Standardizing names in C++ code and in database schema

Decisions made at the TCT meeting Apr 1st, 2009 are covered in a separate document, see Recommendations

Introduction

At the moment many names of database columns and their corresponding C++ names do not match. Here are some examples:

C++	**database**
visitTime	dataObs
exposureTime	expTime
airMass	airmass
photoZP	PHOTZP

Proposal

- We should come up with a project wide standard and apply it consistently for all database columns and all C++ names, such that database columns and their corresponding C++ names fully match wherever it makes sense (there may be unavoidable differences, for example we may want to represent per-filter data as an array in C++).
- We should standardize units, such that all corresponding database columns and their C++ attributes use the same units.

Implementation

Names

How about if we adopt the convention used for naming database columns:

- names should start with small letter. Example: imageId
- Avoid using short, common names like "id", instead, use more descriptive names like imageId, sourceId etc. This helps to simplify writing queries, for example as long as column names are distinct we can use NATURAL JOIN.
- If a name consists of multiple words, use "camel case" convention: capitalize first character of each word except the first, example sourceId, not sourceid and not sourceID
- if a column is specific to a filter, put the filter name first, for example uAmplitude, gPeriod
- put column keeping error for another column one after another, and append Err at the end, for example raErr column should be immediately after ra column
- for right ascension and declination use ra, and decl. Notice that dec is a reserved word in some DBMSes (short for DECIMAL)
- if there are more than one per-filter columns, group them together: all columns for one filter together. Put non-filter specific columns before filter-specific ones.

Units

The units in the database are currently poorly defined. We are actively working on defining units for all columns (see ticket #428), it includes checking what units are used by the C++ code. We could leverage this work to ensure the units in database and C++ are in sync.

Other notes

The ?Schema Browser which allows to browse all database column names, see their descriptions and units should be helpful in the process. (The unit definitions for database columns are kept along with the master schema in EA, as tagged values).

Comment by jbecla on Tue 31 Mar 2009 11:38:24 PM CDT

The following comment was received from Dick Shaw:

I've been meaning for some time now to comment on your proposal for standardizing names of scalars between C++ variables DB columns, which you posted on the Trac at StandardizingNames. I agree that some level of standardization between source code variable names and DB columns would be helpful. I suppose the alternative would be to document and maintain a mapping between them. I agree with many of your suggestions for the specific convention (e.g., camelCase, descriptive names), but I think that some revisions and additional detail would perhaps benefit the cause. My concerns are that the current proposal does not give much guidance for developers to create good names, and that actual practice would lead to uneven "implied" rules. Left unaddressed, this would make it hard for users who want to search for a particular kind of information to guess the corresponding name in the schema. Examples of my concerns are abundant in the CommonVocabulary, although AFAIK you are not asserting that these specific names are consistent with your proposal, right?

Also, it may be helpful to set some expectations up-front for data and metadata that are expected to be persisted and shared among sub-systems. For instance, I'm sure you would agree it is far better to define a datum or schema field fully when first introduced, than to define one partially and fill in the details later. I suggest that a complete definition consists of at least the following information:

- Name (following the convention)
- Units (quantities with no physical units should be so indicated)
- Data Type to be used for internal storage
- Default value or enumeration, and range of validity, if any
- UCD (=Uniform Content Descriptor), if relevant
- Suggestion for presentation format (e.g., "sexagesimal" for time or sky position, even if the internal storage is "double")
- Description (with enough detail to make the quantity clear to users)

I've appended my suggestions for revisions and additions to your proposed naming convention.

Regards, Dick

#---

I would suggest adding the following points to the naming convention:

o The fundamental quantity being described should appear first in the name, with modifiers concatenated afterward. A rule of thumb is to ask what the units of the quantity would be, and make sure that quantity appears first in the name.

- Ex: "dateObs", not "obsDate" for a quantity that fundamentally is a date/time of significance;
- "timeObsEarliest" (or, "timeObsFirst"), not "earliestObsTime"
- "nGoodPix" not "goodPixN" since this is fundamentally a number
- There are some historical exceptions (e.g., expTime from the FITS standard) that must be preserved

o Use care to select the most meaningful name to represent the quantity being described

- "imageMean" not "pixelMean" if we are talking about the mean value of an image, not repeated measurements of a pixel

o Names must not explicitly include units

- Ex: "skyBackground" not "skyADU" to indicate the sky background level
- "expMidpoint" rather than "taiMidPoint"; or "timeRange" not "taiRange"

o Names should _not_ include the underscore character _except_ to append a parameter designation to the root name

- Ex: the x- or y-position (e.g., wcsRms_X), or filter name abbreviation (e.g., photDepth_u)

o Acronyms should be subservient to the camelCase convention.

- Ex: psfFwhm, not psfFWHM to indicate the width at the half-power point of the point-spread function

o Acronyms should be used sparingly, and limited to very common usages in the relevant community.

- CCD, FWHM, ID, PSF, and RA would be fine as name fragments

o Avoid obscure abbreviations: clarity is probably more important than brevity.

- Ex: "apertureDiam" would be better than "apDia"

Comment by rowen on Mon 20 Apr 2009 01:24:52 PM CDT

Mostly this sounds fine to me. A few specific points, some relevant to the TCT board's findings:

- Regarding "i" as an iterator name (C++ 3-23): in my opinion it is a very poor choice for an STL-ish iterator that points to something. For that I suggest we adopt names that name the source, e.g. "inImageIter". I believe most existing code already uses a name that is more obvious than "i" and so violates rule 3-23.
- Dick Shaw's suggestion to use foo_i instead of iFoo is reasonable, but it can be tricky to decide when the underscore is wanted so I am somewhat against it. In a similar vein, I would like a rule for X and Y -- is it a prefix or a suffix? (And if a suffix, do we want X or _x)?
- Need we use decl instead of dec if it is part of a longer name? I propose not, though it makes one more thing to remember.

Add comment

Your email or username: