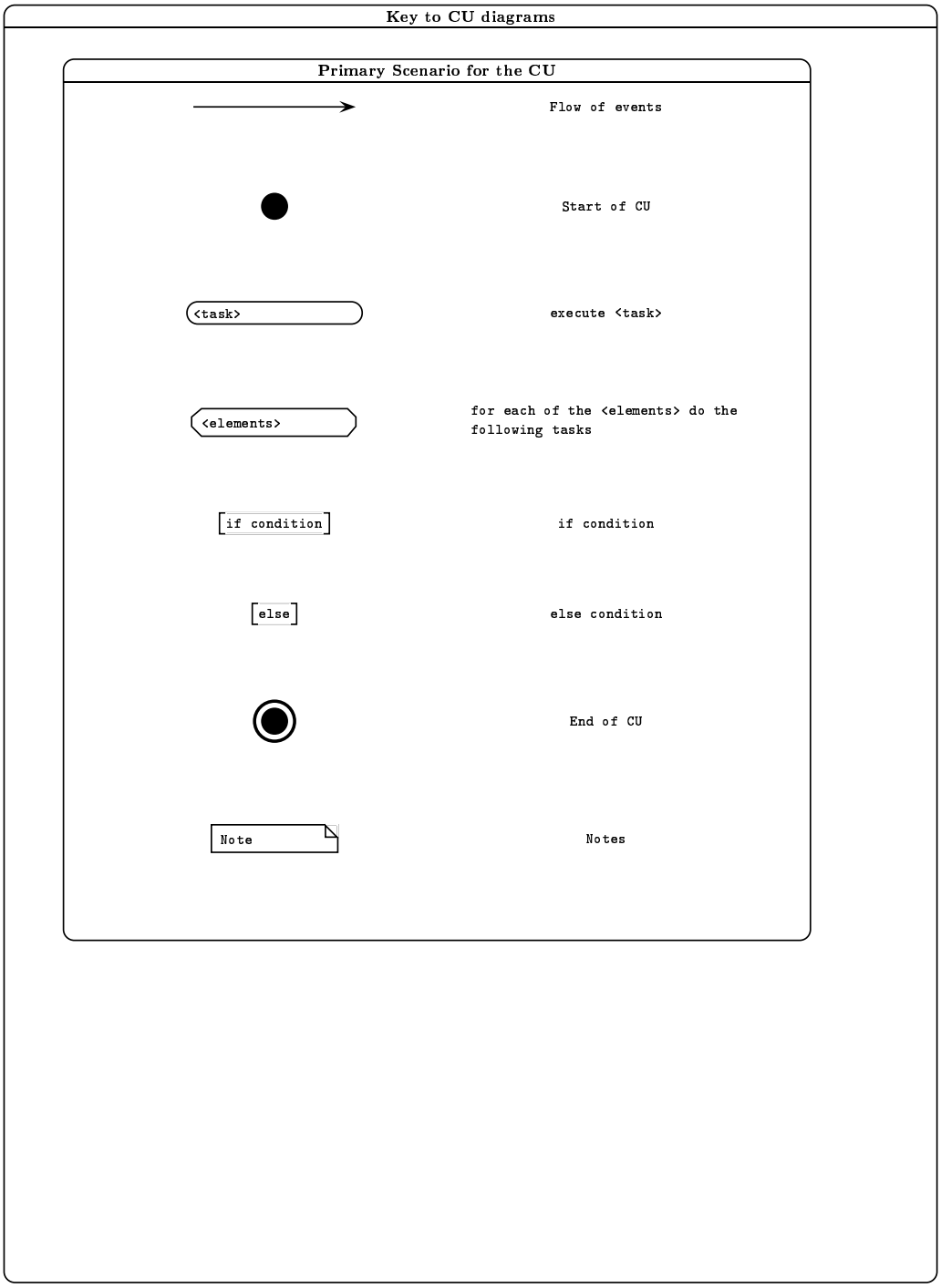

Figure 15: *Key to the curation activity diagrams.*

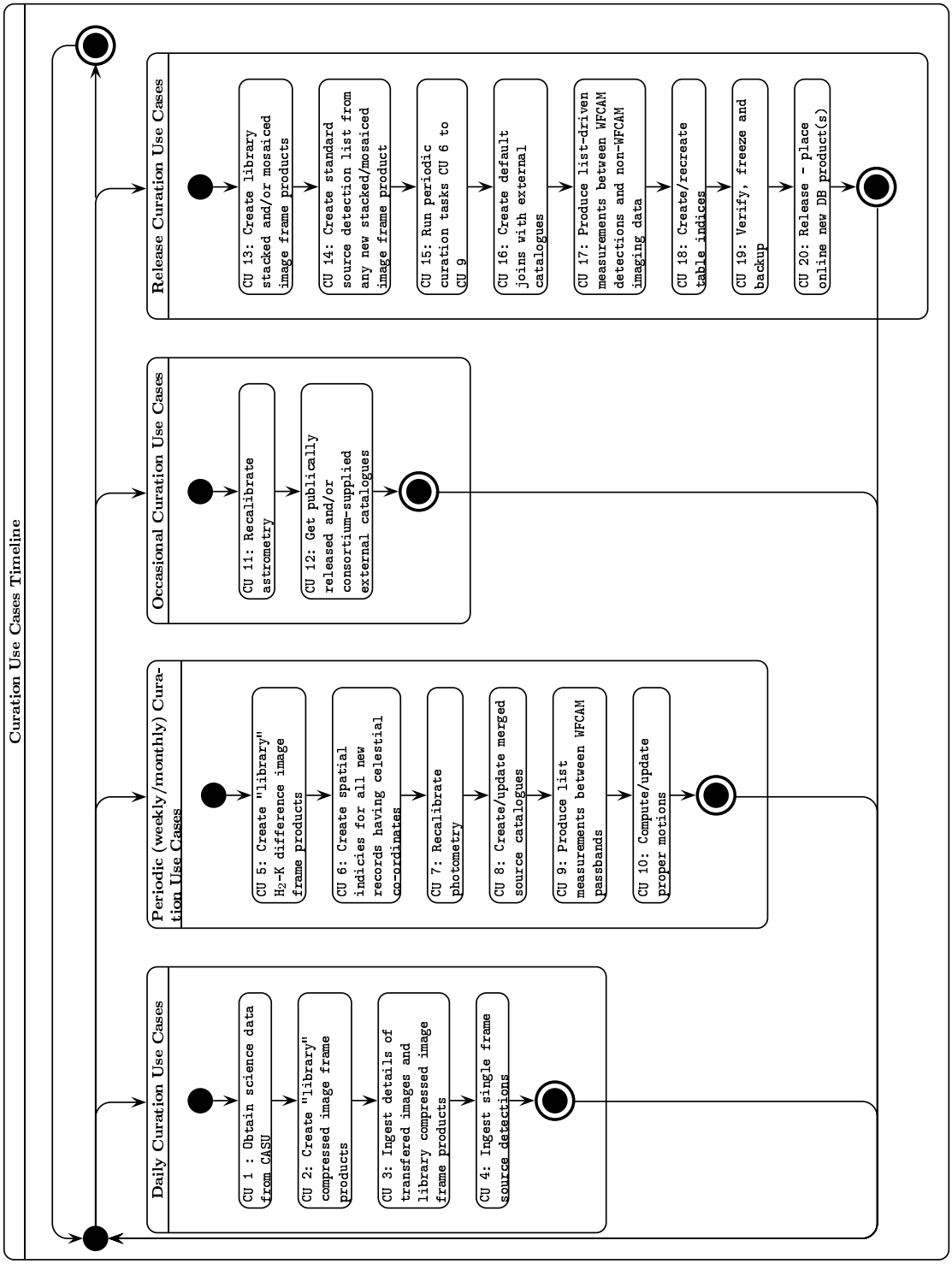

Figure 16: *Top–level activity diagram for the curation tasks.*

9.1 CU1 : Obtain science data from CASU

```
CU1 : Obtain science data from CASU : Primary Scenarios

Precondition(s): Any previous transfer is complete;
                 Time is after normal working hours;
                 Network connection closed

Flow of events:

1. Initiate network connection

2. For each new source directory

   a) Test source directory for read-readiness

   b) For each file in a read-ready source directory

      i) Check WSA filesystem for existing file (rerun duplicate)

      ii) Transfer the file to the destination directory

      iii) Verify that the transfer was successful

      iv) log the file transfer in transfer log file

   end

end

3. Close network connection

4. Copy transfer log to permanent area to save a record of all transactions

5. Update DB system curation log for CU1 for all relevant programmes

Postcondition(s): Network connection closed;
                  Intermediate log file closed

Special requirement(s): V1; Priority 1
```

Implementation details: files will be copied straight to their final destination to avoid further IO over–heads; implementation will be via a Perl script running on the mass–storage Linux server side; the CU1 script will be invoked by a Linux shell script 'cron' job to automatically execute each night. The final step will communicate with the DBMS server via the Perl DBI (database interface) module.

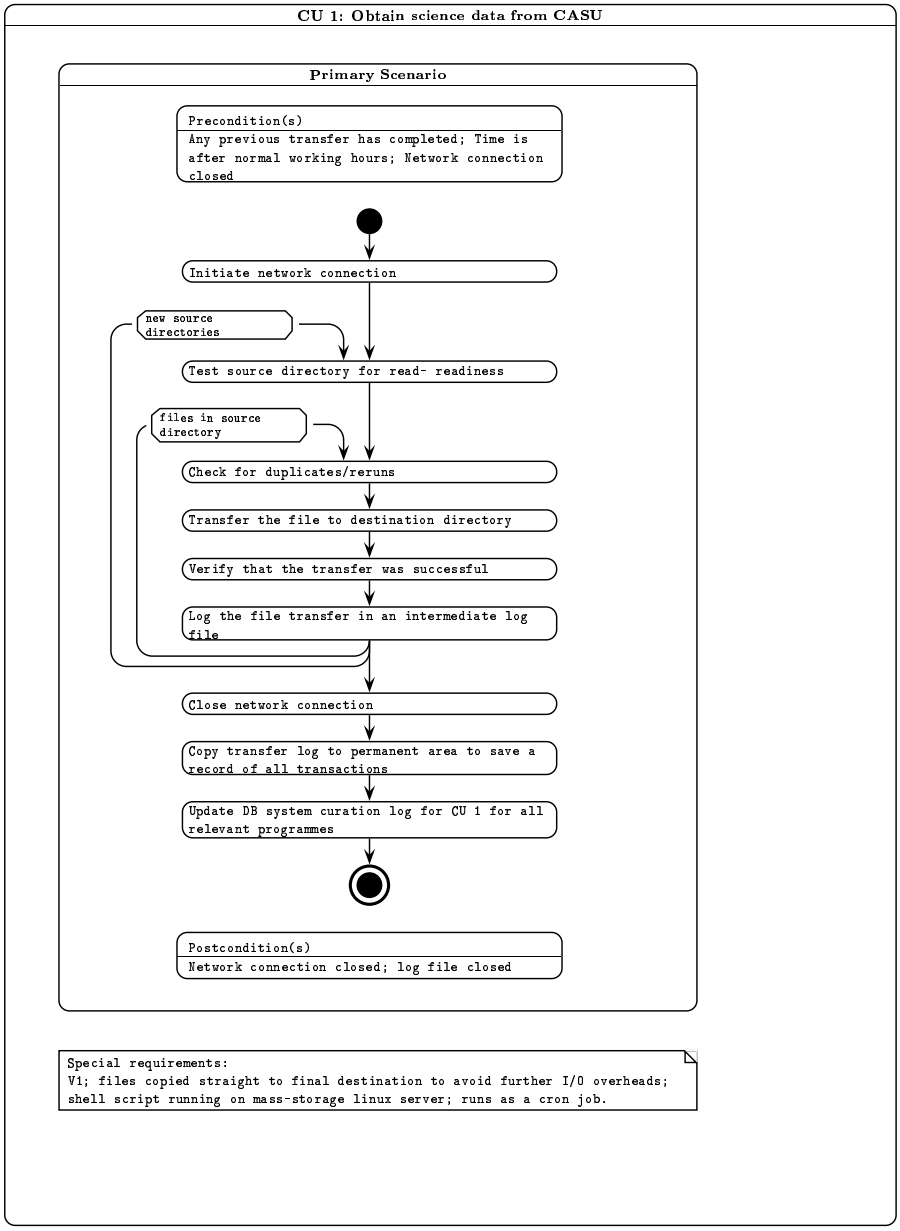

Figure 17: *Activity diagram for curation task CU1.*

9.2 CU2: Create library compressed image frame products

```
CU2: Create "library" compressed image frame products : Primary Scenarios

Precondition(s): Log of recently transfered files exists and is complete;
                 No transfer is currently underway

Flow of events:

1. For each image file listed in the log

   a) For each image array in the image file

      i) Read in image array

      ii) Compress image

      iii) Write compressed file

      iv) log newly created compressed image file

   end

end

2. Update DB system curation log for CU2

Postcondition(s): Log file closed

Special requirement(s): V1; priority 2
```

Implementation details: this task will again run on the mass–storage Linux server side. Perl scripting will be used to 'glue' the necessary procedures together; existing code (eg. ImageMagick) will be employed to compress FITS to JPEG. Communication with the DBMS for step 2 will be as for CU1.

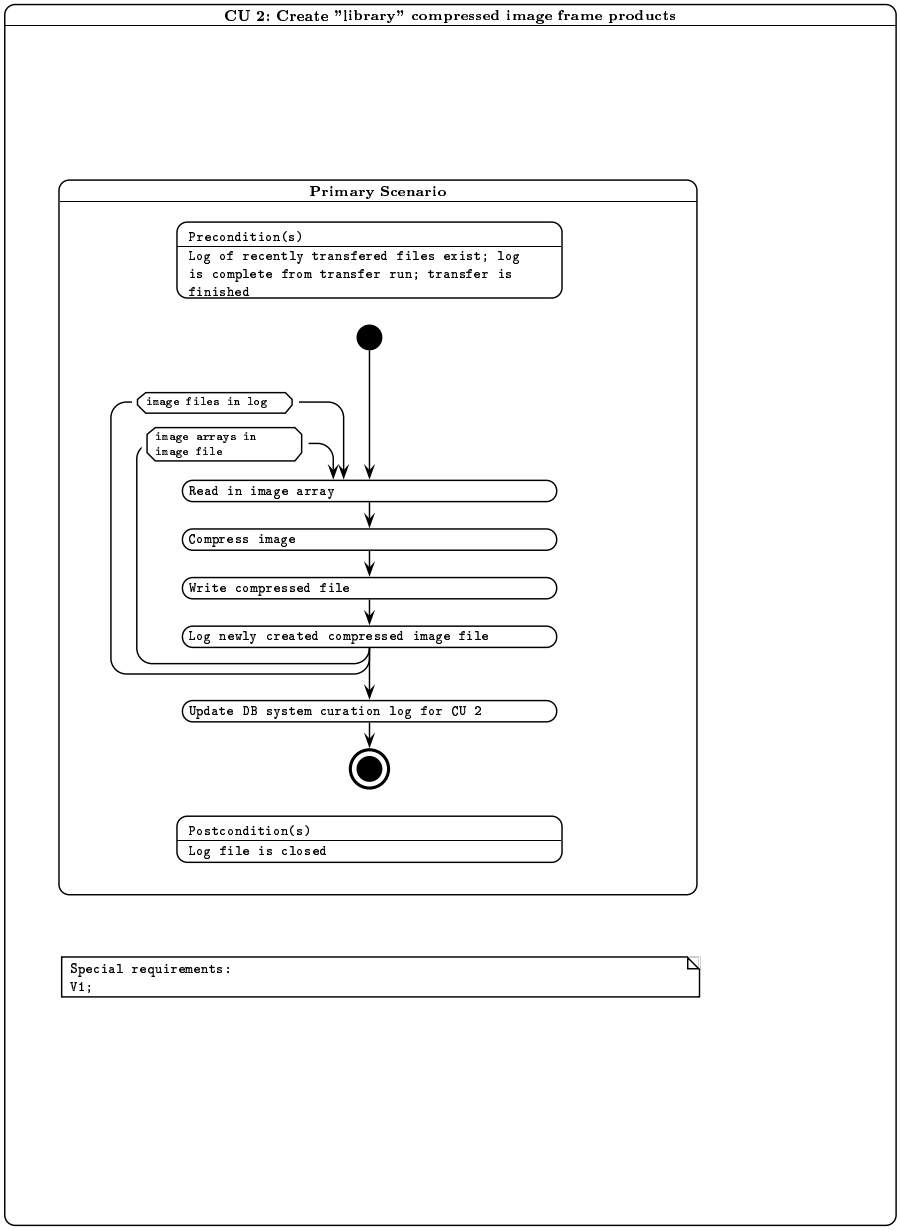

Figure 18: *Activity diagram for curation task CU2.*

9.3 CU3: Ingest details of image products

```
CU3: Ingest details of transfered images and library compressed image frame
     products: Primary Scenarios

Precondition(s): Logs of image products are ready;
                 DB system is ready for updates

Flow of events:

1. Write-lock the DB system

2. For each image container file logged

   a) Determine image type (multiframe, combiframe, calibration, ...)

   b) For each image/extension

      i) strip descriptor keys to appropriate intermediate file

   end

   c) For each associated catalogue file

      i) Strip additional descriptor keys to appropriate intermediate file

   end

end

3. For each intermediate file

   a) Bulk load information into appropriate DBMS table

4. Update foreign keys in DBMS

5. Delete intermediate files

6. Update DB system curation log for CU3

Postcondition(s): DB system is write-unlocked

Special requirement(s): V1; priority 1
```

Implementation details: a Perl script running on the Linux server side will execute a C/C++ application to strip metadata to intermediate ASCII CSV files. The procedure for bulk loading into the DBMS is yet to be decided.

