

Appendix: Astronomy in ArrayDB

Schema

I presume that any type may be suffixed by [{dimension list}] to turn it into a sub-array with fixed regular dimensions.

I use the syntax:

```
CREATE ARRAY Name [ {dimensions} ] ( {attributes} );
```

Dimensions may be `FIXED` (limited in range) or `VARIABLE` (unlimited in range). They may also be `REGULAR` (essentially non-negative integers) or `IRREGULAR`. If every entry in an array has a different value for one dimension, that dimension is labeled `UNIQUE`.

I use the spatial dimension before the primary key to enable proper spatial partitioning of the data.

We have one main array of observations, `Source`. Sources are associated with one or more `Objects` or `MovingObjects`. If the `Source` is associated with exactly one `Object` or `MovingObject`, it is given an `objectId` or `movingObjectId`; otherwise the field is `NULL`. If the `Source` is associated with multiple `Objects` or `MovingObjects`, it has an entry in one or both of the one-to-many mapping tables `_Source2Object` and `_Source2MovingObject`.

`Object` is a (likely materialized) view that joins aggregates over `Source` with externally-computed summaries and classifications. Some of the details of how `Sources` associated with multiple `Objects` are rolled up are not yet known. `MovingObject` is not shown here but is a separate similar view of the `Sources` that have `movingObjectIds`.

We also have dimension arrays `ObjectType` and `Filter` and a one-to-many mapping table, `_Object2Type`.

```
CREATE ARRAY Source
[
  raDec1 SPH_COORD_DEG(DOUBLE) IRREGULAR VARIABLE COMMENT 'RA and Dec
    coordinates of the source centroid (degrees). Need to support
    accuracy ~0.0001 arcsec',
  taiMidPoint DOUBLE IRREGULAR VARIABLE COMMENT 'If a DIASource
    corresponds to a single exposure, taiMidPoint represents tai
    time of the middle of exposure. For multiple exposures, this
    is middle of beginning-of-first-exposure to end-of-last-
    exposure',
  objectId BIGINT IRREGULAR VARIABLE COMMENT 'Identifier of associated
    Object, if only one.',
  movingObjectId BIGINT IRREGULAR VARIABLE COMMENT 'Identifier of
    associated Moving Object, if only one.',
  sourceId BIGINT IRREGULAR VARIABLE UNIQUE COMMENT 'Unique id.'
```

```

]
(
ampExposureId BIGINT NULL COMMENT 'Pointer to Amplifier where source
    was measured. If the Source belongs to multiple AmpExposures,
    then table Source2AmpExposure is used, and this pointer is
    NULL',
filterId TINYINT NOT NULL COMMENT 'Pointer to an entry in Filter
    table: filter used to take Exposure where this Source (or
    these Sources) were measured.',
rowCol COORD2(DOUBLE) NOT NULL COMMENT 'Pixel coordinates (Y,X) of
    the source centroid.',
c COORD3(DOUBLE) NOT NULL COMMENT 'x,y,z-components of the (RA,Dec)
    unit vector',
taiRange FLOAT NULL COMMENT 'If a DIASource corresponds to a single
    exposure, taiRange equals the exposure length. If DIASoure
    corresponds to multiple exposures, taiRange equals the end-of-
    last-exposure minus beginning-of-first-exposure',
fwhmA FLOAT NOT NULL COMMENT 'Size of the object along major axis
    (pixels).',
fwhmB FLOAT NOT NULL COMMENT 'Size of the object along minor axis
    (pixels).',
fwhmTheta FLOAT NOT NULL COMMENT 'Position angle of the major axis
    w.r.t. X-axis (measured in degrees).',
psfMag DOUBLE NOT NULL COMMENT 'PSF magnitude of the object',
apMag DOUBLE NOT NULL COMMENT 'Aperture magnitude',
modelMag DOUBLE NOT NULL COMMENT 'model magnitude (adaptive 2D
    gauss)',
petroMag DOUBLE NULL COMMENT 'Petrosian flux',
apDia FLOAT NULL COMMENT 'Diameter of aperture (pixels)',
snr FLOAT NOT NULL COMMENT 'Signal-to-Noise Ratio for the PSF
    optimal filter.',
chi2 FLOAT NOT NULL COMMENT 'Chi-square value for the PSF fit',
sky FLOAT NULL,
moment0 FLOAT NULL COMMENT 'Sum of all flux of all pixels that
    belong to a source.',
moment1_x FLOAT NULL COMMENT 'Center of light - x component.',
moment1_y FLOAT NULL COMMENT 'Center of light - y component.',
moment2_xx FLOAT NULL COMMENT 'Standard deviation about center of
    light - xx component.',
moment2_xy FLOAT NULL COMMENT 'Standard deviation about center of
    light - xy component.',
moment2_yy FLOAT NULL COMMENT 'Standard deviation about center of
    light - yy component.',
moment3_xxx FLOAT NULL COMMENT 'Skewness of the profile - xxx
    component.',
moment3_xxy FLOAT NULL COMMENT 'Skewness of the profile - xxy
    component.',
moment3_xyy FLOAT NULL COMMENT 'Skewness of the profile - xyy
    component.',
moment3_yyy FLOAT NULL COMMENT 'Skewness of the profile - yyy
    component.',
moment4_xxxx FLOAT NULL COMMENT 'Kurtosis - xxxx component.',
moment4_xxyy FLOAT NULL COMMENT 'Kurtosis - xxyy component.',
moment4_xxyy FLOAT NULL COMMENT 'Kurtosis - xxyy component.',

