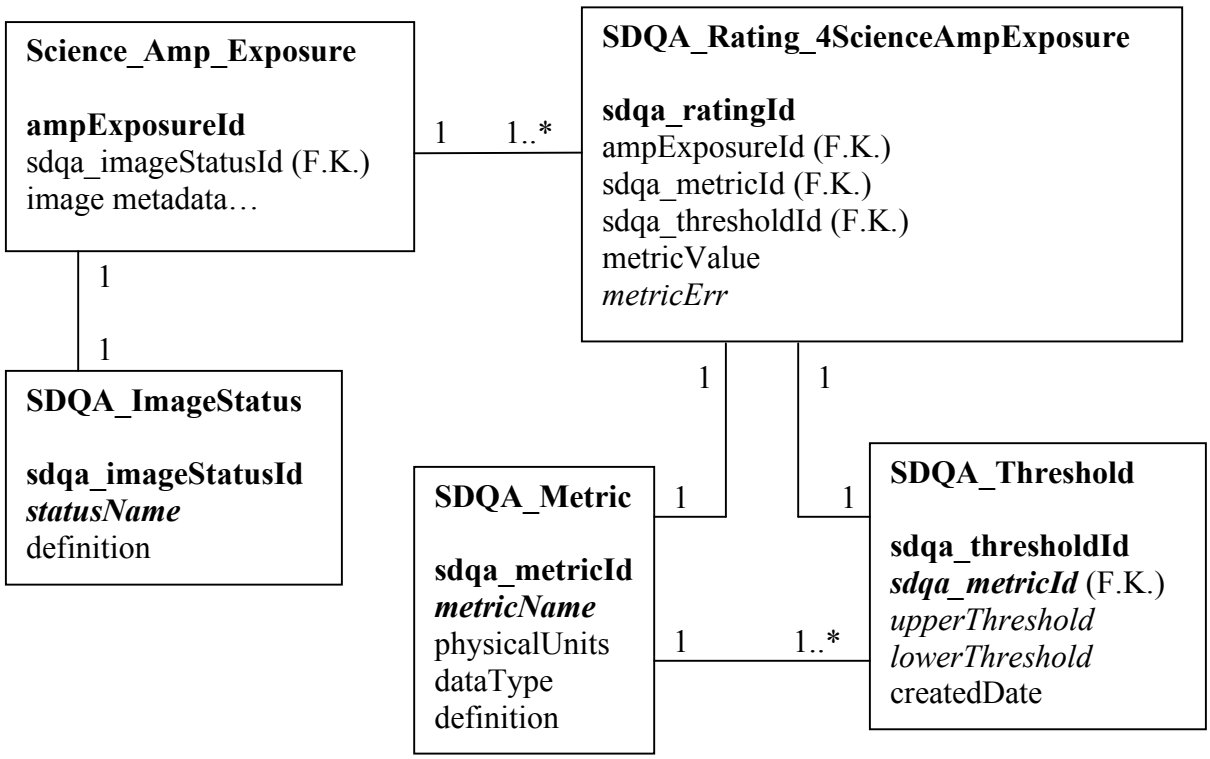
Proposed Database Schema for LSST SDQA Subsystem (Image Data Only)

Russ Laher
8 December 2008

Database Table Name	Primary Key	Comment (except for Science_Amp_Exposure, all other tables are new)
Science_Amp_Exposure	ampExposureId	Metadata about image data at the amplifier level.
SDQA_ImageStatus	sdqa_imageStatusId	Unique set of status names and their definitions, to be defined (e.g., "passed", "failed", etc.). There will be only a handful of these records.
SDQA_Rating_4ScienceAmpExposure	sdqa_ratingId	Various SDQA ratings for a given amplifier image. There will be approximately 30 of these records per image record.
SDQA_Metric	sdqa_metricId	Unique set of metric names and associated metadata (e.g., "nDeadPix", "median", etc.). There will be approximately 30 records total in this table.
SDQA_Threshold	sdqa_thresholdId	Version-controlled metric thresholds. Total number of these records is approximately equal to 30 times the number of times the thresholds will be changed over the entire period of LSST operations (of order ~100), with most of the changes occurring in the first year of operations.



Database-field nomenclature:

1. Bold-faced font means primary key (serial).
2. Bold-faced, italicized font means alternate primary key(s).
3. Regular-faced font means not-null field.
4. Regular-faced, italicized font means null field.
5. Foreign keys are indicated with (F.K.).

Detailed Description of Database-Table Fields

Database Table: SDQA_ImageStatus

Field	Data Type	Not NULL	Unique	Description
sdqa_imageStatusId	Int2	True	True	P.K.
statusName	Varchar(30)	True	True	Alt. P.K. One-word, camel-case, descriptive name of a possible image status (e.g., "passedAuto", "marginallyPassedManual", etc.)
definition	Varchar(255)	True	False	Detailed definition of the image status.