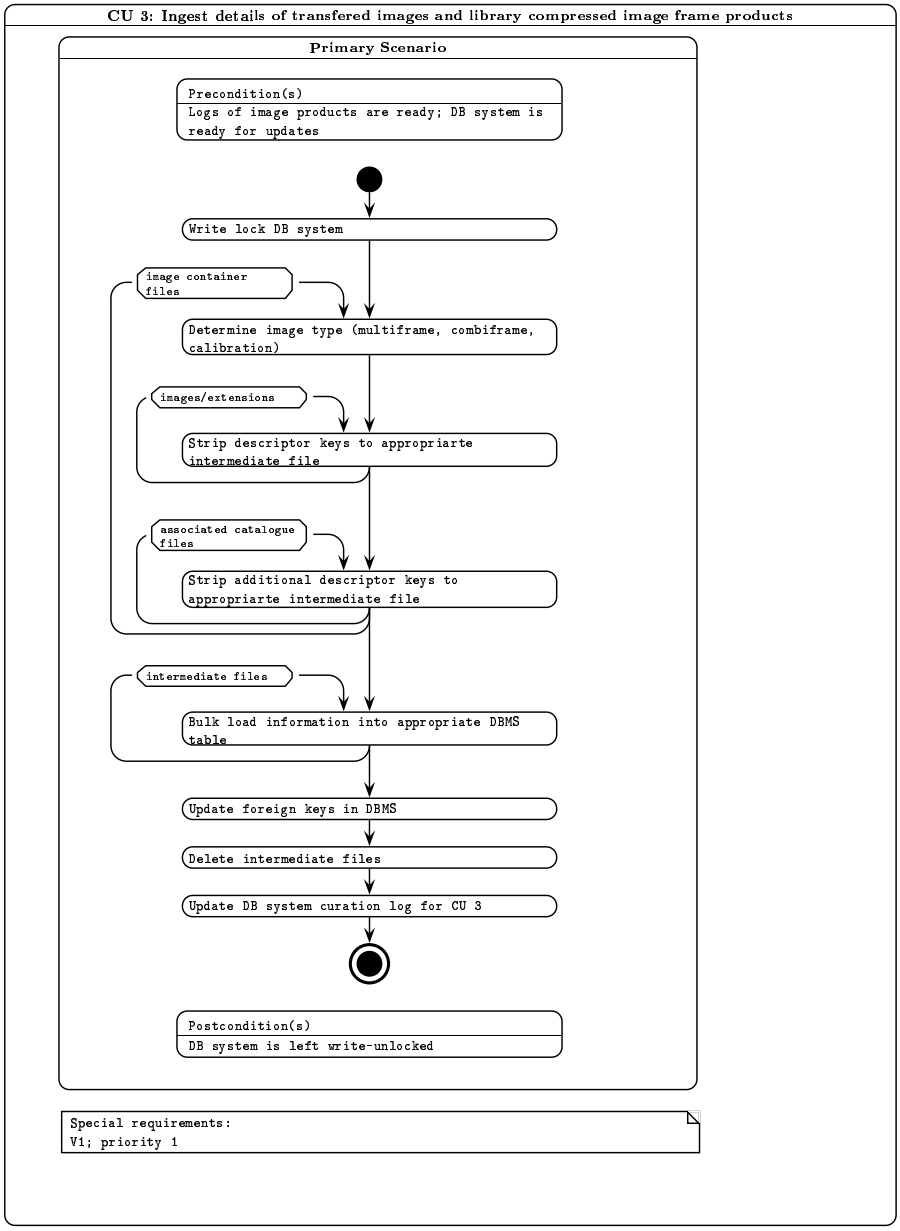

Figure 19: *Activity diagram for curation task CU3.*

9.4 CU4: Ingest single frame source detections

```
CU4: Ingest single frame source detections : Primary Scenarios

Precondition(s): Log of transfered files is ready and complete;
                 DB system is ready for updates

Flow of events:

1. Write-lock DB system

2. For each catalogue container file

   a) Determine destination programme

   b) For each catalogue extension

      i) Strip catalogue attributes to appropriate intermediate file

   end

end

3. For each intermediate file

   a) Bulk load intermediate file into DBMS

end

4. Update foreign keys (?)

5. Delete intermediate files and log files

6. Update DB system curation log for CU4

Postcondition(s): DB system is left write-unlocked

Special requirement(s): V1; priority 1
```

Implementation details: CU4 will be implemented to step 2 in the same way as CU3, employing the same software modules where possible. For step 3, we will clearly need an efficient bulk load solution – this needs further investigation.

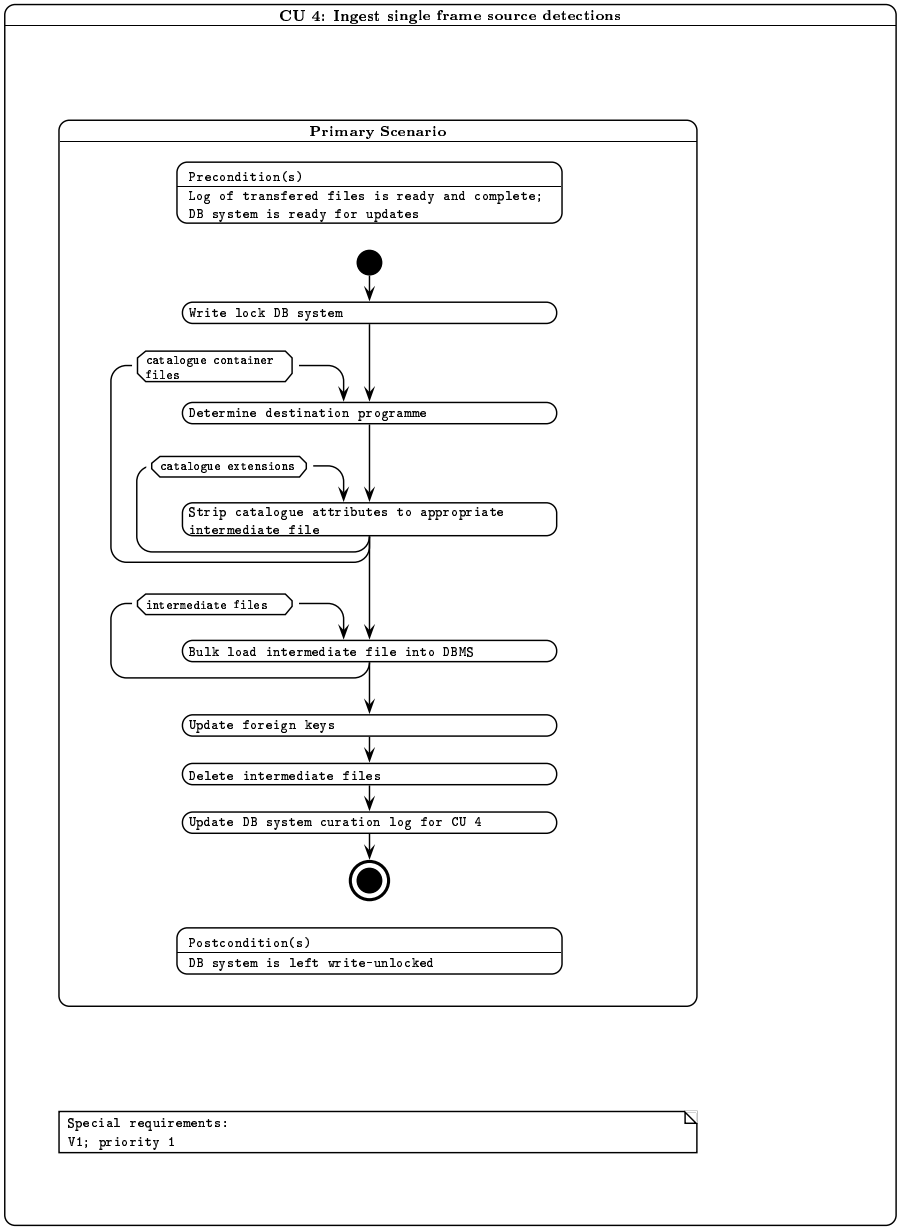

Figure 20: *Activity diagram for curation task CU4.*

9.5 CU5: Create library H2-K difference image frame products

```
CU5: Create "library" H2-K difference image frame products : Primary Scenarios

Precondition(s): DB system is ready for updates

Flow of events:

1. Write-lock DB system

2. Query image DB to determine list of new images available for this task

3. For each image pair

   a) create difference image

   b) insert difference image into filesystem

   c) Update image database to reflect change

4. Update foreign keys (?)

5. Update DB system curation log for CU5

Postcondition(s): Image DB system is left write-unlocked

Special requirement(s): V1; priority 3
```

Implementation details: a Perl script running on the Linux server side will use Perl DBI to query the DBMS server for images appropriate for default difference imaging (ie. those H_2 and K band images from the UKIDSS GPS). Existing applications written in C++ [16] (in use, for example, in the interface to the SuperCOSMOS Halpha Survey [14]) will be employed to perform the difference imaging – this software employs adaptive kernel matching. Information concerning the newly created image products will be communicated back to the DBMS – ie. updates to the appropriate tables in the image database – via Perl DBI.

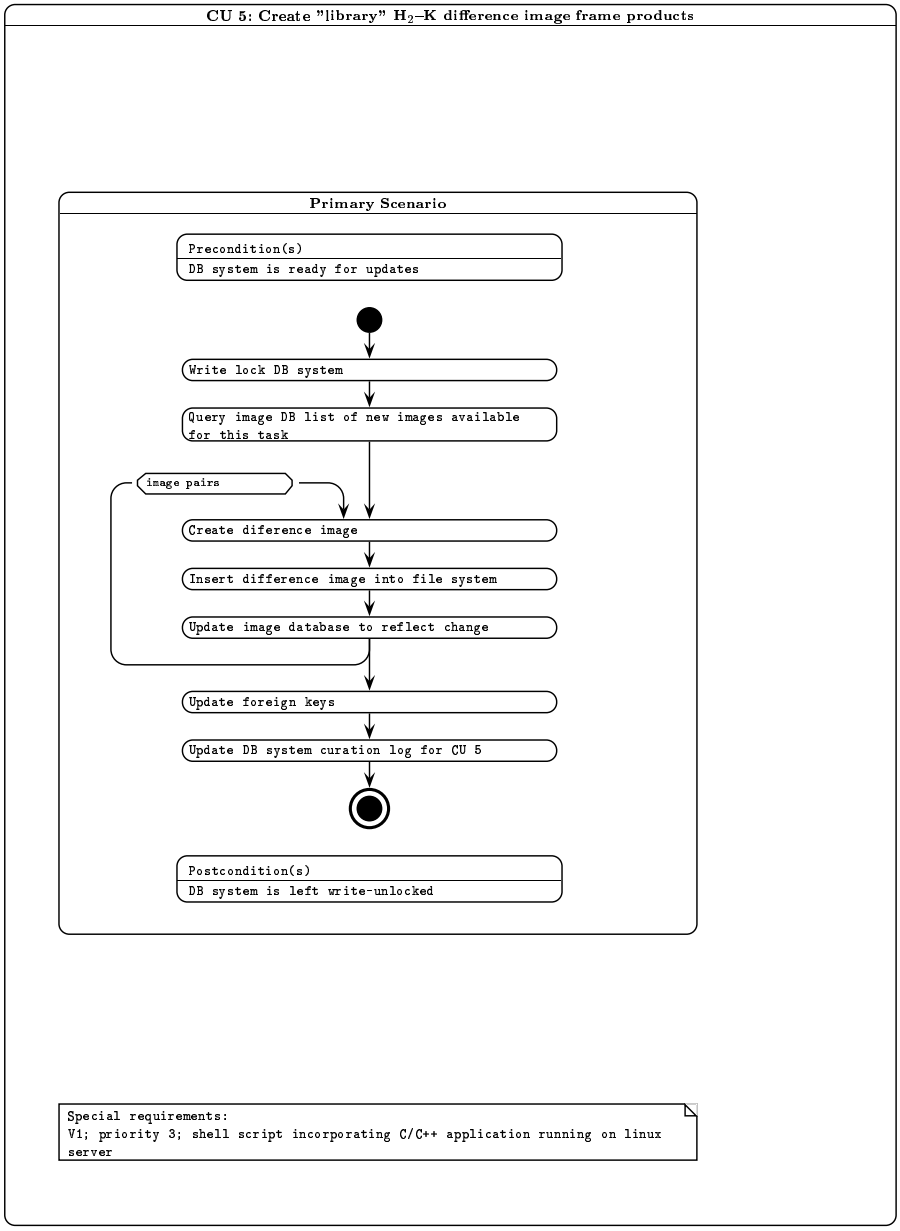

Figure 21: *Activity diagram for curation task CU5.*

9.6 CU6: Create spatial index attributes

```
CU6: Create spatial index attributes for all new records having celestial
     co-ordinates : Primary Scenarios

Precondition(s): DB system is ready for update

Flow of events:

1. Write-lock DB system

2. For every new occurence of explicit celestial co-ordinates

   a) Calculate spatial index attributes

   b) Update record index attributes

end

3. Create/recreate spatial indices for subsequent curation purposes

4. Update DB system curation log for CU6

Postcondition(s): DB system is left write-unlocked

Special requirement(s): V1; priority 1
```

Implementation details: as described in Section 7.1, V1.0 implementation will use the HTM scheme. This task will be implemented via SQL scripts making use of appropriate routines from SkyServer: `spHTM_lookup`, `spHTM_Cover`, `spHTM_To_String` etc. (for more details, see [15]).

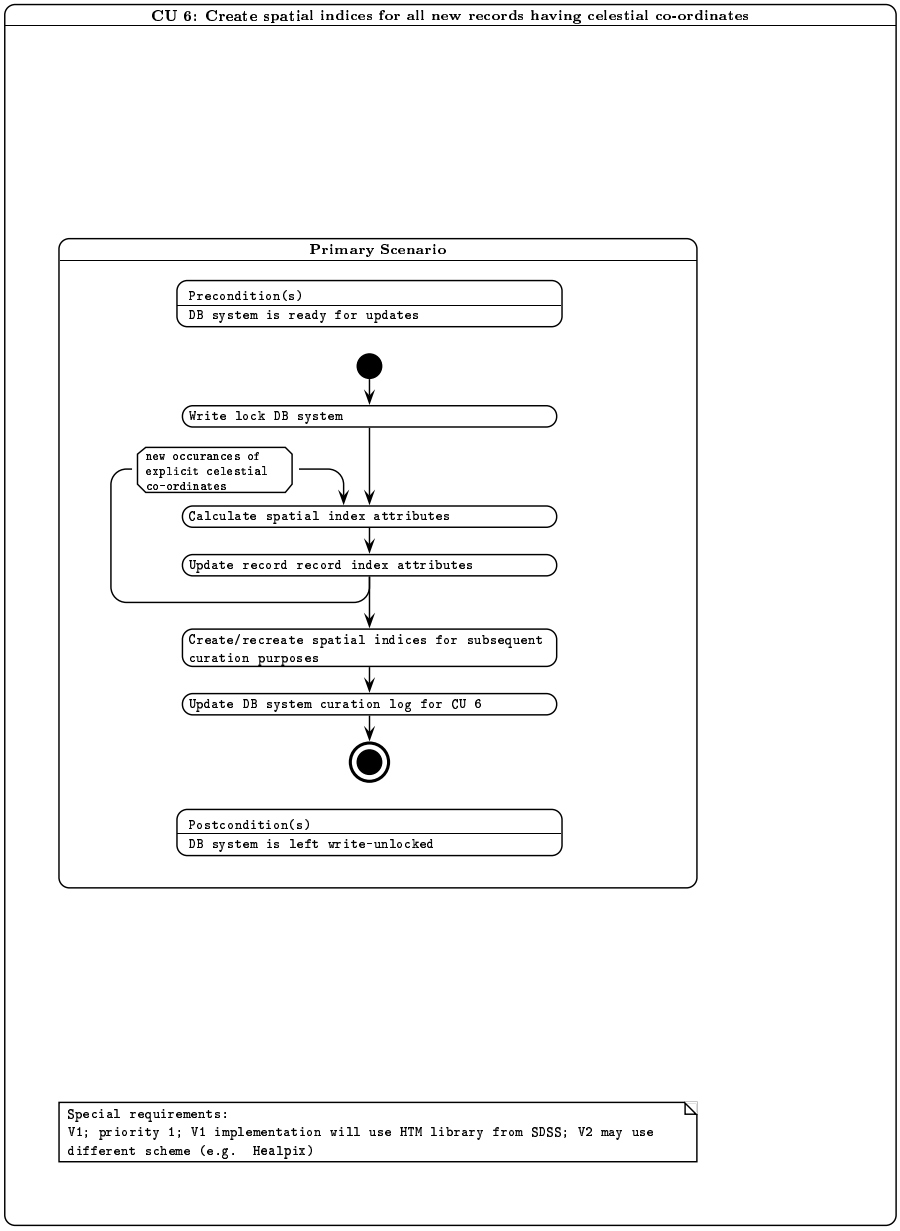

Figure 22: *Activity diagram for curation task CU6.*

9.7 CU7: Recalibrate photometry

```
CU7: Recalibrate photometry : Primary Scenarios

Precondition(s): Catalogue of photometric standards (primary, and/or
                 secondary) exists; DB system is read for updates

Flow of events:

1. Write-lock DB system

2. For every source detection list

   a) Create "join" with photometric standard table(s)

end

3. Calculate new, constrained photometric solution

4. For each stored frame

   a) Move current calibration to previous

   b) Update current photometric calibration coefficients

   c) Apply current photometric calibration where photometry explicitly
      stored

end

5. Add new photometric calibration version number

6. Update DB system curation log for CU7

Postcondition(s): DB system is left write-unlocked

Special requirement(s): V1; priority 1; implementation likely Java/C++
                        application running on catalogue server
```

The algorithm for the photometric solution will be similar to the 'global photometric solution' algorithm employed for 2MASS [10], but will be flexible in allowing for nightly extinction solution (or common extinction measurements across several nights) as opposed to global extinction solutions. Implementation of step 3 will require a portable coding language that can easily handle large, sparsely filled matrices; this means that implementation will likely be in C++ as opposed to Java. Once again, Perl scripting will be used to control the task workflow.