```

```

moment4_xyyy FLOAT NULL COMMENT 'Kurtosis - xyyy component.',
moment4_yyyy FLOAT NULL COMMENT 'Kurtosis - yyyy component.',
flag4association SMALLINT NULL,
flag4detection SMALLINT NULL,
flag4wcs SMALLINT NULL COMMENT 'Problem/special conditions
        indicator'
)
;

CREATE ARRAY _Source2Object
[
    sourceId BIGINT IRREGULAR VARIABLE,
    objectId BIGINT IRREGULAR VARIABLE
]
(
    splitPercentage FLOAT NOT NULL COMMENT 'Percentage of the Source
        associated with this Object, or probability that the Source is
        associated with this Object'
)
;

CREATE ARRAY _Source2MovingObject
[
    sourceId BIGINT IRREGULAR VARIABLE,
    movingObjectId BIGINT IRREGULAR VARIABLE
]
(
    splitPercentage FLOAT NOT NULL COMMENT 'Percentage of the Source
        associated with this MovingObject, or probability that the
        Source is associated with this MovingObject'
)
;

CREATE ARRAY ObjectSummary
[
    objectId BIGINT IRREGULAR VARIABLE UNIQUE COMMENT 'Unique id.'
]
(
    mag FLOAT[6] NULL COMMENT 'model-derived magnitudes',
    mu SPH_COORD_DEG(DOUBLE) NULL COMMENT 'derived proper motion,
        mu_alpha*cos(Dec) and mu_delta (measured in arcsec/yr)',
    parallax FLOAT NULL COMMENT 'derived parallax for the object',

    posErrA FLOAT[6] NULL COMMENT 'Large dimension of the position error
        ellipse, assuming gaussian scatter (arcsec).',
    posErrB FLOAT[6] NULL COMMENT 'Small dimension of the position error
        ellipse, assuming gaussian scatter (arcsec).',
    posErrTheta FLOAT[6] NULL COMMENT 'Orientation of the position error
        ellipse (degrees).',
    varProb SMALLINT[6] NULL COMMENT 'Probability of variability in %
        (100% = variable object). Note: large photometric errors do
        not necessarily mean variability.',

```

```

amplitude FLOAT[6] NULL COMMENT 'Characteristic magnitude scale of
    the flux variations',
primaryPeriod FLOAT NULL COMMENT 'period that represent periods for
    all filters.',
period FLOAT[6] NULL COMMENT 'Period of flux variations (if
    regular)',
ixx FLOAT[6] NULL COMMENT 'Adaptive second moment',
iyy FLOAT[6] NULL COMMENT 'Adaptive second moment',
ixy FLOAT[6] NULL COMMENT 'Adaptive second moment',
timescale FLOAT[6] NULL COMMENT 'Characteristic timescale of flux
    variations (measured in days). This is to complement period
    for variables without a well-defined period. LSST images have
    sampling frequency of ~0.1Hz.',
scalegram FLOAT[6,25] NULL COMMENT '"Scalegram": time series as the
    average of the squares of the wavelet coefficients at a given
    scale. See Scargel et al 1993 for more details.',
zone INTEGER NULL COMMENT 'zone is an index to speed up spatial
    queries.'
)
;