Database Table: SDQA_Rating_4ScienceAmpExposure

Field	Data Type	Not NULL	Unique	Description
sdqa_ratingId	Int8	True	True	P.K.
ampExposureId	Int8	True	True	F.K. Pointer to Science_Amp_Exposure table.
sdqa_metricId	Int2	True	True	F.K. Pointer to SDQA_Metric table.
sdqa_thresholdId	Int2	True	True	F.K. Pointer to SDQA_Threshold table.
metricValue	Float8	True	False	Value of SDQA metric.
metricErr	Float8	False	False	Uncertainty of SDQA metric.

Database Table: SDQA_Metric

Field	Data Type	Not NULL	Unique	Description
sdqa_metricId	Int2	True	True	P.K.
metricName	Varchar(30)	True	True	Alt. P.K. One-word, camel-case, descriptive name of a possible metric (e.g., "nSatPix", "median", etc.)
physicalUnits	Varchar(30)	True	False	Physical units of metric.
dataType	Char(10)	True	False	Flag indicating whether data type of the metric value is integer (0) or float (1).
definition	Varchar(255)	True	False	Detailed definition of the metric.

Database Table: SDQA_Threshold

Field	Data Type	Not NULL	Unique	Description
sdqa_thresholdId	Int2	True	True	P.K.
sdqa_metricId	Int2	True	True	First of two alt. P.K.s. Pointer to SDQA_Metric table.
upperThreshold	Float8	False	False	Threshold for which a metric value is tested to be greater than.
lowerThreshold	Float8	False	False	Threshold for which a metric value is tested to be less than.
createdDate	TimeStamp	True	False	Database timestamp when the record is inserted.

Discussion

The SDQA_Metric.metricName could subsume parametric dependence on filter, air mass, seeing, SNR, etc., or the associated SDQA_Threshold.<upper|lower>threshold values could be parameterized in terms of these independent variables. The method to be selected is TBD, and depends partly on results from the modelling effort to parameterize the thresholds.

The SDQA_Rating records will be bulk-loaded (no loading via database stored functions). However, this table can be queried via database stored functions, in which case, the function can threshold the metric value and return a flag value, equal to -1, 1, or 0, depending upon whether the value is below lower threshold, above upper threshold, or between lower and upper thresholds (for metrics with both upper and lower thresholds). Flag values may prove useful in evaluating the sdqa_imageStatusId of an image.

The above database schema is shown in relation to processed images at the amplifier level. It could easily be replicated for storing SDQA information for processed images at the CCD level, and for raw images at either amplifier or CCD level. One reason for having some SDQA information at the amplifier-level: an amplifier/channel may wind up malfunctioning, and this cannot be pinpointed if only CCD-level information is available.

Sample Database Content

Note that all content is provisional and subject to change during development.

The SDQA_ImageStatus and SDQA_Metric records, once defined, will be sacrosanct. E.g.,

```
ptf3=# select * from sdqa_imageStatus order by sdqa_imageStatusid;

sdqa_imageStatusid |       statusname        |                                 definition
-------------------+-------------------------+--------------------------------------------
                 1 | passedAuto              | Image passed by automated SDQA.
                 2 | marginallyPassedAuto    | Image marginally passed by automated SDQA.
                 3 | marginallyFailedAuto    | Image marginally failed by automated SDQA.
                 4 | failedAuto              | Image failed by automated SDQA.
                 5 | indeterminateAuto       | Image is indeterminate by automated SDQA.
                 6 | passedManual            | Image passed by manual SDQA.
                 7 | marginallyPassedManual  | Image marginally passed by manual SDQA.
                 8 | marginallyFailedManual  | Image marginally failed by manual SDQA.
                 9 | failedManual            | Image failed by manual SDQA.
                10 | indeterminateManual     | Image is indeterminate by manual SDQA.

(10 rows)


ptf3=# select * from sdqa_metric;

 sdqa_metricId | metricName | physicalUnits | dataType | definition
---------------+------------+---------------+----------+-----------------------
             1 | nGoodPix   | counts        | f        | Number of good pixels.
             2 | nDeadPix   | counts        | f        | Number of dead pixels.
             3 | nHotPix    | counts        | f        | Number of hot pixels.
             4 | nSpurPix   | counts        | f        | Number of spurious pixels.
             5 | nSatPix    | counts        | f        | Number of saturated pixels.
             6 | nObjPix    | counts        | f        | Number of object pixels.
             7 | nNanPix    | counts        | f        | Number of NaNed pixels.
             8 | nDirtPix   | counts        | f        | Number of pixels with dirt.
             9 | nStarPix   | counts        | f        | Number of star pixels.
            10 | nGalxPix   | counts        | f        | Number of galaxy pixels.
            11 | nObjSex    | counts        | f        | Number of SExtractor objects.
            12 | fwhmSex    | Arcsec        | t        | SExtractor FWHM of PSF.
            13 | gMean      | D.N.          | t        | Image global mean.
            14 | gMedian    | D.N.          | t        | Image global median.
            15 | cMedian1   | D.N.          | t        | Image upper-left corner median.
            16 | cMedian2   | D.N.          | t        | Image upper-right corner median.
            17 | cMedian3   | D.N.          | t        | Image lower-right corner median.
            18 | cMedian4   | D.N.          | t        | Image lower-left corner median.
            19 | gMode      | D.N.          | t        | Image global mode.
            20 | MmFlag     | counts        | f        | Image global mode.
            21 | gStdDev    | D.N.          | t        | Image global standard deviation.
            22 | gMAbsDev   | D.N.          | t        | Image mean absolute deviation.
            23 | gSkewns    | D.N.          | t        | Image skewness.
            24 | gKurtos    | D.N.          | t        | Image kurtosis.
            25 | gMinVal    | D.N.          | t        | Image minimum value.
            26 | gMaxVal    | D.N.          | t        | Image maximum value.
            27 | pTile1     | D.N.          | t        | Image 1-percentile.
            28 | pTile16    | D.N.          | t        | Image 16-percentile.
            29 | pTile84    | D.N.          | t        | Image 84-percentile.
            30 | pTile99    | D.N.          | t        | Image 99-percentile.

(30 rows)
```

The SDQA_Threshold table will contain a complete history of the threshold settings associated with each SDQA_metricId. Records for which the thresholds currently apply have vBest > 0, and no-longer-current records have vBest=0. Threshold fields that are blank (have NULL values) indicate threshold non-applicability. Here is an example:

```
ptf3=# select sdqa_thresholdid, sdqa_metricid, version, vbest, upperthreshold,
lowerthreshold from sdqa_threshold where vbest>0;
```

sdqa_thresholdid	sdqa_metricid	version	vbest	upperthreshold	lowerthreshold
61	1	3	1		7.5e+06
62	2	3	1	1000	
63	3	3	1	1000	
64	4	3	1	14000	
65	5	3	1	2500	
66	6	3	1	70000	
67	7	3	1	1000	
68	8	3	1	1000	
69	9	3	1		10
70	10	3	1		10
71	11	3	1		200
72	12	3	1	6.2	4.4
73	13	3	1	50000	10
74	14	3	1	50000	0
75	15	3	1	50000	0
76	16	3	1	50000	0
77	17	3	1	50000	0
78	18	3	1	50000	0
79	19	3	1	50000	-40
80	20	3	1	2	
81	21	3	1	1000	100
82	22	3	1	50000	10
83	23	3	1	200	10
84	24	3	1	50000	10
85	25	3	1	50000	-32000
86	26	3	1	74000	10000
87	27	3	1	1000	-1000
88	28	3	1	20000	-100
89	29	3	1	50000	5
90	30	3	1	70000	20

```
(30 rows)
```

Sample Database Queries

Return all “passed” amplifier images, assuming you know that sdqa_imageStatusId = 1 corresponds to sdqa_statusName = “passed”:

```
select ampExposureId
from Science_Amp_Exposure
where sdqa_imageStatusId = 1;
```

Return all “marginallyFailed” amplifier images, assuming you do not know the sdqa_imageStatusId corresponding to sdqa_statusName = “marginallyFailed”:

```
select ampExposureId
from Science_Amp_Exposure a, SDQA_ImageStatus b
where a.sdqa_imageStatusId = b.sdqa_imageStatusId
and b.statusName = 'marginallyFailed';
```

Return all amplifier images and their sdqa_ratingIds with metricValues that are greater than 5.0:

```
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b
where a.ampExposureId=b.ampExposureId
and metricValue > 5.0;
```

A query for specific metrics, where the corresponding sdqa_metricIds are known, might look like the following:

```
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b
where a.ampExposureId=b.ampExposureId
and (sdqa_metricId = 13 and metricValue > 5.0)
union
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b
where a.ampExposureId=b.ampExposureId
and (sdqa_metricId = 14 and metricValue > 50.0);
```

The same query for specific metrics, where the corresponding sdqa_metricIds are not known, might look like the following:

```
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Metrics c
where a.ampExposureId = b.ampExposureId
and b.sdqa_metricId = c.sdqa_metricId
and (metricName = 'gMean' and metricValue > 5.0)
union
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Metric c
where a.ampExposureId = b.ampExposureId
and b.sdqa_metricId = c.sdqa_metricId
and (metricName = 'gMedian' and metricValue > 50.0);
```

For more complex queries, complexity can be handled with temporary tables in database-stored functions. Alternatively, multiple tables can be joined; here is a sample query on metric values and their best-version thresholds, assuming the desired sdqa_metricIds are known:

```
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Threshold c
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
     and c.sdqa_metricId = 13
     and (b.metricValue > c.upperThreshold
          or b.metricValue < c.lowerThreshold)
     and c.vBest > 0)
union
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Threshold c
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
     and c.sdqa_metricId = 5
     and b.metricValue < c.upperThreshold
     and c.vBest > 0);
```

If the desired sdqa_metricIds are not known, then the sdqa_metricNames can be used:

```
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Threshold c, SDQA_Metric d
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
     and c.sdqa_metricId = d.sdqa_metricId
     and d.metricName = 'gMean'
     and (b.metricValue > c.upperThreshold
          or b.metricValue < c.lowerThreshold)
     and c.vBest > 0)
union
select a.ampExposureId, sdqa_ratingId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Threshold c, SDQA_Metric d
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
     and c.sdqa_metricId = d.sdqa_metricId
     and d.metricName = 'nSatPix'
     and b.metricValue < c.upperThreshold
     and c.vBest > 0);
```

Here is a sample self-join query of the SDQA_Rating table that returns a list of ampExposureIds for images with two metrics simultaneously subject to certain conditions:

```
select a.ampExposureId
from Science_Amp_Exposure a,
   SDQA_Rating_4ScienceAmpExposure b, SDQA_Threshold c,
   SDQA_Metric d,
   SDQA_Rating_4ScienceAmpExposure e, SDQA_Threshold f,
   SDQA_Metric g
where a.ampExposureId = b.ampExposureId
and b.ampExposureId = e.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
         and c.sdqa_metricId = d.sdqa_metricId
         and d.metricName = 'gMean'
         and (b.metricValue > c.upperThreshold
             or b.metricValue < c.lowerThreshold)
         and c.vBest > 0)
and (e.sdqa_thresholdId = f.sdqa_thresholdId
         and f.sdqa_metricId = g.sdqa_metricId
         and g.metricName = 'nSatPix'
         and (e.metricValue > f.upperThreshold)
         and f.vBest > 0);
```

The above self-join query is quite complex and does not readily lend itself to handling additional metric constraints. However, there is a simpler alternative that is easily extensible in the aforementioned regard:

```
select a.pId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
      SDQA_Threshold c, SDQA_Metric d
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
         and c.sdqa_metricId = d.sdqa_metricId
         and d.metricName = 'gMean'
         and (b.metricValue > c.upperThreshold
             or b.metricValue < c.lowerThreshold)
         and c.vBest > 0)
intersect all
select a.pId
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
      SDQA_Threshold c, SDQA_Metric d
where a.ampExposureId = b.ampExposureId
and (b.sdqa_thresholdId = c.sdqa_thresholdId
         and c.sdqa_metricId = d.sdqa_metricId
         and d.metricName = 'nSatPix'
         and b.metricValue > c.upperThreshold
         and c.vBest > 0);
```

Finally, here is an interesting query, with sample results that show metric values in relation to their lower thresholds:

```
select metricName, a.ampExposureId, sdqa_ratingId,
b.sdqa_metricId, metricValue, d.lowerThreshold
from Science_Amp_Exposure a, SDQA_Rating_4ScienceAmpExposure b,
     SDQA_Metric c, SDQA_Threshold d
where a.ampExposureId = b.ampExposureId
and b.sdqa_metricId = c.sdqa_metricId
and b.sdqa_thresholdId = d.sdqa_thresholdId
and d.lowerThreshold is not NULL
and b.metricValue < d.lowerThreshold
order by metricName, a.ampExposureId, sdqa_ratingId;
```

metricname	ampExposureId	sdqa_ratingid	sdqa_metricid	metricvalue	lowerthreshold
nGalxPix	63	1264	10	0	10
nGalxPix	64	1294	10	0	10
nGalxPix	65	1324	10	0	10
nGalxPix	66	1354	10	0	10
nGalxPix	67	1384	10	0	10
nGalxPix	68	1414	10	1	10
nGalxPix	69	1444	10	0	10
nGalxPix	70	1474	10	0	10
nGalxPix	71	1504	10	0	10
nGalxPix	72	1534	10	0	10
nGalxPix	73	1564	10	0	10
nGalxPix	74	1594	10	0	10
nGalxPix	75	1624	10	0	10
nGalxPix	76	1654	10	0	10
nGalxPix	77	1684	10	0	10
nGalxPix	78	1714	10	0	10
nGalxPix	79	1744	10	0	10
nGoodPix	64	1297	1	7.47411e+06	7.5e+06
nGoodPix	71	1507	1	7.20877e+06	7.5e+06
nObjSex	68	1424	11	169	200
nObjSex	77	1694	11	105	200
nObjSex	78	1724	11	38	200
nStarPix	77	1703	9	2	10
nStarPix	78	1733	9	1	10

(24 rows)