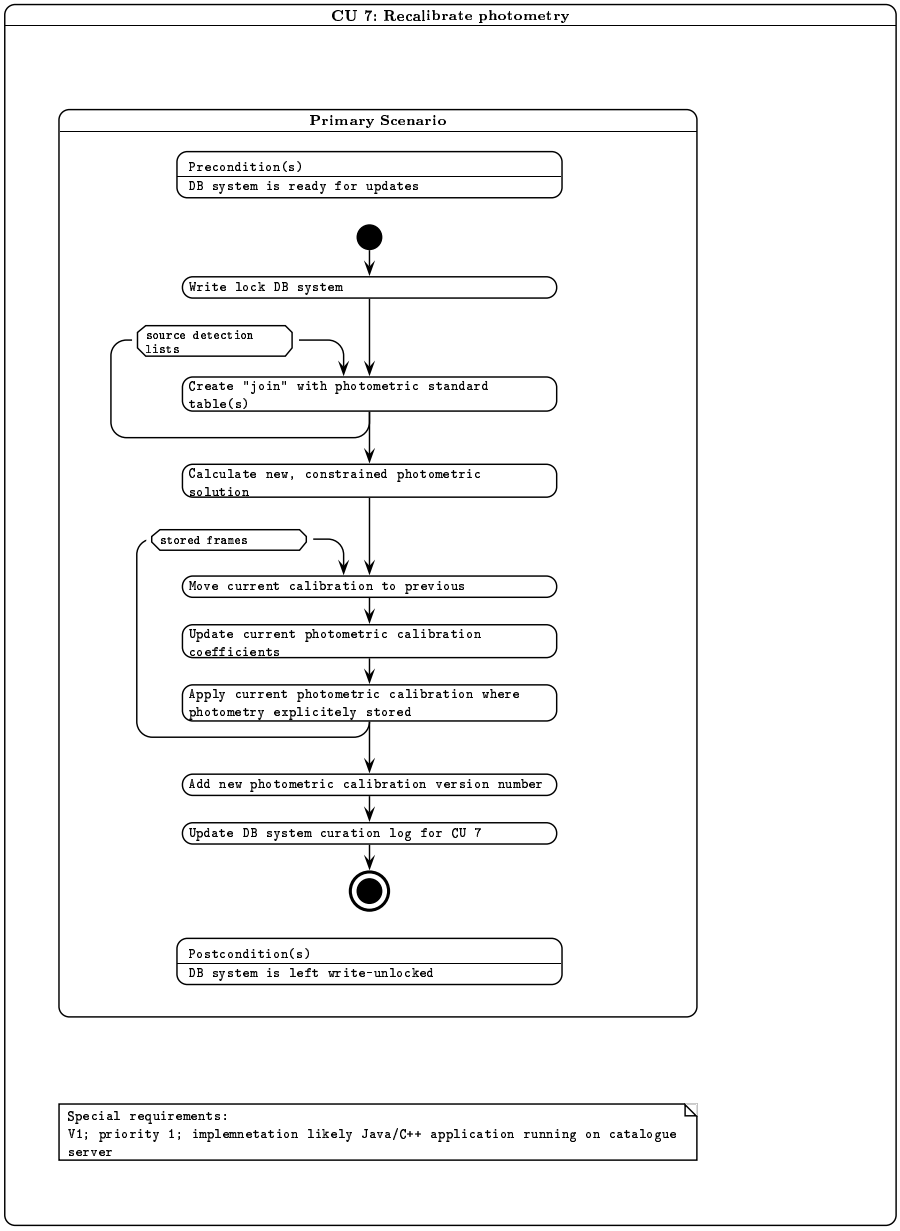

Figure 23: *Activity diagram for curation task CU7.*

9.8 CU8: Create/update merged source catalogues

```
CU8: Create/update merged source catalogues : Primary Scenarios

Precondition(s): DB system ready for update
                 Prescription for merged source catalogue is available
                 (eg. YJx2HK for UKIDSS LAS)

Flow of events:

1. Write-lock DB system

2. For every field stored for this programme

   a) For every image stored for this field

      i) If this image has not been listed in the merge event log, then

         Get detections for ALL passbands for this field;

         Produce "pair pointers" between each combination of colours
         taken two at a time;

         Produce merged source record using pointers and merge logic;

         Update existing merged source records for this field, including
         primary/secondary flag for duplicates (overlaps etc) and
         applying photometric calibration

         Update merge event log with all image IDs used, with flag of any
         new image(s)

      endif

   end

3. Update DB system curation log for CU8

Postcondition(s): DB system is left write-unlocked

Special requirement(s): V1; priority 2
```

Implementation details: the merging algorithm for a set of WFCAM detections in different passbands for the same field will follow methods developed for the SSS: the individual passband lists are sorted; pair pointers are produced between every combination of the passbands taken two at a time where the nearest object out to a maximum pairing radius of 3 arcsec is pointed to. An association algorithm then cycles through each image in each passband in turn, checking for consistent pointers forwards and backwards and keeping track of which detections have been written into the merged list so that each image is merged in only once. This general utility will be implemented such that it can be run for any programme for which a passband prescription set is specified in the *required filter* table (see Figure 13).

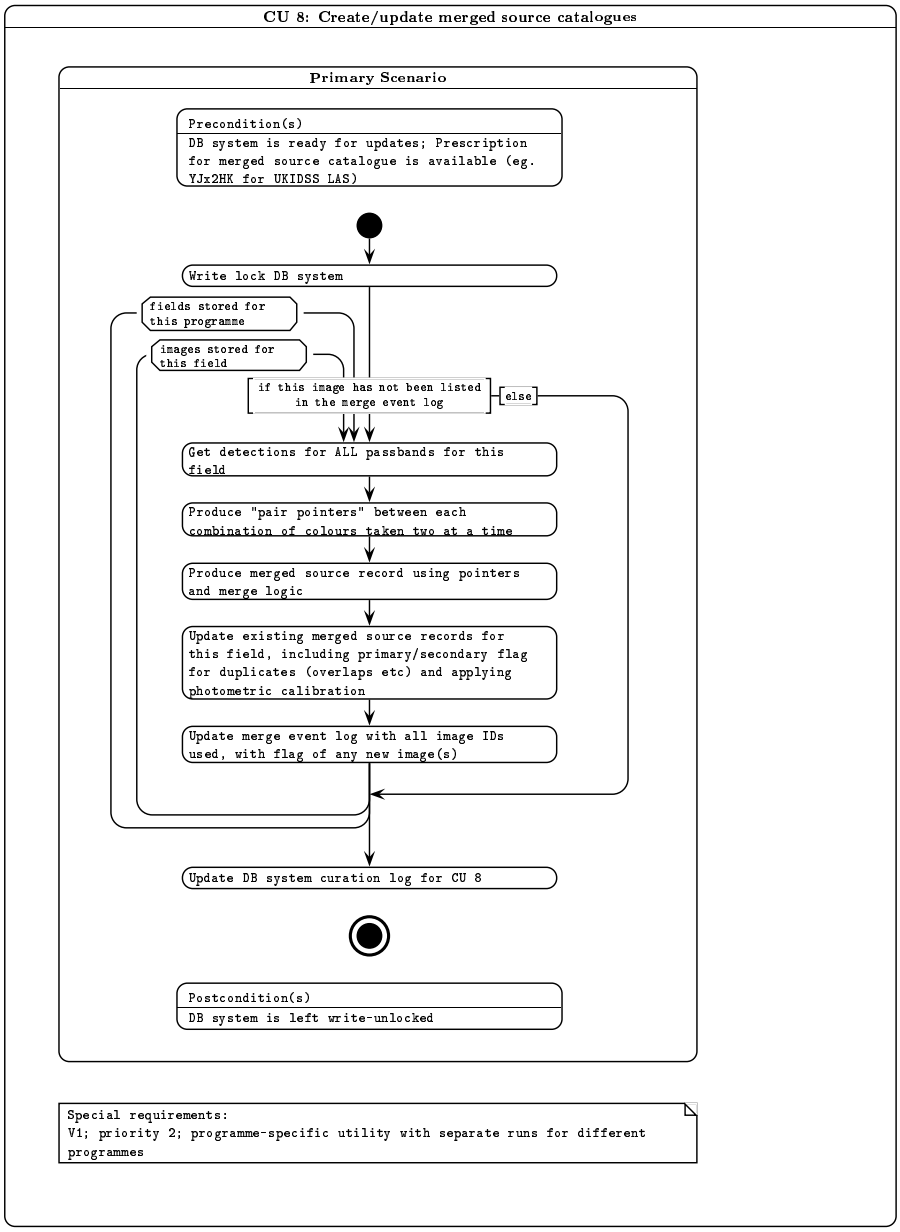

Figure 24: *Activity diagram for curation task CU8.*

9.9 CU9: Produce list measurements between WFCAM passbands

```
CU9: Produce list measurements between WFCAM passbands : Primary Scenarios

Precondition(s): DB system is ready for update

Flow of events:

1. Write-lock DB system

2. For every field logged in merge event with one or more new images

   a) Extract field master source list from merged source list

   b) Pass master source list plus image details to CASU list-driven
      photometry tool

   c) Translate FITS binary output into intermediate file (csv)

   d) Ingest intermediate file into list-driven remeasurements list

   e) Reset all new/old flags for images in this field as now "old"

   f) Delete intermediate file

end

3. Update DB system curation log for CU9

Postcondition(s): DB system left write-unlocked

Special requirement(s): V1; priority 3
```

Implementation details: this task will be implemented on the Linux server side. A Perl script will query the appropriate merged source table on the DBMS server side via Perl DBI to create a master driving source list for a given set of new images in the same field. The list, plus the location of the relevant images will be passed to the list–driven source re-measurement tool (a C++ application supplied by CASU). The FITS binary table output will then be translated and ingested using the same modules and methods as detailed in CU4.

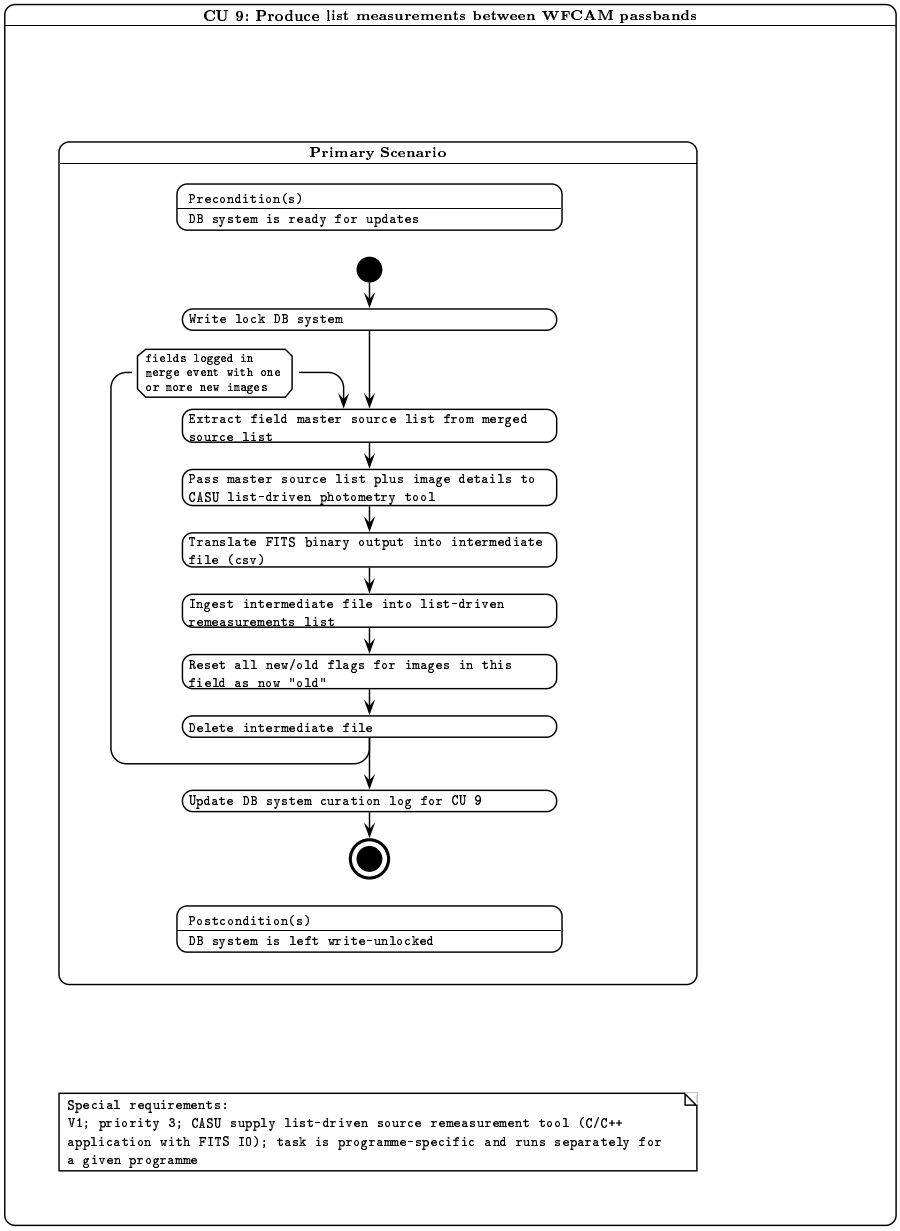

Figure 25: *Activity diagram for curation task CU9.*

9.10 CU10: Compute/update proper motions

```
CU10: Compute/update proper motions : Primary Scenarios

Precondition(s): Merged source catalogues have been created/updated;
                 DB system is ready for updates

Flow of events:

1. Write-lock DB system

2. For every programme field having two or more images separated by at
   least one year, where more images than have previously been used are
   available

   a) Obtain detection lists for each independent image (ie. no reruns)

   b) Associate detection lists (using existing merge information and
      proximity)

   c) Map out all position errors (small-scale) between measurement sets

   d) For each associated detection

      i) Compute full astrometric solution

      ii) Update astrometric attributes of corresponding merged source
          record

   end

end

3. Update DB system curation log for CU10

Postcondition(s): DB system is left write-unlocked

Special requirement(s): V2; priority 2
```

Implementation details: algorithmically, this task is analogous to existing implementations for photographic plates within the SSA, where relative positional shifts are determined after positional error mapping has removed all systematic errors on all scales down to a few arcminutes [7]. It is presently unclear as to the best coding implementation for this task – large vector and matrix manipulation is required, which clearly rules out SQL and possibly Java.

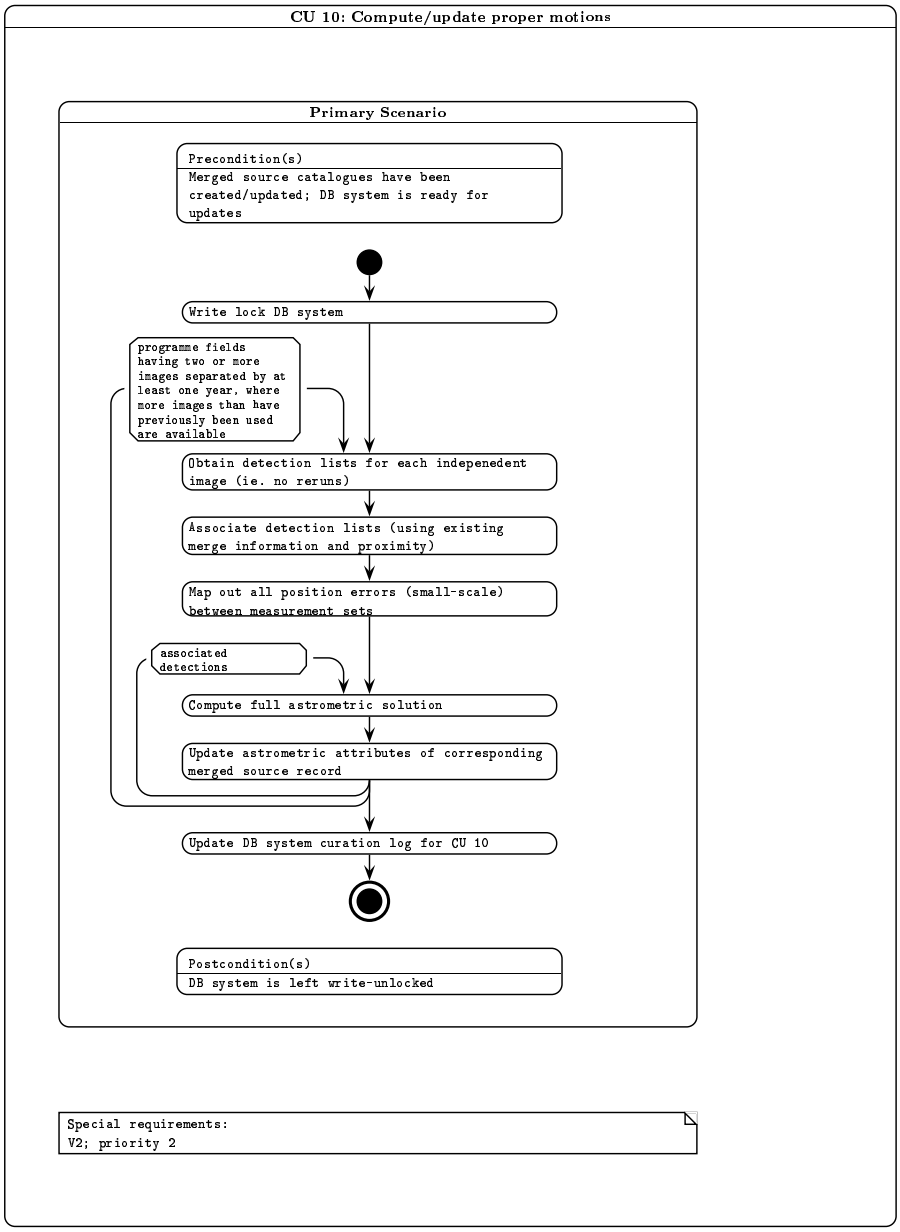

Figure 26: *Activity diagram for curation task CU10.*

9.11 CU11: Recalibrate astrometry

```
CU11: Recalibrate astrometry : Primary Scenarios

Precondition(s): DB system ready for update;
                 New astrometric reference catalogue exists and/or
                 low-level systematic errors are suspected to be
                 present

Flow of events:

1. Write-lock DB system

2. Analyse data set(s) for systematic errors, creating "mask(s)"

3. If systematic errors are present OR new reference catalogue is
   available, then

   For each stored detector/combi frame;

      i) If new reference catalogue exists, then

         Move current astrometric calibration to previous;

         Derive new astrometric calibration

      endif

      ii) Update current astrometric calibration details, including any
          systematic error correction mask

      iii) Apply current astrometric calibration, to all records having
           explicitly stored co-ordinates

      iv) Update spatial index attributes

   end

4. Add new astrometric calibration version number

5. Recreate spatial indices for any subsequent curation procedure.

6. Update DB system curation log for CU11

Postcondition(s): DB system left write-unlocked

Special requirement(s): V2; priority 3
```

Implementation details: this task will not be executed frequently. An intercative sequence of steps using a combination of SQL scripts, modules from CU6, and some application codes will be employed.