```

This next view uses two new primitives. ARRAY(TYPE) defines a temporary array filled with NULLs. SET_VALUE(array, value, {indexlist}) puts the value into the array at the position defined by the indexlist. All other elements of the array remain as is.

```

DEFINE U AS 1; DEFINE G AS 2; DEFINE R AS 3; DEFINE I AS 4;
DEFINE Z AS 5; DEFINE Y AS 6;

CREATE VIEW Object AS
SELECT
    AVERAGE(Source.raDecl) AS raDecl,
    MIN(taiMidpoint) AS earliestObsTime,
    MAX(taiMidpoint) AS latestObsTime,
    ObjectSummary.mag[U] - ObjectSummary.mag[G] AS ugColor,
    ObjectSummary.mag[G] - ObjectSummary.mag[R] AS grColor,
    ObjectSummary.mag[R] - ObjectSummary.mag[I] AS riColor,
    ObjectSummary.mag[I] - ObjectSummary.mag[Z] AS izColor,
    ObjectSummary.mag[Z] - ObjectSummary.mag[Y] AS zyColor,
    AVERAGE(Source.c) AS c,
    BIT_OR(Source.flag4association) AS flag4association,
    BIT_OR(Source.flag4detection) AS flag4detection,
    BIT_OR(Source.flag4wcs) AS flag4wcs,
    AVERAGE(SET_VALUE(ARRAY(FLOAT[6]), Source.psfMag, Source.filterId))
        AS psfMag[:f],
    AVERAGE(SET_VALUE(ARRAY(FLOAT[6]), Source.petroMag,
        Source.filterId)) AS petroMag[:f],
    AVERAGE(SET_VALUE(ARRAY(FLOAT[6]), Source.apMag, Source.filterId))
        AS apMag[:f],
    SUM(SET_VALUE(ARRAY(FLOAT[6]), 1, Source.filterId)) AS numObs[:f],
    ObjectSummary.*
FROM Source, ObjectSummary

```

```

WHERE Source.objectId = ObjectSummary.objectId
[
  DIMENSION(raDecl),
  DIMENSION(ObjectSummary.objectId)
]
;

CREATE ARRAY Filter
[
  filterId TINYINT REGULAR VARIABLE UNIQUE COMMENT 'Unique id'
]
(
  name VARCHAR(20) NOT NULL COMMENT 'Filter name',
  centerFreq FLOAT NOT NULL COMMENT 'Central frequency of the
    bandpass'
)
;

CREATE ARRAY _Object2Type
[
  objectId BIGINT IRREGULAR VARIABLE COMMENT 'Pointer to an entry in
    Object table',
  typeId SMALLINT REGULAR VARIABLE COMMENT 'Pointer to an entry in
    ObjectType table',
]
(
  probability TINYINT NULL DEFAULT 100 COMMENT 'Probability that given
    object is of given type. Range 0-100 %'
)
;

CREATE ARRAY ObjectType
[
  typeId SMALLINT REGULAR VARIABLE UNIQUE COMMENT 'Unique id.'
]
(
  description VARCHAR(255) NULL
)
;

```

Views

We require a view that selects star-type objects.

```

CREATE VIEW Star AS
  SELECT *
  FROM Object
  JOIN _Object2Type USING (objectId)
  JOIN ObjectType USING (typeId)
  WHERE ObjectType.description = "star"
  AND _Object2Type.probability > 90; -- 0-100%

```

Sample Queries

Find an object with particular objectId

A trivial query: fetch a single Object.

```
SELECT *
FROM Object
WHERE objectId = :objectId;
```

Select time series data for all objects in a given area of the sky, in a given photometric band

An inexpensive query: fetch the Sources for a small region of sky.

```
SELECT *
FROM Source, Filter
WHERE SPH_DIST(raDecl, SPH_COORD_DEG(:ra, :decl)) < 1800
  AND Source.filterId = Filter.filterId
  AND Filter.name = :filterName
ORDER BY objectId, taiMidPoint ASC;
```

Note the spherical distance function and that 1800 arcsec = 0.5 degree.

Search for Cataclysmic Variables and pre-CVs with White Dwarfs and very late secondaries

A set of full column scans on Objects, returning all columns.

```
SELECT *
FROM Object
WHERE ugColor < 0.4
  AND grColor < 0.7
  AND riColor > 0.4
  AND izColor > 0.4;
```

Provide a histogram of star-like objects with rare colors

Scans of Objects in the Star view with GROUP BY, HAVING, and ORDER BY. We assume a 1-based array index.

```
DEFINE U AS 1; DEFINE G AS 2; DEFINE R AS 3; DEFINE I AS 4;
DEFINE Z AS 5; DEFINE Y AS 6;

SELECT ROUND(ugColor,0) AS UG,
       ROUND(grColor,0) AS GR,
```

```

        ROUND(riColor,0) AS RI,
        ROUND(izColor,0) AS IZ,
        ROUND(zyColor,0) AS ZY,
        COUNT(*) AS pop
FROM    Star
WHERE   (mag[U]+mag[G]+mag[R]+mag[I]+mag[Z]+mag[Y]) < 150
        -- exclude bogus magnitudes (== 999)
GROUP BY UG, GR, RI, IZ, ZY
HAVING pop < 500
-- Common buckets have 500 or more members, so delete them
ORDER BY pop;