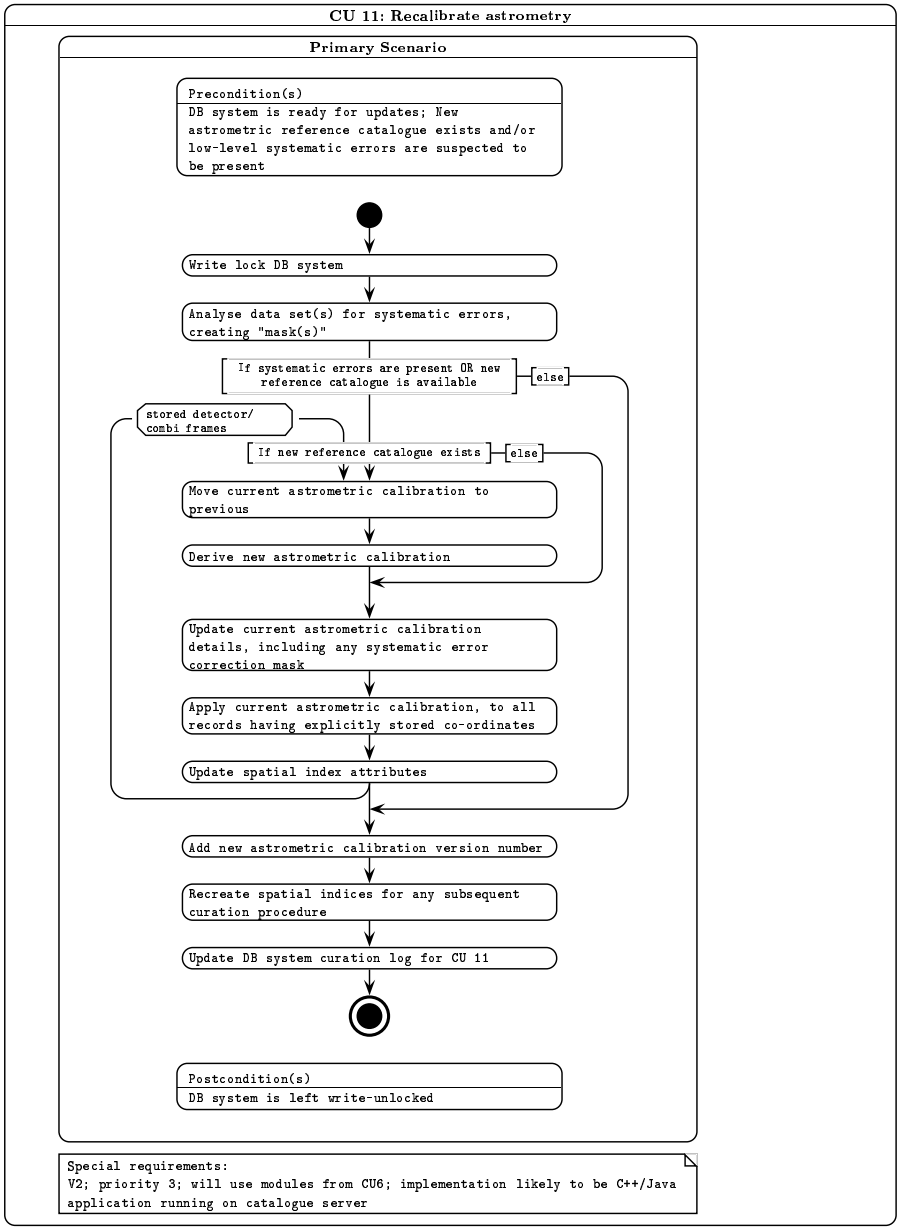

Figure 27: *Activity diagram for curation task CU11.*

9.12 CU12: Get external catalogues

```
CU12: Get publically released and/or consortium-supplied external
      catalogues : Primary Scenarios

Precondition(s): New data release is announced/communicated;
                 Stable catalogue data are available;
                 DB schema exists for the catalogue data;
                 DB system is ready for updates

Flow of events:

1. Write-lock DB system

2. If previous version of dataset is held in the archive, then

      Remove previous version

   endif

3. Obtain catalogue product(s) (eg. CD-ROM, DVD-ROM, or network copy)

4. If data not supplied as a DBMS database, then

      Translate catalogue product(s) into intermediate files

      Bulk load intermediate file(s) into DB system

      Remove intermediate files

   else

      Copy DB files into DB system

   endif

5. Update DB system curation log for CU12

Postcondition(s): DB system left write-unlocked

Special requirement(s): V1; priority 2
```

Implementation details: this relatively infrequent task will require an interactive sequence of procedures. External catalogues other than the SDSS will likely be supplied in a variety of formats, hence small bespoke translation software modules will be written where necessary, on a case–by–case basis, for implementation on the Linux mass–storage server. Bulk loading scripts for execution on the DBMS server will be written in SQL to ingest intermediate translated files.

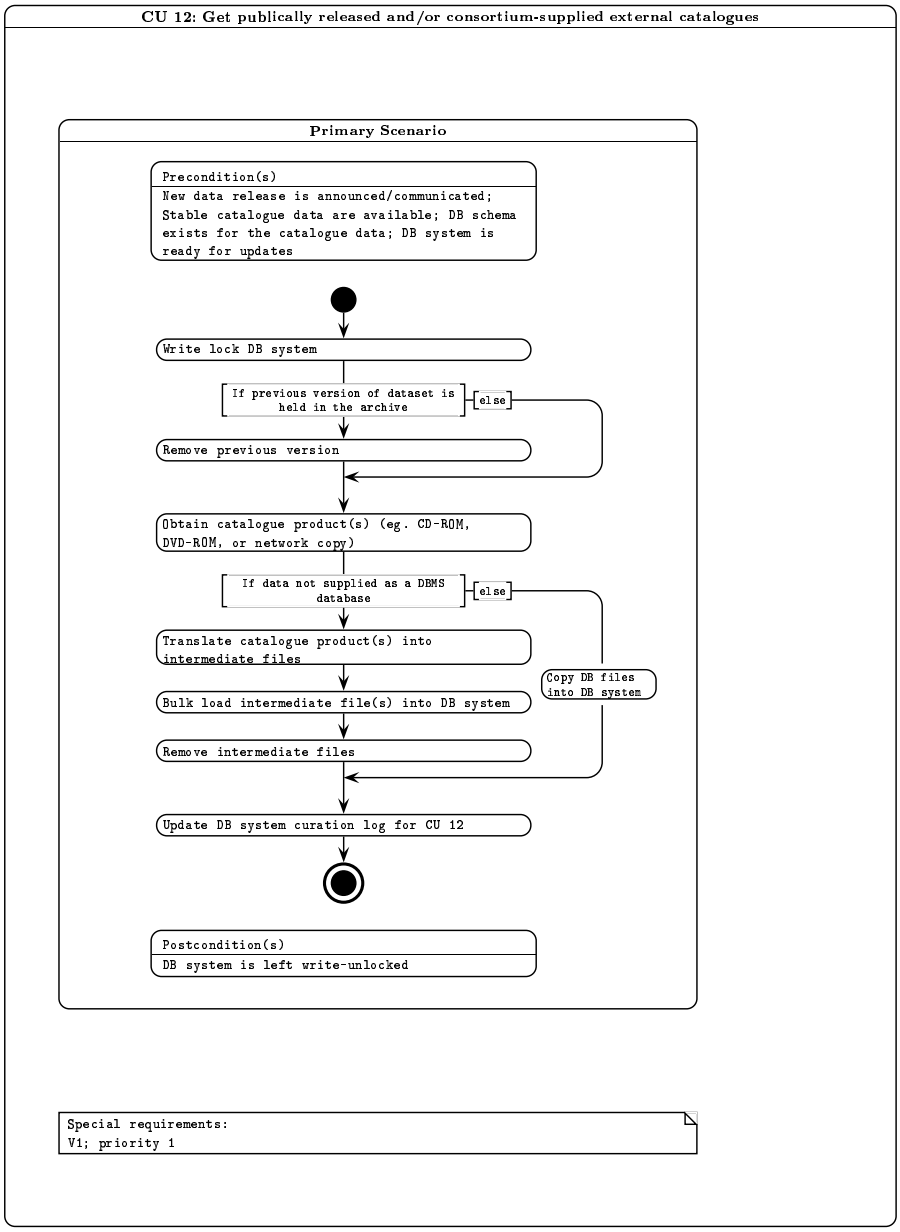

Figure 28: *Activity diagram for curation task CU12.*

9.13 CU13: Create library stacked /mosaiced image frame products

```
CU13: Create library stacked and/or mosaiced image frame
      products: Primary Scenarios

Precondition(s): DB system ready for update;
                 One or more new unstacked images exist for a given programme

Flow of events:

1. Write-lock DB system

2. For each passband required for this programme

   a) Query image DB for list of all independent images in the same passband
      of a given field taken for a given programme

   b) Pass images to CASU stacking/mosaicing tool(s)

   c) Insert output stacked/mosaiced image into the image file system

   d) Strip image product FITS keys to intermediate file (csv)

   e) Ingest details of new combiframe into combiframe image DB with new
   version number

   f) Remove intermediate file

end

3. Update curation log for relevant programme for CU13

Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 2
```

Implementation details: a Perl script running on the Linux server side will query the DBMS via Perl DBI to obtain lists of images to be manipulated. The image data and list obtained will then be passed to the appropriate combining tool (stacking or mosaicing tool supplied as C++ applications from CASU). The Perl script will then insert the resulting FITS file into the flat file store, having stripped the metadata to an intermediate file using the same module as in CU3. Finally, new image product details will be ingested using the same implementation as CU3 and CU4.

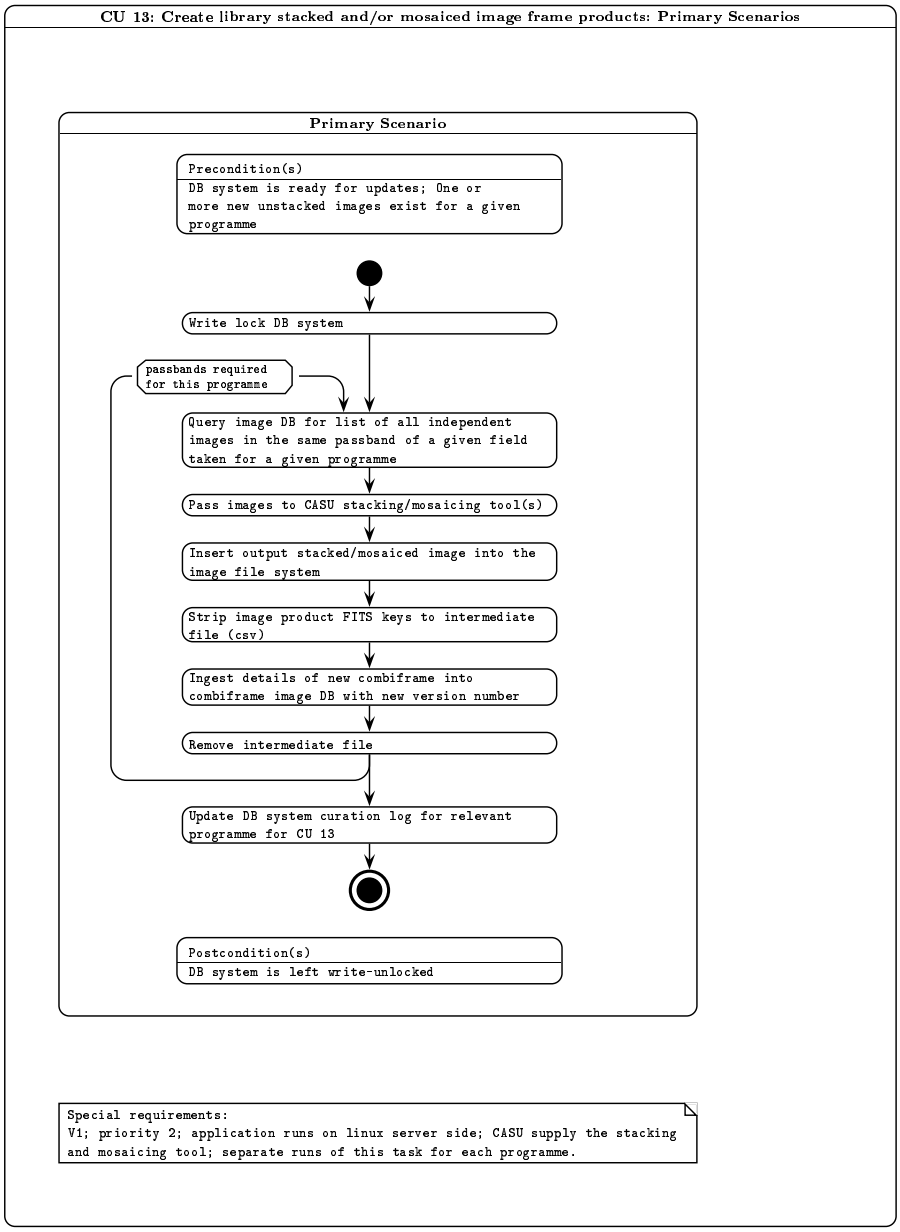

Figure 29: *Activity diagram for curation task CU13.*

9.14 CU14: Create standard source detection list from any new image frame product

```
CU14: Create standard source detection list from any new stacked/mosaiced
      image frame product: Primary Scenarios

Precondition(s): New stacked/mosaiced image exists;
                 DB system is ready for update

Flow of events:

1. Write-lock DB system

2. Pass image frame to CASU standard source extraction tool

3. Strip FITS keys in output FITS file to intermediate file; translate FITS
   binary table to intermediate file

4. Ingest new detections into combiframe detection list

5. Remove intermediate file(s)

6. Update curation log for relevant programme/passband for CU14

Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 2
```

Implementation details: a Perl script will be implemented on the Linux server side that invokes the CASU–supplied standard source extraction tool (a C++ application) for a specified image. Modules to strip the resulting image FITS keys and FITS binary table detection list (as for CU3 and CU4) will then execute, followed by DBMS ingest as detailed for CU4.

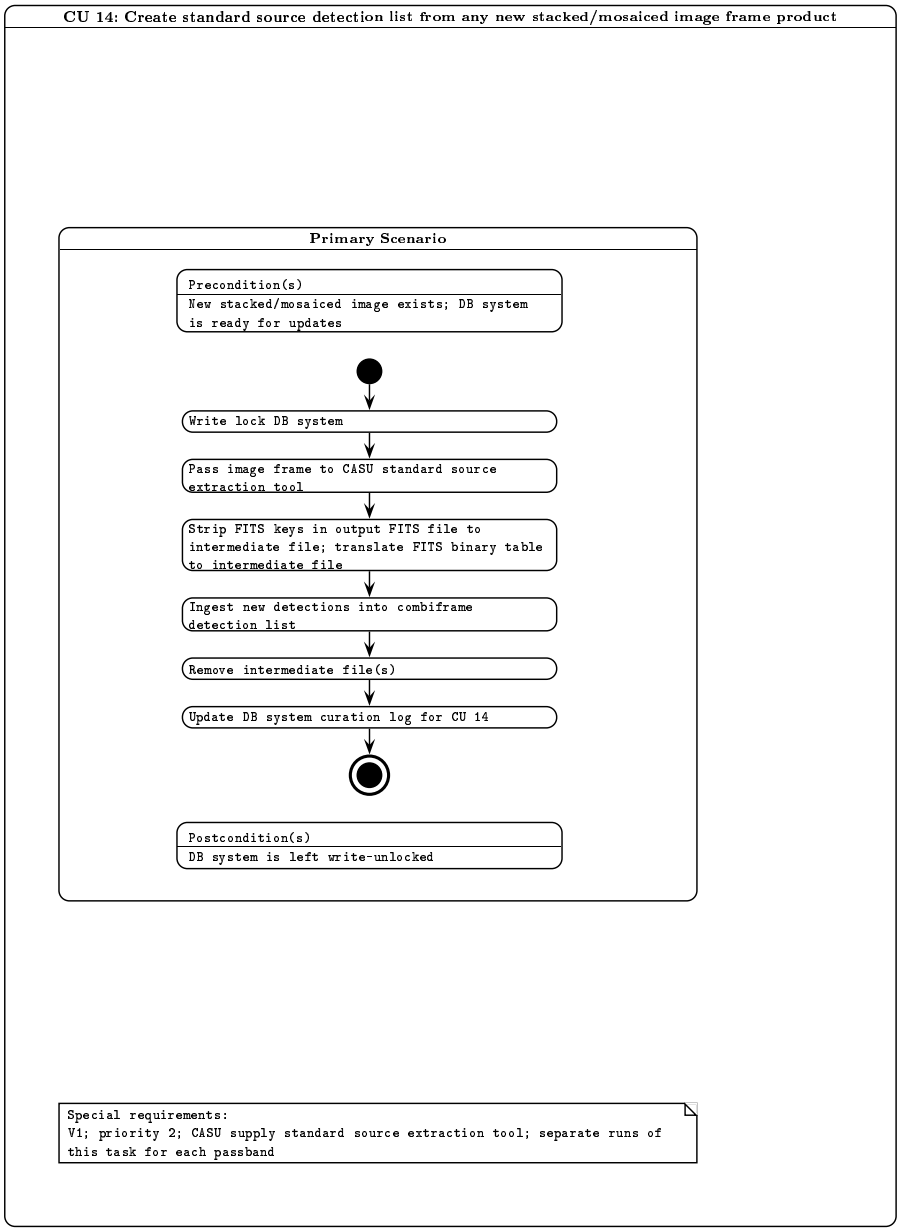

Figure 30: *Activity diagram for curation task CU14.*

9.15 CU15: Run periodic curation tasks CU6 to CU9

```
CU15: Run periodic curation tasks CU6 to CU9: Primary Scenarios

Precondition(s): DB system ready for update;
                 CU14 has been run more recently than the last run of this
                 task

Flow of events:

1. Run CU6

2. Run CU7

3. Run CU8

4. Run CU9

5. Update curation log for CU15

Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 2
```

Implementation details: this task is sufficiently infrequently run that interactive invocation of tasks CU6 through CU9 will cover implementation; those scripts/applications will be implemented in a generic way to enable them to be applicable to this task.

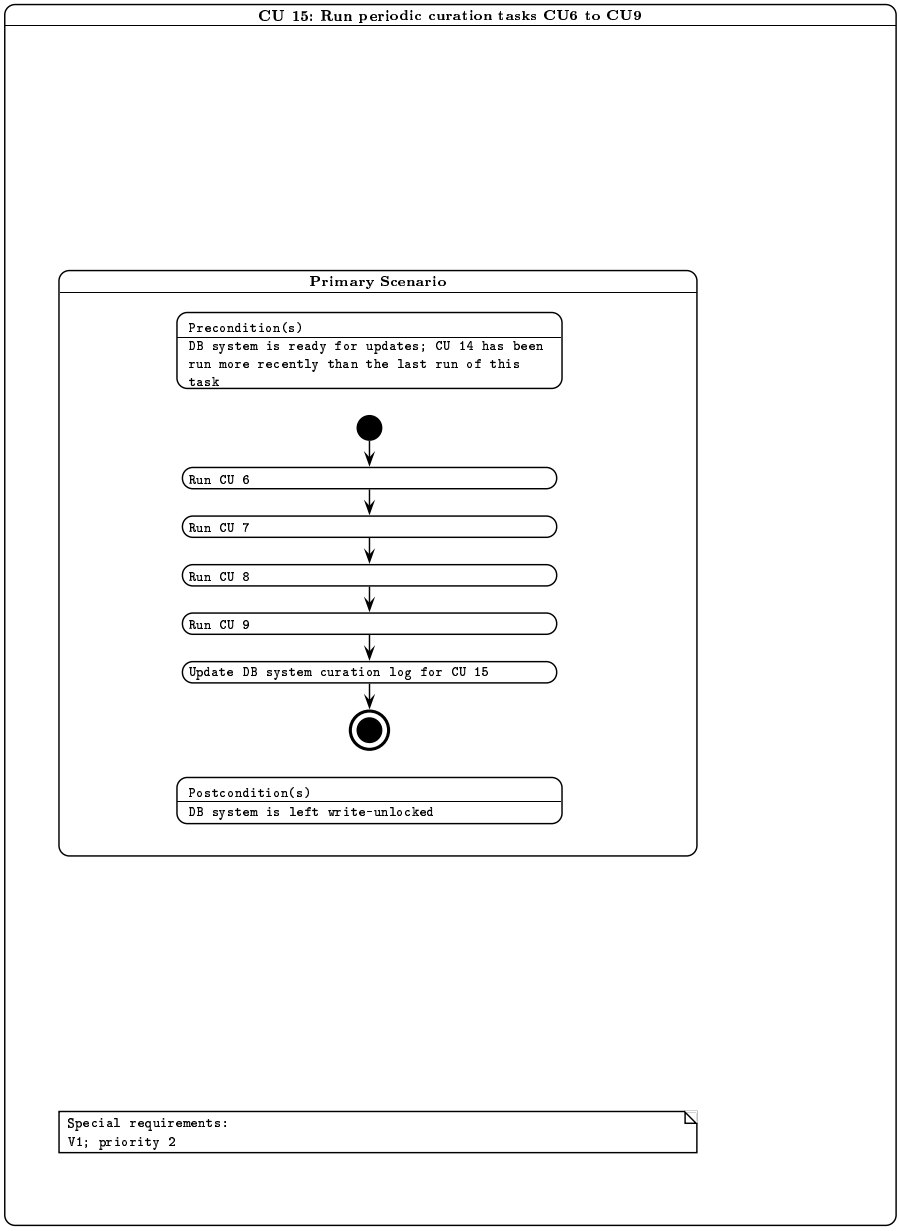

Figure 31: *Activity diagram for curation task CU15.*

9.16 CU16: Create default joins with external catalogues

```
CU16: Create default joins with external catalogues: Primary Scenarios

Precondition(s): DB system ready for update;
                 One or more new external catalogues exist

Flow of events:

1. Write-lock DB system

2. Create join table between required survey pair

3. Update curation log for CU16

Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 1
```

Implementation details: this task will be implemented in SQL script, employing the `fGetNearby` defined function from SkyServer to create a cross–neighbours table. The task will of course run on the DBMS load server side.

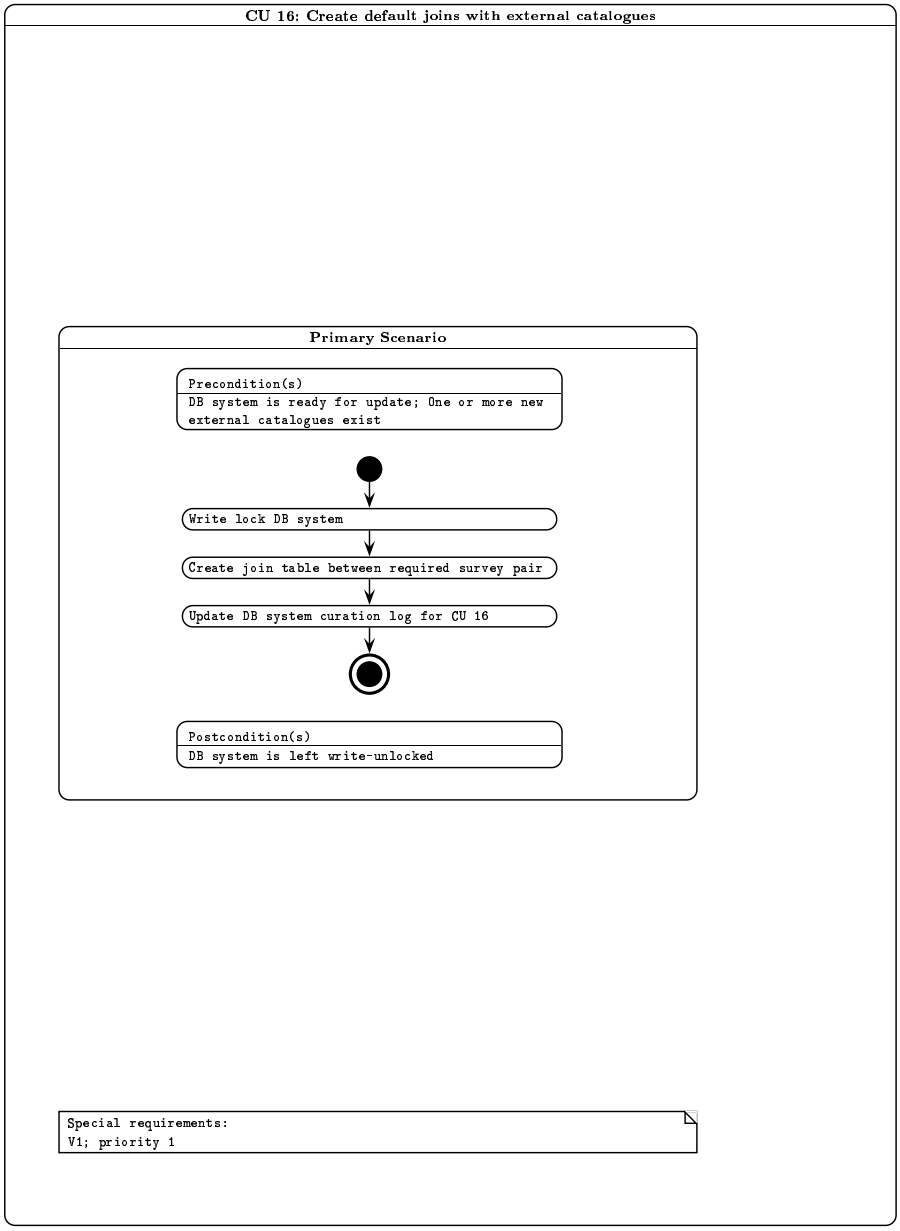

Figure 32: *Activity diagram for curation task CU16.*

9.17 CU17: Produce list–driven measurements between WFCAM/non–WFCAM data

```
CU17: Produce list-driven measurements between WFCAM detections and
      non-WFCAM imaging data: Primary Scenarios

Precondition(s):

Flow of events:

1. Write-lock DB system

2. For every image set in non-WFCAM pixel dataset

   a) Determine sky coverage of image

   b) Extract list of WFCAM sources that fall within this image, where no
      previous non-WFCAM measurements exist

   c) Pass this master driving list plus image details to CASU list-driven
      photometry tool

   d) Translate FITS binary output into intermediate file (csv)

   e) Ingest intermediate file into appropriate list-driven measurement
      table

   f) Delete intermediate file

end

3. Update curation log for CU17 for relevant programme

Postcondition(s): DB system left write-unlocked

Special requirements: V2; priority 2
```

Implementation details: this task will run as a Perl script on the Linux server side, where the list–driven source re-measurement tool will be supplied by CASU as a C++ application. Generally, implementation will follow that for CU9.

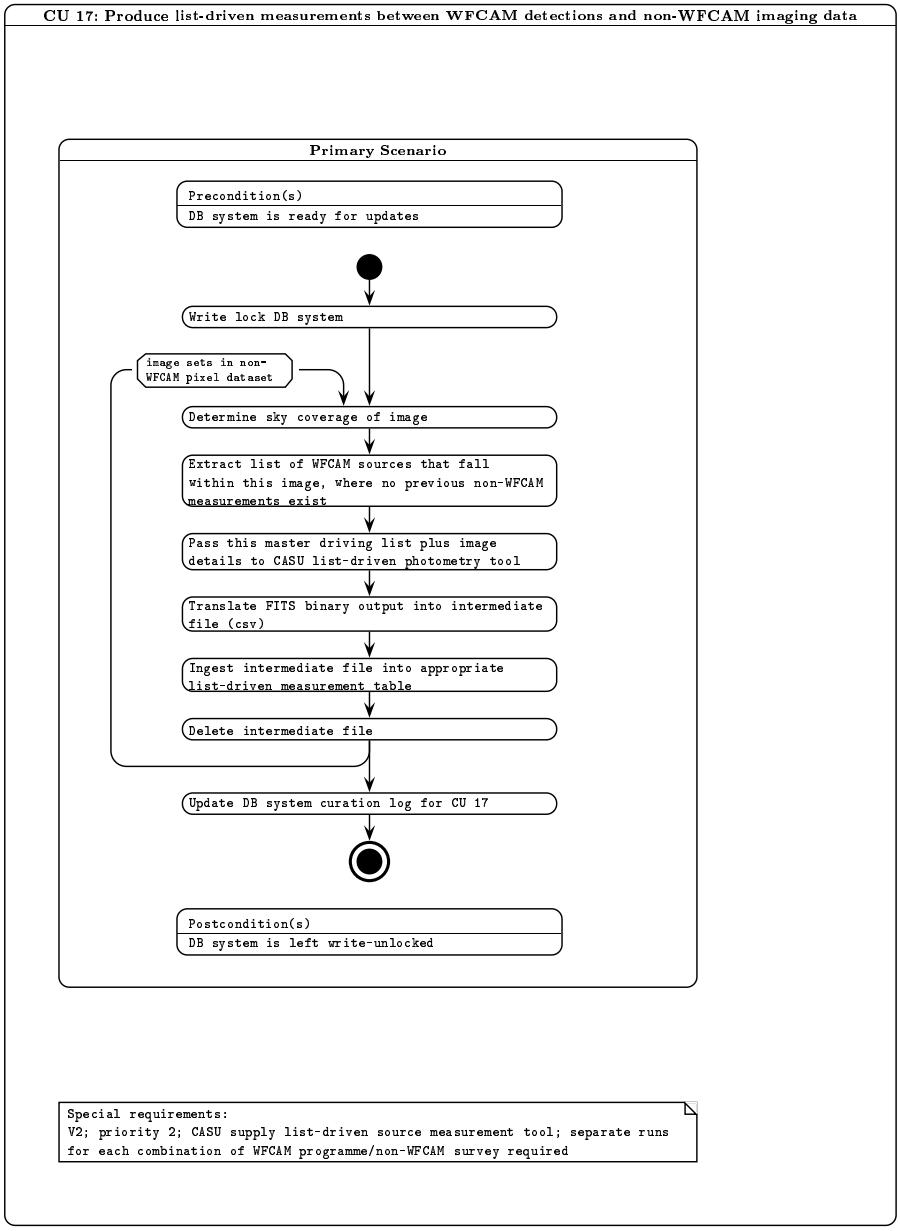

Figure 33: *Activity diagram for curation task CU17.*

9.18 CU18: Create/recreate table indices

```
CU18: Create/recreate table indices: Primary Scenarios

Precondition(s): DB system is ready for update

Flow of events:

1. Write-lock DB system

2. For every programme that has been updated since the last index update

   a) For every table for that programme

      i) Create/recreate required indices for this table

   end

   b) Update curation log for this programme

end


Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 1
```

Implementation details: this task will be implemented using SQL scripts on the DBMS server side.

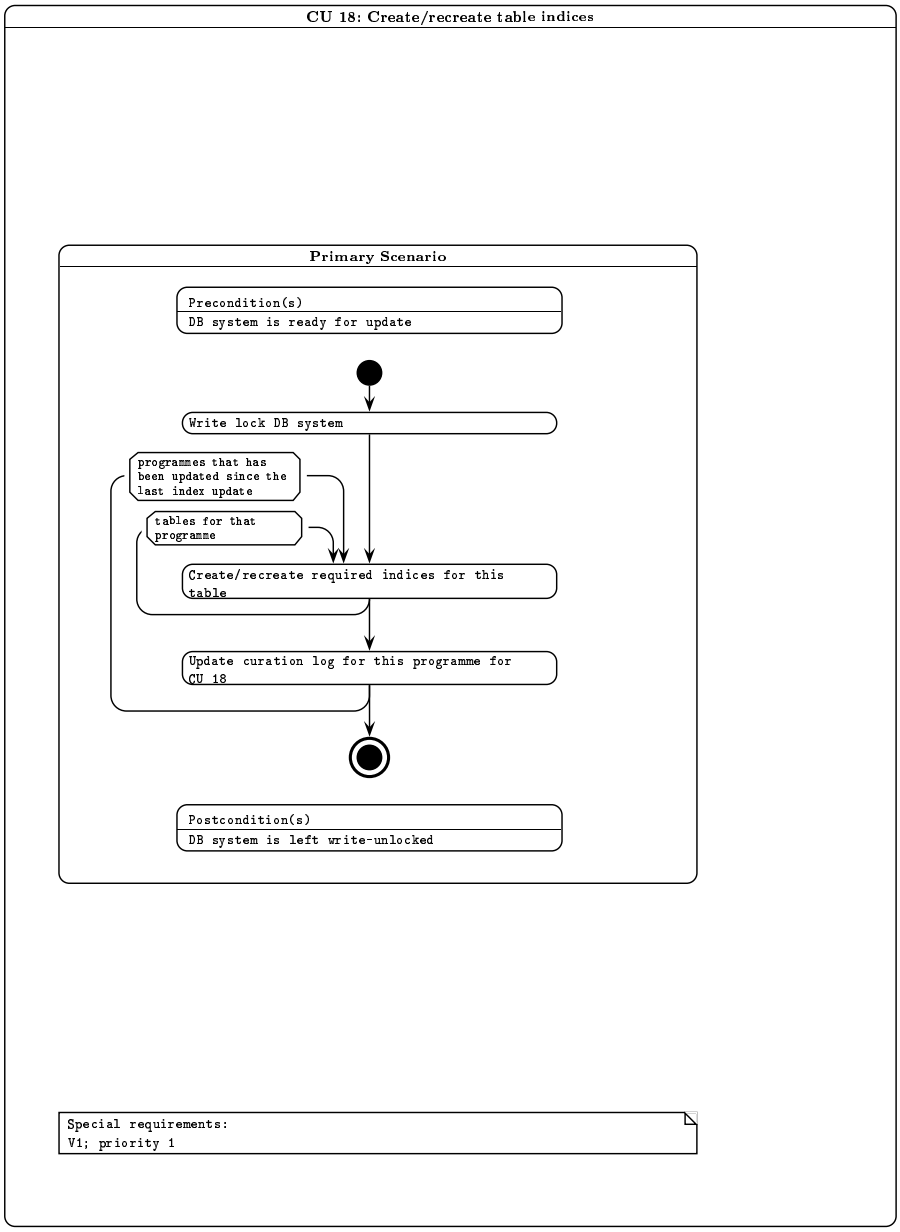

Figure 34: *Activity diagram for curation task CU18.*

9.19 CU19: Verify, freeze and backup

```
CU19: Verify, freeze and backup: Primary Scenarios

Precondition(s): DB system is ready for update

Flow of events:

1. Write-lock DB system

2. Verify that the latest required curation task has the most recent time
   stamp

3. Create world-readable subset of programme DB based on observation
   date, current release date and proprietorial period

4. Backup DB onto removable media

5. Copy DB out to online catalogue server

6. Update curation log for CU19 for relevant programme

Postcondition(s): DB system left write-unlocked

Special requirements: V1; priority 1
```

Implementation details: SQL script will be used to implement all steps except 4 and 5, which will be invoked interactively.