```

Find stars with stellar neighbor within distance x where at least one of the stellar neighbors has the colors of a white dwarf

```

SELECT S1.objectId AS s1, S2.objectId AS s2
FROM   Star S1 -- S1 is the white dwarf
JOIN   Star S2 -- S2 is the second star
WHERE  SPH_DIST(S1.raDecl, S2.raDecl) < 5 -- arcseconds
        AND S1.ugColor < 0.4 -- and S1 meets Paul Szkody's color cut
        AND S1.grColor < 0.7 -- for white dwarfs
        AND S1.riColor > 0.4
        AND S1.izColor > 0.4;

```

The join condition is spatial.

Find all galaxies in dense regions

```

SELECT g.objectId, g.raDecl
FROM   Galaxy AS g, Galaxy AS neighbor
WHERE  g.objectId <> neighbor.objectId
        AND SPH_DIST(g.raDecl, neighbor.raDecl) < 360 -- arcsec
GROUP BY g.objectId
HAVING COUNT(neighbor.objectId) > 10000;

```

The original query used a neighbor table; this has been replaced with a spatial self-join. The original DISTINCT/WHERE with subquery has been replaced by GROUP BY/HAVING.

Find the brightness of the closest source within ? arcmin

We assume a 1-based array index. APPROX_GT/LT takes into account the error in the position or magnitude; the third parameter is the number of standard deviations to use.

```

DEFINE U AS 1; DEFINE G AS 2; DEFINE R AS 3; DEFINE I AS 4;
DEFINE Z AS 5; DEFINE Y AS 6;

SELECT o.raDecl, o.flag4detection, o.objectId,
        o.psfMag[G], o.psfMag[R], o.psfMag[I],
        o.mag[G], o.mag[R], o.mag[I],

```

```

        o.petroMag[R],
        SPH_DIST(o.raDecl, n.raDecl) AS distance,
        n.mag[R], n.mag[G]
FROM   Object AS o, Object AS n
WHERE  APPROX_GT(o.raDecl[1], 120.0, 3.0)
      AND APPROX_LT(o.raDecl[1], 240.0, 3.0)
      AND APPROX_GT(o.mag[R], 16.0, 3.0)
      AND APPROX_LT(o.mag[R], 21.0, 3.0)
      AND n.objectId = (
          SELECT nn.objectId
          FROM   Object AS nn
          WHERE  SPH_DIST(o.raDecl, nn.raDecl) < :arcmin * 60
          ORDER BY SPH_DIST(o.raDecl, nn.raDecl)
          LIMIT 1
        );

```

This query could be rewritten without `LIMIT` using `GROUP BY` and `MIN()`, but there might not be a unique object at the minimum distance.

Find objects that have increased in brightness suddenly

We really would like to process the time series in serial fashion, ordered by `taiMidpoint`, so as not to get the duplicate entries that this self-join would give.

```

SELECT Star.objectId, Current.taiMidpoint, Past.taiMidpoint
FROM   Star, Source AS Current, Source AS Past
WHERE  Star.objectId = Current.objectId
      AND Current.objectId = Past.objectId
      AND Current.filterId = Past.filterId
      AND Current.taiMidpoint - Past.taiMidpoint < 2 WEEKS
      AND APPROX_GT(Current.psfMag - Past.psfMag, 0.1, 3.0);

```

Find all pairs of objects that have similar time series

This is likely the most expensive query we would ever want to contemplate. The idea is to gather, for each object, the sequence of `psfMag` values in each filter with their associated times and then compare these sequences using a user-defined aggregate function.

```

SELECT S1.objectId, S2.objectId,
       TimeSeriesSimilarity(
           SEQUENCE(S1.psfMag, S1.filterId, S1.taiMidpoint),
           SEQUENCE(S2.psfMag, S2.filterId, S2.taiMidpoint)
       ) AS similarity
FROM   Source AS S1, Source AS S2
WHERE  S1.objectId <> S2.objectId
GROUP BY S1.objectId, S2.objectId
HAVING similarity > 0.95
ORDER BY similarity DESC;

```