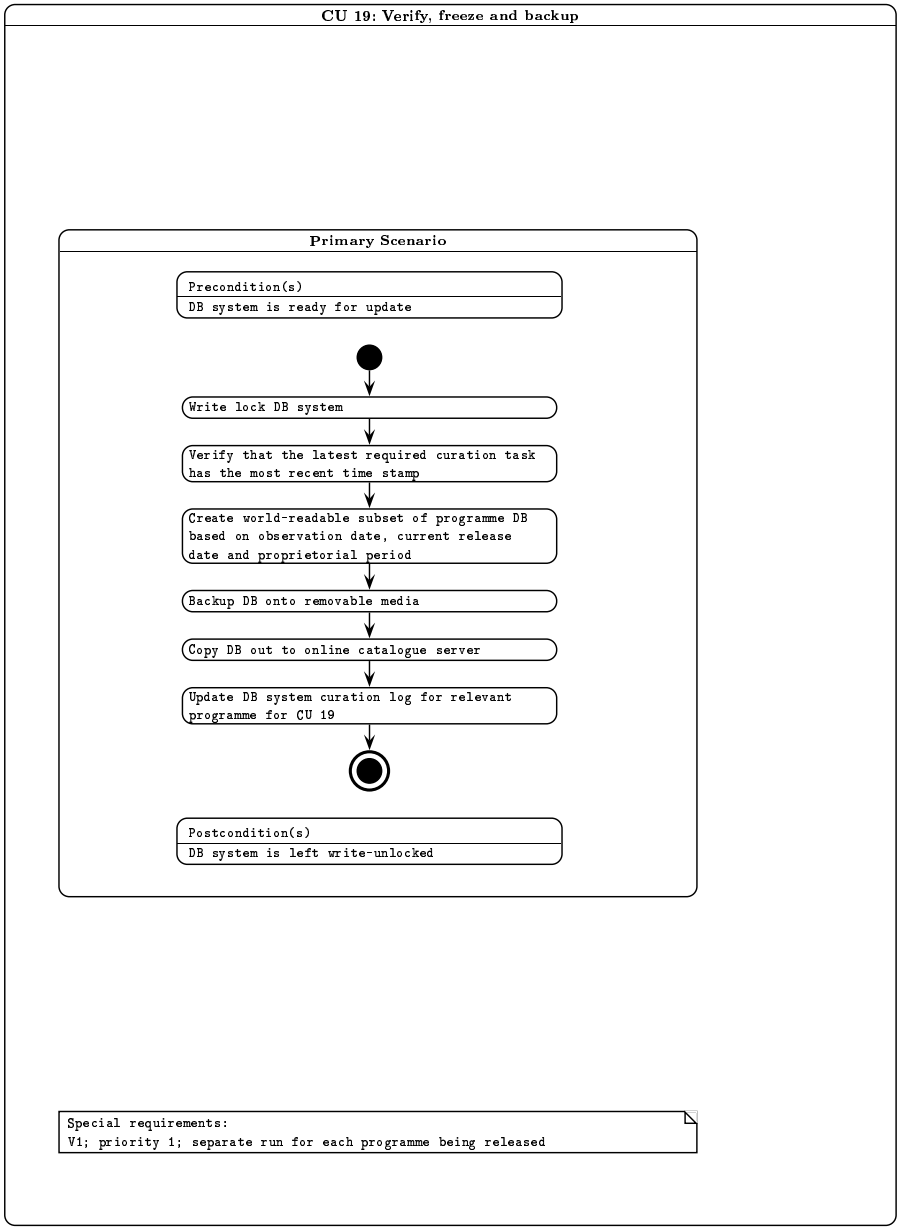

Figure 35: *Activity diagram for curation task CU19.*

9.20 CU20: Release – place online new DB product(s)

```
CU20: Release - place online new DB product(s): Primary Scenarios

Precondition(s): Public access to online server is disabled

Flow of events:

1. Remove the current "world" view

2. For every DB being newly released

   a) Remove previous corresponding existing DB

3. Recreate "world" view on all world survey subsets

4. Update release information in both online and offline curation log

5. Update curation log for CU20

Postcondition(s): Public access to online server enabled

Special requirements: V1; priority 1
```

Implementation details: SQL script will be used to implement this task. The script will run on the public–access server system.

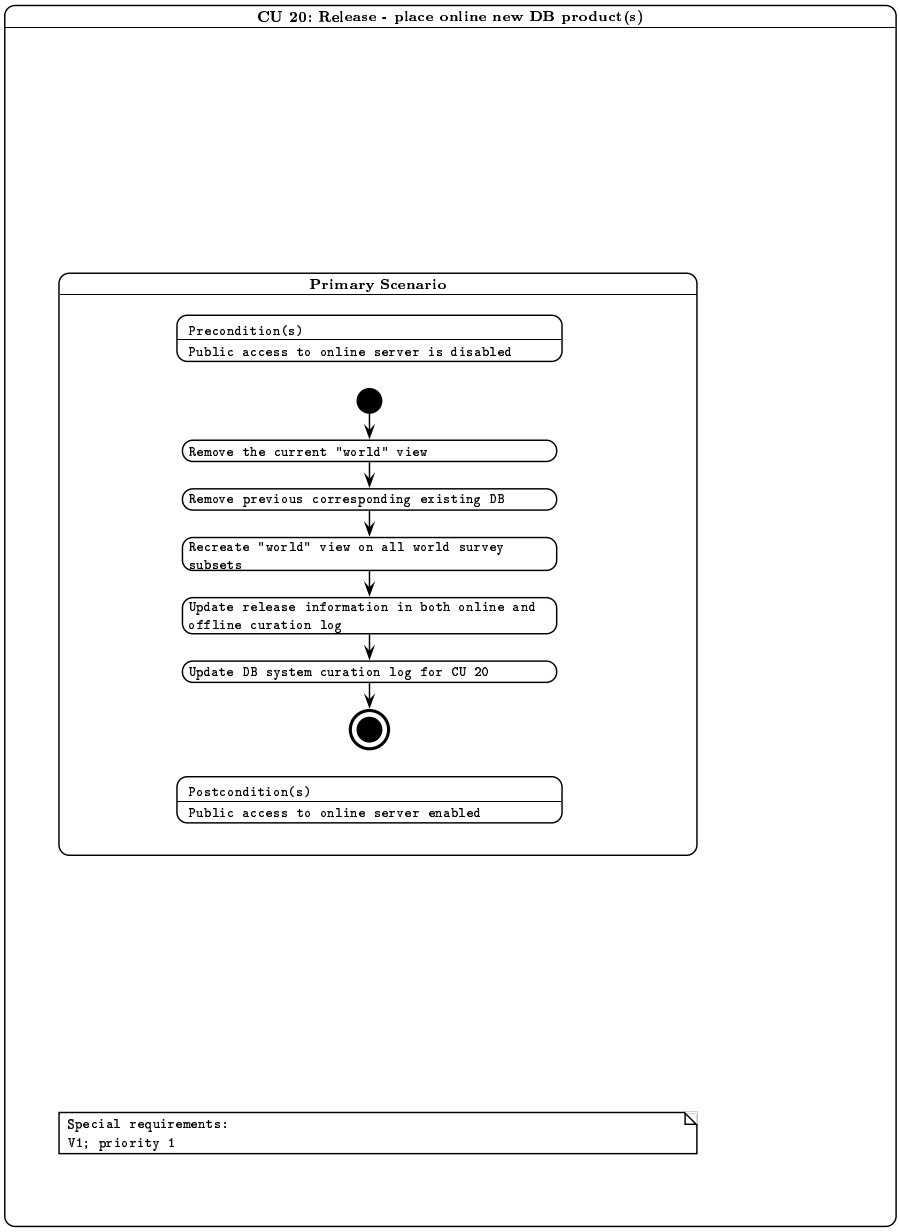

Figure 36: *Activity diagram for curation task CU20.*

References

[1] Hierarchical Triangular Mesh indexing; http://www.sdss.jhu.edu/htm/index.html

[2] The Sloan Digital Sky Survey at Johns Hopkins University; http://www.sdss.jhu.edu/

[3] Jim Gray, private communication.

[4] 20 Queries for the SuperCOSMOS Science Archive
http://www.roe.ac.uk/~nch/wfcam/misc/20queries.sql

[5] SQL Server database schema for the SuperCOSMOS Science Archive
http://www.roe.ac.uk/~nch/wfcam/misc/sss1412.sql

[6] Multi–threaded query agent and engine for a very large astronomical database, Thakar, A. et al., in Proc. ADASS IX, eds. Manset, Veillet & Crabtree, ASP Conf. Ser., 216, 231 (2000)

[7] The SuperCOSMOS Sky Survey, Paper III: Astrometry, Hambly et al., MNRAS, 326, 1315 (2001)

[8] A modified magnitude system that produces well–behaved magnitudes, colours and errors even for low signal–to–noise ratio measurements, Lupton et al., AJ, 118, 1406 (1999)

[9] Hierarchical Equal Area isoLatitude PIXelisation of the celestial sphere; http://www.eso.org/science/healpix/index.html

[10] A global photometric analysis of 2MASS calibration data, Nikolaev et al., AJ, 120, 3340 (2000)

[11] Achieving 1% photometric accuracy with the ESO wide field imager, Manfroid et al., ESO Messenger, June 2001, 16

[12] Archive scientist curation use cases for the WSA;
http://www.roe.ac.uk/~nch/wfcam/curate/curate.html

[13] The SuperCOSMOS Sky Survey;
http://www-wfau.roe.ac.uk/sss

[14] The SuperCOSMOS Halpha Survey; http://www-wfau.roe.ac.uk/sss/halpha

[15] Personal SkyServer; http://research.microsoft.com/~gray/SDSS/personal_skyserver.htm

[16] Real–time difference imaging analysis of MOA Galactic Bulge observations during 2000, Bond I. et al., MNRAS, 327, 868 (2001)

[17] WFCAM Science Archive Science Requirements Analysis Document;
http://www.roe.ac.uk/~nch/wfcam/srd/wsasrd/wsasrd.html

[18] WFCAM Science Archive interface control document;
http://www.roe.ac.uk/~nch/wfcam/VDF-WFA-WSA-004-I1/VDF-WFA-WSA-004-I1.html

[19] WFCAM Science Archive data flow document;
http://www.roe.ac.uk/~nch/wfcam/VDF-WFA-WSA-005-I1/VDF-WFA-WSA-005-I1.html

[20] WFCAM Science Archive hardware/OS/DBMS design document;
http://www.roe.ac.uk/~nch/wfcam/VDF-WFA-WSA-006-I1/VDF-WFA-WSA-006-I1.html

[21] WFCAM management and planning document;
http://www.roe.ac.uk/~nch/wfcam/VDF-WFA-WSA-002-I1/VDF-WFA-WSA-002-I1.html

[22] The UKIDSS Proposal; http://www.ukidss.org/sciencecase/sciencecase.html

[23] Example SQL schema scripts for the WFCAM Science Archive
http://www.roe.ac.uk/~nch/wfcam/misc/wsa.sql

10 APPENDICES

10.1 Curation use cases

For the purposes of progressing database design in the WSA project, this document describes a set of 'curation use cases'. The idea is to examine the WSA Science Requirements Document and Usages of the WFCAM Science Archive to define the required curation procedures, and hence to broadly outline the data entities in the WSA and their upkeep. In this context, 'curation' refers to the process of transfering and ingesting WFCAM data, generation of new data products from the standard pipeline data products, and management of those data products in the Science Archive.

The curation use cases are split into four broad categories, both for reasons of clarity and because requirements on various timescales give rise to distinct tasks:

- Daily curation use cases
- Periodic (weekly/monthly) curation use cases
- Occasional curation use cases
- Release time curation use cases

Also, the idea is to spread curation tasks in time as much as possible so as to spread the processing load on catalogue servers.

Because various survey data have various proprietory periods, it will be necessary to curate various databases having different access restrictions. We propose to curate online UKIDSS, open time and 'world' (ie. unrestricted access) databases, as well as incremental, offline versions of the same.

A database entity (table) will be required that keeps track of which curation tasks have been applied at any given time and to any given programme dataset.

10.1.1 Daily curation use cases

The following tasks occur on a daily basis. Pipeline processing will take place on a night–by–night basis and daily transfer and ingest is necessary to keep up with the end–to–end system data rate:

CU1: Obtain science data from CASU.

Pipeline processed data will be transfered from the pipeline processing centre (CASU) via the internet. This use case consists of

- transfering data
- verfication of transfered data - ensure integrity of data files obtained
- log transfer - keep track of which data have been transfered and at what time

Logged information from all curation tasks must be held in the database so that it is queryable by users. These logs will take the form of separate tables of data, or in this case, logged information held with the image metadata.

CU2: Create 'library' compressed image frame products

For speedy access to image data (eg. for finder chart purposes), compressed images are to be employed:

- JPEGs/GIFs

- 'science quality' lossy compressed frames, eg. H–compress or equivalent

CU3: Ingest details of transfered and library compressed images into archive DBMS, including their standard astrometric and photometric calibrations

Gbyte–sized FITS images will not be stored as BLOBs in the DBMS as this would result in heavy IO usage that would impact on more time–critical catalogue DBMS queries. Images will be stored as flat files; however their details (FITS header keywords, filenames, astrometric details, pipeline processing information, quality control information etc.) will be stored in the DBMS to track these files (from curation and use points of view).

CU4: Ingest single frame source detections into appropriate detection lists in the incremental (offline) archive DBMS.

All output from the standard pipeline source extraction algorithm needs to be stored in the archive.

10.1.2 Periodic (weekly/monthly) curation use cases

The following use cases occur periodically. The idea here is that the WSA usages require production of data products (catalogues) within the DBMS - it will be impossible merely to append data to existing entities (tables) within the DBMS since creating a science-usable survey dataset requires many non-linear operations (eg. pairing, indexing, astrometric and photometric recalibration). Exactly what timescale is practical will become clear with experience (eg. weekly, fortnightly or monthly) but the order in which the following use cases occur is clearly important (eg. spatial indexing must precede merging, which in turn must precede computation of proper motions, etc).

CU5: Create "library" H_2–K difference image frame products

After accumulation of a certain amount of image data it will be possible to update products resulting from combinatorial operations on individual image frames where this is not possible on a daily basis if image multiples are not guaranteed to be observed and processed together (generally the case).

CU6: Create spatial index attributes for all records having celestial co-ordinates

The simplest usages of the WSA (position/proximity searches) and also curation use cases such as pairing will be made much more efficient if the database entities are spatially indexed in some way (in it's simplest form, such indexing would take the form of sorting on one co-ordinate; the WSA will use a more sophisticated approach, eg. Hierarchical Triangular Mesh (HTM) indexing.

CU7: Recalibrate photometry

Full–blown photometric solutions over one or more photometric nights within an observing block will be undertaken. This will consist of:

- Associating detection records with photometric standards
- Solving for photometric coefficients (zeropoints, first– and second–order extinction coefficients, etc.)
- Updating calibration coefficients within the DB, and updating any records where colours are stored explicitly

CU8: Create/update merged source catalogues to the prescription available for a given survey from the appropriate detection list (CU4)

Standard CASU processing will not produce any merged catalogue products. Once the single passband detections are stored within the archive, a detection association algorithm will execute and produce

merged multi–colour, multi–epoch records appropriate to the available data within a given survey dataset. Separate merged source catalogue products will be required for different UKIDSS sub-surveys and open time programmes. The 'world' catalogues will be recreated at release time (CU19). Each merged catalogue dataset will have an associated table of logged information containing details of the merge run and what fields have been included to date.

CU9: Produce list–driven measurements between WFCAM passbands

Standard source detection will involve setting a threshold for detection; however in the context of data mining it may be important to have source extraction and detection limits (ie. photometric measurements) at positions and with apertures/profiles/deblending defined, for example, by detections across all bands. This philosophy follows the SDSS, where flux measurements at standard positions and in standard apertures are made in all bands when a detection is present in at least one band.

CU10: Compute/update proper motions and other multi–passband derived quantities

Other multi–colour attributes include, for example, extinction and dereddened apparent magnitudes which have been suggested within the UKIDSS GPS.

10.1.3 Occasional curation use cases

Occasional tasks are associated with newly available, externally produced data products from other survey programmes that are required to be held in the WSA for the purposes of joint querying to enable many of the science goals of UKIDSS, for example. Astrometric recalibration will also be undertaken occasionally.

CU11: Recalibrate image/detection astrometry

After data have accumulated for a sufficient time, low-level systematic errors in astrometry may become apparent, and it may be possible to remove these; furthermore, new astrometric reference catalogues may become available over the lifetime of the WSA, in which case astrometric recalibration is in order.

CU12: Get publicly released and/or consortium–supplied (eg. complementary optical) external catalogues

It will be necessary to update the locally stored (but externally produced) survey products (eg. SDSS, 2MASS, etc.) where new releases of those products have been made; the UKIDSS consortium also has a programme of complementary optical imaging for the purposes of combined optical/IR science.

10.1.4 Release curation use cases

Proprietory PI programmes should be released to the proprietors asap; updates to UKIDSS surveys will naturally occur on a timescale dictated by blocks of WFCAM on–telescope periods.

CU13: Create "library" stacked and/or mosaiced image frame products

After accumulation of a certain amount of image data it will be possible to update products resulting from combinatorial operations on individual image frames.

CU14: Create standard source detection list from any new "library" frame product(s)

For example, if a new UDS stack has been made, then standard source detection should be run on the resulting image data. Curation use cases CU6 to CU9 would then apply:

CU15: run periodic curation tasks 6 though 9

– for any newly created stacked/mosaiced image product.

CU16: Create default joins with external catalogues

For the purposes of cross–IDs/neighbours for rapid cross-querying.

CU17: Produce list-driven measurements between WFCAM detections and non–WFCAM image data, where possible

Again, following CU9, and with usages such as U1 in mind. That example concerns the UKIDSS LAS and SDSS UGRIZ imaging data, where IR sources with upper limits in IZ are sought for candidate very cool objects.

CU18: Create/recreate table indices

Within a given table, an index will be created on a combination of commonly used attributes so that at query time, the query optimizer will make use of these indices to greatly enhance performance.

CU19: Verify, 'freeze', and backup

Verification takes the form of examining the curation log to check that no further curation is needed for any given programme/survey dataset. This curation task will create a 'world' readable subset of the given programme dataset based on the proprietory period of the observations (tracked through the database) and the current release date. Prepared survey DBs should be fixed and backed up for security

CU20: Release – place new DB products online

This task is the final step: any newly created database products will be placed on a publicly accessible catalogue server. At the same time, a 'world'–readable 'view' of the programme subsets will be created to present a single, logical database of all WFCAM observations having unrestricted access at that date.

10.2 The SuperCOSMOS Science Archive database

The SuperCOSMOS Science Archive (SSA) has been developed as a prototype Tbyte–scale archive to gain experience prior to arrival of large amounts of WFCAM data. The SSA is an implementation of the SuperCOSMOS Sky Survey (SSS; [13]) in Windows/SQL Server as opposed to the flat–file system accessible via the existing web interface. The SSA has many similarities to the WSA:

- spatial indexing is required for speedy access;
- many photometric, astrometric and morphological descriptors per detected source;
- multi–colour, multi–epoch merged source catalogue results from multi–passband, multi–epoch imaging;

but there are some significant differences – eg. the SSS is limited to a small number of all–sky 'atlas' surveys (eg. the ESO/SRC J/EJ southern hemisphere survey) as opposed to a more complex 'wedding cake' arrangement of science–driven programmes (eg. UKIDSS [22]).

The development of the SSS has closely followed the science archive developments for the SDSS [2]. We followed the '20 queries' approach for the SSA [4], including SDSS/SSA cross–queries since this is a key requirement on the WSA (see the SRAD).

In designing the SSA schema, we started with the flat–file formats in the existing public archive, and reformmatted this in line with the relational model (Figure 2 in Section 5.2) and included all housekeeping and other metadata to create a self–contained database that preserves the provenance of all detected sources right back to the original source photographic plates and the Schmidt surveys

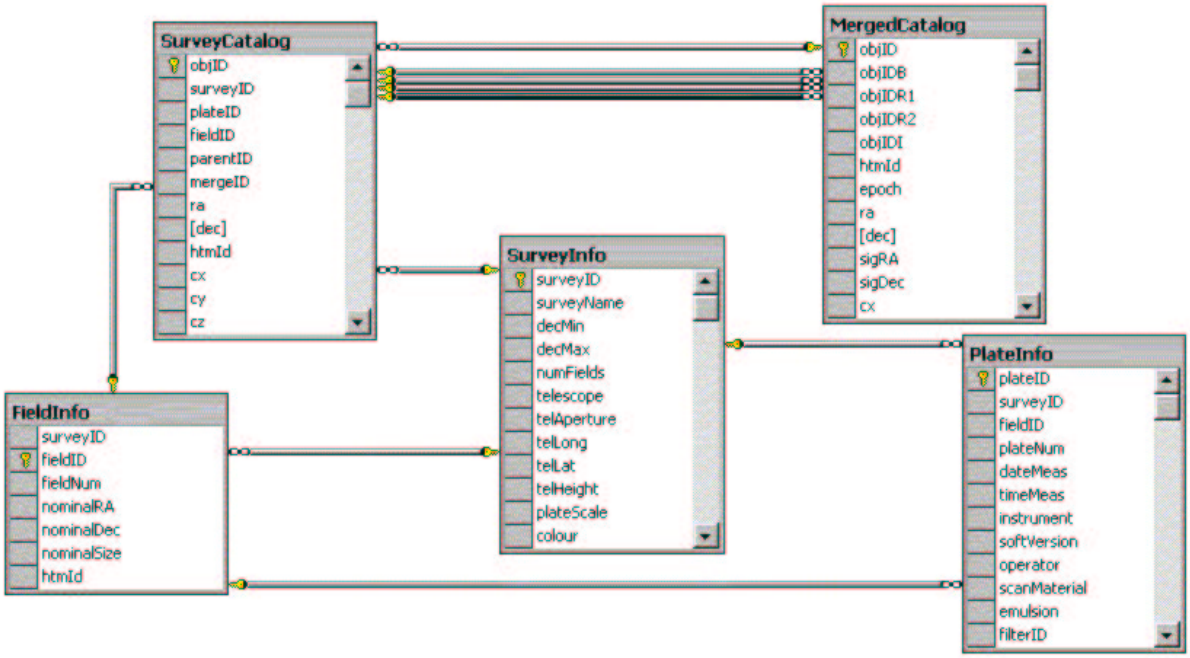

Figure 37: *SQL Server database diagram of the five tables that comprise the SuperCOSMOS Science Archive. Foreign key references are shown as connections between the relevant tables.*

to which they belong. Figure 37 shows an SQL Server 'database diagram' of the SSA created from the schema file [5].

So far in the SSA prototyping programme we have loaded SSS data corresponding to the SDSS–EDR regions and have exercised the database by testing our 20 queries (including SDSS/SSA joint queries). Once hardware money becomes available and we have settled on our final catalogue server design (see the HDD, AD05) we will load in the full SSA dataset to yield a Tbyte–scale archive that is intended to fulfill the following purposes:

- to gain experience in optimisation of database curation operations (eg. different indexing schemes and disk volume arrangement) for a Tbyte–scale catalogue dataset, with direct relevance to WSA curation;
- to provide a test–bed for development of WSA software – ie. curation and user interface software.

Of course, in prototyping the WSA via the SSA, we will create a public service that is useful in itself: users will be able to access SSS data in a faster and more flexible way than is currently possible.

10.3 Example SQL script files for multiframe and associated calibration table schemas

```
--
-- MultiframeSchema.sql
--
-- Database schema file for multiframe images, programme and filter
-- data in the WFCAM Science Archive. Contains sql scripts to create
-- the relevant tables.
--
```

```
-- Original author: Nigel Hambly, WFAU, IfA, University of Edinburgh
--
-- Revision history:
-- Login name of the user who checked in this revision:  $Author$
-- Date and time (UTC) when the revision was checked in: $Date$
-- Login name of the user who locked the revision:       $Locker$
-- CVS revision number:                                  $Revision$
--

CREATE TABLE Programme(
--
--  This table contains details of the observation programmes undertaken
--  with WFCAM and for which observations are stored and curated in the WSA.
--
--        The unique ID number (assigned internally within the WSA)
--        identifying all WFCAM observation programmes is as follows:
--
--          programmeID = 0 : not a programme: used for calibration/confidence
--                            frames common to many programmes
--                      1 : Commissioning data
--                    101 : UKIDSS Large Area Survey (LAS)
--                    102 :   "    Galactic Plane Survey (GPS)
--                    103 :   "    Galactic Clusters Survey (GCS)
--                    104 :   "    Deep Extragalactic Survey (DXS)
--                    105 :   "    Ultra-Deep Survey (UDS)
--                   1000+: Open Time programmes (PI)
--                  10000+: Service programmes;
--
--  Required constraints: primary key is (programmeID)
--
programmeID   int not null, -- UID of the archived programme coded as above
title         varchar(32) not null, -- a short title for the programme,
--            eg. "UKIDSS Large Area Survey"
description   varchar(256) not null, -- a concise description of the programme
reference     varchar(256) not null, -- a reference for the programme,
--            eg. "http://www.ukidss.org/surveys/surveys.html"
propPeriod    real not null -- the proprietory period for any data taken
--            for this programme in years, eg. 1.0 for open time.
       )

CREATE TABLE Filter(
--
-- This table stores details of the WFCAM filters.
--
--        The unique ID number (assigned internally within the WSA)
--        identifying all the WFCAM filters is as follows:
--
--          filterID  =  1 : Y filter
--                       2 : J   "
--                       3 : H   "
--                       4 : K   "
```

```
--                      5 : H2  "
--
--  Required constraints: primary key is (filterID)
--
filterID     tinyint not null, -- UID of filter, defined as above
name         varchar(16) not null, -- the name of the filter,
--           eg. "MKO J", "UKIDSS Y", "Narrow band H2" etc;
description  varchar(256) not null -- a concise description of the filter
--
--     ??
--     wave01              real not null, -- first wavelength value in microns
--     resp01              real not null, -- normalised response at wave01
--       .
--       .
--       .
--     wave50              real not null, -- first wavelength value in microns
--     resp50              real not null, -- normalised response at wave01
--     ??
       )

CREATE TABLE DifferenceImage(
--
-- This table stores details of all default difference images stored in the
-- archive.
--
--  Required constraints: primary key is (diffimID)
--                        masterID references Multiframe(multiframeID)
--                        secondID references Multiframe(multiframeID)
--                        astromID references Multiframe(multiframeID)
--
diffimID     int not null, -- the unique ID number (assigned sequentially as
--           the images are made within the WSA) identifying all stored
--           difference images
masterID     bigint not null, -- the ID of the "master" multiframe
secondID     bigint not null, -- the ID of the subtracted multiframe
astromID     bigint not null, -- the ID of the superframe for the astrometry
filename     varchar(256) not null, -- the Linux flat filename for the image
--           eg. server:/path/filename.fit
creationDate varchar(32) not null, -- the date that this difference image
--           was created in the format YYYY-MM-DDThh.mm.ss;
swVersion    real not null -- version number of the difference image
--           software used to create the image
             )

CREATE TABLE Multiframe(
--
-- This table stores details of all multiframes stored in the archive.
--
--  Required constraints: primary key is (multiframeID)
--                        filterID references Filter(filterID)
--                        programmeID references Programme(programmeID)
```

```
--                          (programmeID,fieldNum) references
--                                      SurveyField(programmeID,fieldNum)
--                          darkID references DarkFrame(darkID)
--                          confID references ConfFrame(confID)
--                          flatID references FlatFrame(flatID)
--                          frinID references FrinFrame(frinID)
--                          skyID references SkyFrame(skyID)
--                          diffimID references DifferenceImage(diffimID)
--                          (JulianDay,filterID) references
--                                   MultiframeSetUp(JulianDay,filterID)
--
--
multiframeID   bigint not null, -- the UID of the multiframe made up from
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
programmeID    int not null, -- WSA assigned programme UID
diffimID       int not null, -- WSA assigned difference image UID
JulianDay      int not null, -- the Julian Day number of the UKIRT night
--
--     primary HDU FITS keys go here:
--
OBSERVER       varchar(64) not null, -- Observers names
USERID         varchar(64) not null, -- Userid logged in as
OBSREF         varchar(64) not null, -- PATT or other reference
PROJECT        varchar(64) not null, -- Time-allocation code
MSBID          varchar(64) not null, -- Id min.-schedulable block
OBJECT         varchar(64) not null, -- Object name from telescope
--
-- NB!!
--
fieldNum       int not null, -- UKIDSS standard field ID
--
--  Essential that we have this for curation purposes.
--
RECIPE         varchar(64) not null, -- Data reduction recipe to be used
-- OBSTYPE = 'OBJECT  '           / Type (BIAS|DARK|ARC|FLAT|OBJECT|SKY)
OBSNUM         int not null, -- Observation number
GRPNUM         int not null, -- Group number applied to all members
GRPMEM         int not null, -- Group membership
STANDARD       varchar(1) not null, -- Is target a standard star observation?
NOFFSETS       smallint not null, -- Number of offset positions in a pattern
NJITTER        smallint not null, -- Number of positions in tel pattern
JITTER_I       smallint not null, -- Serial number in this tel jitter pattern
JITTER_X       real not null, -- [arcsec] X (RA) offset in tel jitter pattern
JITTER_Y       real not null, -- [arcsec] Y (Dec) offset in tel jitter pattern
NUSTEP         smallint not null, -- Number of positions in microstep pattern
USTEP_I        smallint not null, -- Serial number in this microstep pattern
USTEP_X        real not null, -- [arcsec] X (RA) offset in microstep pattern
USTEP_Y        real not null, -- [arcsec] Y (Dec) offset in microstep pattern
```

```
NFOC            smallint not null, -- Number of positions in focus scan
NFOCSCAN        smallint not null, -- Number of focus scans in focus test
-- UTDATE  = '20010607'           / UT date as integer in yyyymmdd format
DATE_OBS        varchar(64) not null, -- Date and time (UTC) of start of obs
DATE_END        varchar(64) not null, -- Date and time (UTC) of end of obs
-- WCSAXES =                    2 / Number of axes in world co-ordinate system
RADESYS         varchar(4) not null, -- Mean equator of co-ordinates
EQUINOX         real not null, -- [yr] Equinox of object position
RABASE          float not null, -- [h] Right ascension of base position
DECBASE         float not null, -- [deg] Declination of base position
TRAOFF          real not null, -- [arcsec] Right ascension telescope offset
TDECOFF         real not null, -- [arcsec] Declination telescope offset
AMSTART         real not null, -- Airmass at start of observation
AMEND           real not null, -- Airmass at end of observation
TELRA           float not null, -- [h] Current telescope right ascension
TELDEC          float not null, -- [deg] Current telescope declination
GSRA            float not null, -- [h] Right ascension of guide star
GSDEC           float not null, -- [deg] Declination of guide star
-- READMODE= 'CDS_v1  '           / Name of camera readmode
-- CAPPLICN= 'dunno   '           / Name of camera readout application
-- READOUT = 'CDS     '           / Camera readout (CDS|NDR|SAR|RRR)
EXP_TIME        real not null, -- [s] Integration time per exposure
NEXP            smallint not null, -- Number of exposures in integration
READINT         real not null, -- [s] Interval between reads
NREADS          smallint not null, -- Number of reads per exposure
-- FILTER  = 'J       '           / Combined filter name
filterID        tinyint not null, -- UID of combined filter (assigned in WSA)
FOC_MM          real not null, -- [mm] Focus position
AIRTEMP         real not null, -- [degC] Air temperature
BARPRESS        real not null, -- Ambient pressure
DEWPOINT        real not null, -- [degC] Dewpoint
DOMETEMP        real not null, -- [degC] Dome temperature
HUMIDITY        real not null, -- Relative Humidity
MIRRBSW         real not null, -- [degC] Temperature mirror B SW
MIRRNE          real not null, -- [degC] Mirror temperature NE
MIRRNW          real not null, -- [degC] Mirror temperature NW
MIRRSE          real not null, -- [degC] Mirror temperature SE
MIRRSW          real not null, -- [degC] Mirror temperature SW
MIRRBTNW        real not null, -- [degC] Mirror bottom temp. NW
MIRRTPNW        real not null, -- [degC] Mirror top temp. NW
SECONDAR        real not null, -- [degC] Temperature of secondary
TOPAIRNW        real not null, -- [degC] Top air NW
TRUSSENE        real not null, -- [degC] Truss leg ENE
TRUSSWSW        real not null, -- [degC] Truss leg WSW
WINDDIR         real not null, -- [deg] Wind direction, azimuth
WINDSPD         real not null, -- [km/h] Wind speed
CSOTAU          real not null, -- Tau at 225 GHz from CSO
TAUDATE         varchar(64) not null, -- Time and date opf Tau reading
TAUSRC          varchar(64) not null, -- Source of opacity data
CNFINDEX        int not null, -- Configuration index
--
```

```
--  end of PHDU FITS keys.
--
fileName       varchar(256) not null, -- the filename for the multiframe,
--             eg. server:/path/filename.fit
CatName        varchar(256) not null, -- the filename of the associated
--             catalogue MEF, eg. server:/path/filename.fits
-- from Detector ext HDU: more sensible if these are here:
-- FLATCOR        varchar(64) not null, -- filename of flat correction
-- RSTANOM        varchar(64) not null, -- Reset anomally correction info
-- CIR_CPM        varchar(64) not null, -- filename of Confidence map
-- SKYSUB         varchar(64) not null, -- filename of sky sub frame
-- CIR_OPM = 'irx_6523_c4_015_opm.fits[0]' / Object mask
darkID         bigint not null, -- UID of library calibration dark frame
confID         bigint not null, -- UID of library calibration confidence frame
flatID         bigint not null, -- UID of library calibration flatfield frame
frinID         bigint not null, -- UID of library calibration fringe frame
skyID          bigint not null  -- UID of library calibration sky sub frame
       )

CREATE TABLE SurveyField(
--
--  This table lists details of standard survey fields used for the public
--  WFCAM surveys (ie. UKIDSS).
--
--  Required constraints: primary key is (programmeID,fieldNum)
--                        programmeID references Programme(programmeID)
--
programmeID   int not null, -- UID of the archived programme
fieldNum      int not null, -- field ID, unique within a given programme,
--            from survey definition
RA            float not null, -- [radians] Right Ascension of field centre
Dec           float not null, -- [radians] Declination of field centre
cx            float not null, -- unit vector of spherical co-ordinate
cy            float not null, -- unit vector of spherical co-ordinate
cz            float not null, -- unit vector of spherical co-ordinate
HTMID         bigint not null, -- HTM index, 20 digits, for co-ordinate
              )

CREATE TABLE DetectorFrame(
--
--  This table contains details of individual detector frames that are part
--  of a multiframe.
--
--  Required constraints: primary key is (multiframeID,extNum)
--                        multiframeID references Multiframe(multiframeID)
--
multiframeID   bigint not null, -- the UID of the relevant multiframe
extNum         tinyint not null, -- the extension number of this frame
CompFile       varchar(256) not null, -- the filename of the compressed image
--             of this image frame, eg. server:/path/filename.jpg
JulianDay      int not null, -- the Julian Day number of the UKIRT night
```

```
deviceID       tinyint not null, -- a device UID identifying every IR device
filterID       tinyint not null, -- UID of combined filter (assigned in WSA)
--             over the lifetime of WFCAM.
--
--  Extension HDU FITS keys:
--
--XTENSION= 'IMAGE   '           / IMAGE extension
BITPIX         smallint not null, -- number of bits per data pixel, eg -32
NAXIS          tinyint not null, -- number of data axes; eg. 2
NAXIS1         int not null, -- length of data axis 1
NAXIS2         int not null, -- length of data axis 2
PCOUNT         smallint not null, -- FITS bintable required keyword; must = 0
GCOUNT         smallint not null, -- FITS bintable required keyword; must = 1
DETECTOR       varchar(16) not null, -- Detector array used, eg. "ALLADIN"
NINT           smallint not null, -- Number of integrations in observation
DROWS          smallint not null, -- [pixel] Number of detector rows
DCOLUMNS       smallint not null, -- [pixel] Number of detector columns
RDOUT_X1       smallint not null, -- Start column of array readout
RDOUT_X2       smallint not null, -- Start column of array readout
RDOUT_Y1       smallint not null, -- Start row    of array readout
RDOUT_Y2       smallint not null, -- Start row    of array readout
PIXLSIZE       real not null, -- [arcsec] Pixel size
--FLATCOR = 'Done with: J_flat_4.fit'
--RSTANOM = 'Done with medlinfilt: 50 25'
--CIR_CPM = 'wfcam_6523_conf.fit[4]' / Confidence map
--SKYSUB  = 'Done TILE_SKY: sky_6523.fits[0] 1.082' / Sky subtraction info
CIRMED         real not null, -- Latest estimate of background
CIR_BVAR       real not null, -- Latest estimate of background variance
CIR_ZERO       real not null, -- Pedestal value relative to group average
CIR_SCAL       real not null, -- Background scale relative to group maximum
--
--  First-pass CASU astrometric solution coefficients moved to calibration
--  entity
CIR_XOFF       real not null, -- Dither offset X
CIR_YOFF       real not null, -- Dither offset Y
-- Can't do much with these until we know how many and what the info is:
--HISTORY 20030305 10:28:51
--HISTORY    $Id: cir_imcore.c,v 1.4 2003/02/03 09:32:36 jim Exp $
              )

CREATE TABLE DarkFrame(
--
--  This table contains details of all library dark multiframes used in
--  the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (darkID)
--
darkID         bigint not null, -- the UID of the dark multiframe made up from
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
```

```
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
fileName       varchar(256) not null -- the filename of the darkframe
--
--  dark frame primary HDU FITS keys go here:
--
               )

CREATE TABLE ConfFrame(
--
--  This table contains details of all library confidence multiframes used in
--  the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (confID)
--
confID         bigint not null, -- the UID of the conf multiframe made up from
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
fileName       varchar(256) not null -- the filename of the confidence frame
--
--  confidence frame primary HDU FITS keys go here:
--
               )

CREATE TABLE FlatFrame(
--
--  This table contains details of all library flatfield multiframes used in
--  the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (flatID)
--
flatID         bigint not null, -- the UID of the flat multiframe made up from
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
fileName       varchar(256) not null -- the filename of the flatfield frame
--
--  flatfield frame primary HDU FITS keys go here:
--
               )

CREATE TABLE FrinFrame(
--
--  This table contains details of all library defringe multiframes used in
--  the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (frinID)
--
frinID         bigint not null, -- the UID of the fringe multiframe made from
```

```
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
fileName       varchar(256) not null -- the filename of the defringe frame
--
--  defringe frame primary HDU FITS keys go here:
--
               )

CREATE TABLE SkyFrame(
--
--  This table contains details of all library sky subtraction multiframes
--  used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (skyID)
--
skyID          bigint not null, -- the UID of the sky multiframe made up from
--             the UKIRT run number, the date observed and the date ingested
UKIRTRunNo     int not null, -- the original UKIRT run number (from filename)
UTDATE         int not null, -- the observation date (YYYYMMDD)
UTDateIngested int not null, -- the ingestion date of the multiframe (YYYYDDMM)
fileName       varchar(256) not null -- the filename of the sky frame
--
--  Darkframe primary HDU FITS keys go here:
--
               )

CREATE TABLE DarkDetFrame(
--
--  This table contains detector-by-detector based details of all library
--  dark frames used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (darkID,extNum)
--                        darkID references DarkFrame(darkID)
--
darkID         bigint not null, -- the UID of the dark multiframe
extNum         tinyint not null, -- the extension number of this frame
compImName     varchar(256) not null  -- the name of the compressed image file
--             of this detector image
--
--  dark frame extension HDU FITS keys go here:
--
               )

CREATE TABLE ConfDetFrame(
--
--  This table contains detector-by-detector based details of all library
--  confidence frames used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (confID,extNum)
```

```
--                        confID references ConfFrame(confID)
--
confID          bigint not null, -- the UID of the confidence multiframe
extNum          tinyint not null, -- the extension number of this frame
compImName      varchar(256) not null  -- the name of the compressed image file
--              of this detector image
--
--  confidence frame extension HDU FITS keys go here:
--
                )

CREATE TABLE FlatDetFrame(
--
--  This table contains detector-by-detector based details of all library
--  flatfield frames used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (flatID,extNum)
--                        flatID references FlatFrame(flatID)
--
flatID          bigint not null, -- the UID of the flatfield multiframe
extNum          tinyint not null, -- the extension number of this frame
compImName      varchar(256) not null  -- the name of the compressed image file
--              of this detector image
--
--  flatfield frame extension HDU FITS keys go here:
--
                )

CREATE TABLE FrinDetFrame(
--
--  This table contains detector-by-detector based details of all library
--  defringe frames used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (frinID,extNum)
--                        frinID references FrinFrame(frinID)
--
frinID          bigint not null, -- the UID of the defringe multiframe
extNum          tinyint not null, -- the extension number of this frame
compImName      varchar(256) not null  -- the name of the compressed image file
--              of this detector image
--
--  defringe frame extension HDU FITS keys go here:
--
                )

CREATE TABLE SkyDetFrame(
--
--  This table contains detector-by-detector based details of all library
--  sky subtraction frames used in the pipeline and propagated into the WSA.
--
--  Required constraints: primary key is (skyID,extNum)
```

```
--                        skyID references SkyFrame(skyID)
--
skyID          bigint not null, -- the UID of the sky subtraction multiframe
extNum         tinyint not null, -- the extension number of this frame
compImName     varchar(256) not null  -- the name of the compressed image file
--             of this detector image
--
--  sky subtraction frame extension HDU FITS keys go here:
--
               )


--
-- CalibSchema.sql
--
-- Database schema file for calibration information held in
-- the WFCAM Science Archive. Contains sql scripts to create
-- the relevant tables.
--
-- Original author: Nigel Hambly, WFAU, IfA, University of Edinburgh
--
-- Revision history:
-- Login name of the user who checked in this revision:  $Author$
-- Date and time (UTC) when the revision was checked in: $Date$
-- Login name of the user who locked the revision:       $Locker$
-- CVS revision number:                                  $Revision$
--

CREATE TABLE MultiframeSetUp(
--
--  This table details combinations of filter set-ups within a given night
--
--  Required constraints: primary key is (JulianDay,filterID)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null  -- the internal WSA filter UID
              )

CREATE TABLE CurrentNightlyExt(
--
-- This table contains, for every combination of filters/observing nights,
-- the nightly extinction values calculated for the most recent run
-- of a constrained, global photometric solution.
--
--  Required constraints: primary key is (JulianDay,filterID)
--                        (JulianDay,filterID) references
--                           MultiframeSetUp(JulianDay,filterID)
--                        versNum references PhotCalVers(versNum)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null,  -- the internal WSA filter UID
versNum       int not null, -- the version number of the calibration
```

```
coeffA        float not null, -- the first-order extinction coefficient
coeffB        float not null  -- the second-order extinction coefficient
              )

CREATE TABLE PreviousNightlyExt(
--
-- This table contains, for every combination of filters/observing nights,
-- the nightly extinction values calculated previously for runs
-- of a constrained, global photometric solution.
--
--  Required constraints: primary key is (JulianDay,filterID,versNum)
--                        (JulianDay,filterID) references
--                           MultiframeSetUp(JulianDay,filterID)
--                        versNum references PhotCalVers(versNum)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null,  -- the internal WSA filter UID
versNum       int not null, -- the version number of the calibration
coeffA        float not null, -- the first-order extinction coefficient
coeffB        float not null  -- the second-order extinction coefficient
              )

CREATE TABLE PhotCalVers(
--
--  This table contains details of all photometric calibration versions
--  that have existed (including the current version) in the WSA.
--
--  Required constraints: primary key is (versNum)
--
versNum       int not null, -- the version number of the calibration
startDate     float not null, -- MJD of the start time for this version
endDate       float not null -- MJD of the end time for this calibration
              )

CREATE TABLE CurrentNightlyZP(
--
--  This table contains, for every combination of detector/night/filter,
--  the current nightly zeropoint coefficients of the photometric calibration
--
--  Required constraints: primary key is (JulianDay,filterID,deviceID)
--                        (JulianDay,filterID,deviceID) references
--                           DetectorSetUp(JulianDay,filterID,deviceID)
--                        versNum references PhotCalVers(versNum)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null,  -- the internal WSA filter UID
deviceID      tinyint not null, -- the device UID
versNum       int not null, -- the version number of the calibration
zp1           float not null, -- the zeroth-order zeropoint coefficient
zp2           float not null, -- the first-order zeropoint coefficient
zp3           float not null  -- the second-order zeropoint coefficient
```

```
             )

CREATE TABLE PreviousNightlyZP(
--
--  This table contains, for every combination of detector/night/filter,
--  previous nightly zeropoint coefficients of the photometric calibration
--
--  Required constraints: primary key is (JulianDay,filterID,deviceID,versNum)
--                        (JulianDay,filterID,deviceID) references
--                           DetectorSetUp(JulianDay,filterID,deviceID)
--                        versNum references PhotCalVers(versNum)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null,  -- the internal WSA filter UID
deviceID      tinyint not null, -- the device UID
versNum       int not null, -- the version number of the calibration
zp1           float not null, -- the zeroth-order zeropoint coefficient
zp2           float not null, -- the first-order zeropoint coefficient
zp3           float not null  -- the second-order zeropoint coefficient
              )

CREATE TABLE DetectorSetUp(
--
-- This table contains every combination of detector/night/filter for science
-- detector frame observations.
--
-- Required constraints: primary key is (JulianDay,filterID,deviceID)
--
JulianDay     int not null, -- the Julian Day number of the UKIRT night
filterID      tinyint not null,  -- the internal WSA filter UID
deviceID      tinyint not null -- the device UID
              )

CREATE TABLE CurrentAstrometry(
--
--  This table contains the current astrometric calibration coefficients for
--  each science detector frame that is a part of a multiframe.
--
--  Required constraints: primary key is (multiframeID,extNum)
--                        (multiframeID,extNum) references
--                           DetectorFrame(multiframeID,extNum)
--                        versNum references AstrCalVers(versNum)
--
multiframeID   bigint not null, -- the UID of the relevant multiframe
extNum         tinyint not null, -- the extension number of this frame
versNum        int not null, -- Version number of current astrometric solution
CTYPE1         varchar(8) not null, -- G&C WCS type, eg 'RA---ZPN'
CTYPE2         varchar(8) not null, -- G&C WCS type, eg 'DEC--ZPN'
CRVAL1         float not null,
CRVAL2         float not null,
CRPIX1         float not null,
```

```
CRPIX2          float not null,
CD1_1           float not null,
CD2_1           float not null,
CD1_2           float not null,
CD2_2           float not null,
PROJP1          float not null,
PROJP3          float not null,
WCSPASS         tinyint not null, -- Pass level of WCS
NUMBRMS         int not null, -- No. of astrometric standards used in fit
STDCRMS         float not null, -- RMS residual of fit to astrometric standards
distortMap      varchar(256), -- filename of 2d non-linear distortion map
centralRA       float not null, -- [radians] RA (J2000) at device centre
centralDec      float not null, -- [radians] Dec (J2000) at device centre
HTMID           bigint not null, -- HTM index (20 deep) of device centre
posAngle        float not null  -- [radians] orientation of image y-axis to N-S
                )

CREATE TABLE PreviousAstrometry(
--
--  This table contains previous astrometric calibration coefficients for
--  each science detector frame that is a part of a multiframe.
--
--  Required constraints: primary key is (multiframeID,extNum,versNum)
--                        (multiframeID,extNum) references
--                           DetectorFrame(multiframeID,extNum)
--                        versNum references AstrCalVers(versNum)
--
multiframeID    bigint not null, -- the UID of the relevant multiframe
extNum          tinyint not null, -- the extension number of this frame
versNum         int not null, -- Version number of current astrometric solution
CTYPE1          varchar(8) not null, -- G&C WCS type, eg 'RA---ZPN'
CTYPE2          varchar(8) not null, -- G&C WCS type, eg 'DEC--ZPN'
CRVAL1          float not null,
CRVAL2          float not null,
CRPIX1          float not null,
CRPIX2          float not null,
CD1_1           float not null,
CD2_1           float not null,
CD1_2           float not null,
CD2_2           float not null,
PROJP1          float not null,
PROJP3          float not null,
WCSPASS         tinyint not null, -- Pass level of WCS
NUMBRMS         int not null, -- No. of astrometric standards used in fit
STDCRMS         float not null, -- RMS residual of fit to astrometric standards
distortMap      varchar(256), -- filename of 2d non-linear distortion map
centralRA       float not null, -- [radians] RA (J2000) at device centre
centralDec      float not null, -- [radians] Dec (J2000) at device centre
HTMID           bigint not null, -- HTM index (20 deep) of device centre
posAngle        float not null  -- [radians] orientation of image y-axis to N-S
                )
```

```
CREATE TABLE AstrCalVers(
--
--  This table contains a record for every astrometric calibration version in
--  the archive.
--
--  Required constraints: primary key is (versNum)
--
versNum       int not null, -- Version number of current astrometric solution
startDate     float not null, -- MJD of the start time for this version
endDate       float not null -- MJD of the end time for this calibration
              )
```

More scripts are available online from the WSA website [23].

10.4 Constraints

```
--
-- WSAConstraints.sql
--
-- File to add primary and foreign key constraints to the tables of the WSA.
--
-- Original author: Nigel Hambly, WFAU, IfA, University of Edinburgh
--
-- Revision History (CVS repository):
-- Login name of the user who checked in this revision:  $Author$
-- Date and time (UTC) when the revision was checked in: $Date$
-- Login name of the user who locked the revision:       $Locker$
-- CVS revision number:                                  $Revision$
--

-- PRIMARY KEYS:

-- Science multiframe database constraints:

ALTER TABLE Programme ADD CONSTRAINT pk_Prog PRIMARY KEY (programmeID)
ALTER TABLE Filter ADD CONSTRAINT pk_Filt PRIMARY KEY (filterID)
ALTER TABLE DifferenceImage ADD CONSTRAINT pk_Dif_Imag PRIMARY KEY (diffimID)
ALTER TABLE Multiframe ADD CONSTRAINT pk_Mul_Fram PRIMARY KEY (multiframeID)
ALTER TABLE SurveyField ADD CONSTRAINT pk_Sur_Fiel PRIMARY KEY (programmeID,fieldNum)
ALTER TABLE DetectorFrame ADD CONSTRAINT pk_Det_Fram PRIMARY KEY (multiframeID,extNum)
ALTER TABLE DarkFrame ADD CONSTRAINT pk_Dar_Fram PRIMARY KEY (darkID)
ALTER TABLE ConfFrame ADD CONSTRAINT pk_Con_Fram PRIMARY KEY (confID)
ALTER TABLE FlatFrame ADD CONSTRAINT pk_Fla_Fram PRIMARY KEY (flatID)
ALTER TABLE FrinFrame ADD CONSTRAINT pk_Fri_Fram PRIMARY KEY (frinID)
ALTER TABLE SkyFrame ADD CONSTRAINT pk_Sky_Fram PRIMARY KEY (skyID)
ALTER TABLE DarkDetFrame ADD CONSTRAINT pk_Dar_Det_Fram PRIMARY KEY (darkID,extNum)
ALTER TABLE ConfDetFrame ADD CONSTRAINT pk_Con_Det_Fram PRIMARY KEY (confID,extNum)
ALTER TABLE FlatDetFrame ADD CONSTRAINT pk_Fla_Det_Fram PRIMARY KEY (flatID,extNum)
ALTER TABLE FrinDetFrame ADD CONSTRAINT pk_Fri_Det_Fram PRIMARY KEY (frinID,extNum)
ALTER TABLE SkyDetFrame ADD CONSTRAINT pk_Sky_Det_Fram PRIMARY KEY (skyID,extNum)
```

```
-- Calibration constraints:

ALTER TABLE MultiframeSetUp ADD CONSTRAINT pk_Mul_Fra_Set_Up PRIMARY KEY (JulianDay,filterID)
ALTER TABLE CurrentNightlyExt ADD CONSTRAINT pk_Cur_Nig_Ext PRIMARY KEY (JulianDay,filterID)
.
.
.
```

Once again, more constraint details can be viewed online at the WSA website [23].

11 ACRONYMS & ABBREVIATIONS

ADnn : Applicable Document No. nn
API : Application Program Interface
BLOB : Binary Large OBject
CASU : Cambridge Astronomical Survey Unit
Combiframe: WSA parlance for any image that is a combination of two or more Multiframes
CSV : Comma Separated Value
CVS : Concurrent Versions System
DBI : Data Base Interface
DBMS : DataBase Management System
DXS : Deep eXtragalactic Survey (UKIDSS)
ERM : Entity–Relationship Model
ESO : European Southern Observatory
FITS : Flexible Image Transport System
GCS : Galactic Clusters Survey (UKIDSS)
GPS : Galactic Plane Survey (UKIDSS)
HDU : Header Data Unit (FITS nomenclature)
HEALPix : Hierarchical Equal Area isoLatitude PIXelisation
HTM : Hierarchical Triangular Mesh
JAC : Joint Astronomy Centre
LAS : Large Area Survey (UKIDSS)
MEF : Multi–Extension FITS
Multiframe : WSA parlance for any frame consisting of several distinct device images
OODBMS : Object–Oriented DBMS
RDBMS : Relational DBMS
SDSS : Sloan Digitial Sky Survey
SQL : Structured Query Language
UID : Unique IDentifier
UDS : Ultra Deep Survey (UKIDSS)
UKIDSS : UKIRT Deep Infrared Sky Survey
UKIRT : United Kingdom Infrared Telescope
VISTA: Visible and Infrared Survey Telescope for Astronomy
WCS : World Co-ordinate System
WFAU : Wide Field Astronomy Unit (Edinburgh)
WSA : WFCAM Science Archive
2MASS : 2 Micron All–Sky Survey

12 APPLICABLE DOCUMENTS

AD01	WSA Science Requirements Analysis Document [17]	VDF-WFA-WSA-002 Issue: 1.3, 20/03/03
AD02	UKIDSS Proposal [22]	
AD03	WSA Interface Control Document [18]	VDF-WFA-WSA-004 Issue: 1.0, 2/04/03
AD04	WSA Data Flow Document [19]	VDF-WFA-WSA-005 Issue: 1.0, 2/04/03
AD05	WSA Hardware Design Document [20]	VDF-WFA-WSA-006 Issue: 1.0, 2/04/03
AD06	Usages of the WFCAM Science Archive	
AD07	WFCAM Science Archive Management and Planning document [21]	VDF-WFA-WSA-002 Issue: 1.0, 2/04/03

13 CHANGE RECORD

Issue	**Date**	**Section(s) Affected**	**Description of Change/Change Request Reference/Remarks**
Draft	18/03/03	All	New document
1.0	02/04/03		First issue (for CDR)

14 NOTIFICATION LIST

The following people should be notified by email whenever a new version of this document has been issued:

WFAU: P Williams, N Hambly
CASU: M Irwin, J Lewis
QMUL: J Emerson
ATC: M Stewart
JAC: A Adamson
UKIDSS: S Warren, A Lawrence