Time Series Data and Associated Concepts

Abstract:
This document discusses concepts critical to fully representing time series data in HDB. Most of these concepts have been presented in other documents. The new information is that pertaining to the representation of 1) statistical datatypes, and 2) calculated datatypes in HDB. A schema which supports these is presented, along with sample data.

Purpose:
Lay out, for your approval, a database design that anticipates future needed datatype transformations for use by current HDB clients. The design will be used to support the specification of calculation methods, within the constraints proposed here, that could be applied to any datatype.

Intended Audience:
HDB Developers, HDB Database of Record Team

Instantaneous and Interval Data

Instantaneous and Interval Defined

There are 2 fundamental categories of data in HDB:

- Data representing an interval; this implies that the data has been aggregated in some way. This data can be:
    - Of a regular "business" interval. These are the intervals that we use to conduct our business, and they therefore are maintained in HDB as separate tables (rather than computing everything "on the fly.")
        - hourly
        - daily
        - monthy
        - yearly
        - water yearly
        - potentially 15 minute, etc.
    - Of an "other" interval, including irregular or non-business intervals. An irregular interval might be data recorded when a "bucket" measuring precipitation becomes full and gets emptied, which would result in randomly-spaced intervals of data. A non-business interval might be "quarterly"; it is regular, but currently not an interval represented by a separate table. (Note that it could become a table in the future if needed.)
- Data which represents an instant in time
    - Definition of instantaneous = measurement taken at a point in time. This is regardless of what the *fundamental* device is doing (e.g., accumulating, averaging 10 readings taken over 20 seconds, etc.)
    - This data is *probably* not externally processed.

It is up to each installation to decide which data is instantaneous and which is interval.

➔We would like to develop guidelines to help you in distinguishing between instantaneous and interval data. One thought: if we know the start and end times, and they're different, it must be an interval value.

Instantaneous and Interval Tables in HDB

We are proposing that all categories of data (instant, other, and regular interval) be stored in separate physical tables in HDB. If new regular intervals emerge and are important for operations (e.g., 15 minute), a new table will be created to hold this data.

Note that the table r_instant exists but appears not to be used much for current business practices. R_other is a new table that will hold irregular interval and non-business interval data.

Our primary reason for having the tables r_instant and r_other is the following: All data that are copied out of r_base and into another r_ table (instant, other, or interval) are checked by the Derivation Application for compliance to valid value ranges specified in the ref_derivation_source table for each site_datatype_id. Only data which meet those criteria are copied from r_base. Hence, data outside r_base has had some level of quality checking applied to it. We anticipate that even higher levels of quality checking will be defined by the data validation effort.

Datatypes and Methods

Requirements For Datatype Representation

Our understanding of the requirements for representing datatypes, and the methods by which values are calculated, in HDB is:

- Users must be able to easily find the appropriate data set for a particular *hydrologic concept* of data, without having any knowledge of exactly *how* each data point was calculated. In other words: retrieve all end-of-period water surface elevation readings for a particular site, regardless of whether the end-of-period value was the last value in the period (within the tolerances set by the user), or whether it was interpolated to fall exactly at the end of the period.
- Users want to be able to determine, at their convenience, *exactly* how a piece of data was calculated. That is, was this particular end-of-period value the last value, or linearly interpolated?

In addition, the requirement (or request) that we heard for the representation of calculated datatypes is as follows:

- Support representation of any datatype which can be calculated (from other datatypes) using Reverse Polish Notation. Any datatypes which require a more complex algorithm will be computed using Riverware, or another external application.

Definitions

Methods and Operators

Time Aggregation Method
An aggregation method class is any class of methods which can be applied to a single datatype, over a specified interval, and result in another datatype at a larger interval. Such method classes include: ave, bop, eop, max, min, tot, tovol.

Operator
An operator (for HDB) is any arithmetic operator (e.g., +,-,*,/) which is used in creating a calculation method or a computed datatype.

Calculation Method Class
A calculation method class is any class of methods which takes two or more pieces of data related to the same datatype, all from the same interval table, and results in another realated datatype at the same interval table. Percent of normal readings (a current divided by a statistical norm) fall into this category.

Also included are accumulation and change. Change is the difference between 2 adjacent pieces of data (same datatype) in a time series. Accumulation could be calculated by adding the running accumulation from the start date to the total value for the current interval. These 2 methods differ from percent of normal in that they use values from the same interval, but different time steps within that interval.

Note that the specific instances of these method classes (for instance, percent of normal to date, percent of normal for the water year, etc) do not need to be broken down into their components in a separate table. Just as the Time Aggregation Application (Derivation Application) knows how to compute a data point average or a linearly-interpolated average, the Calculation Application will know that the percent-of-normal-to-date method class, applied to precipitation data, divides the cumulative reading to date by the statistical normal accumulation to date.

Unclassified Method Classes
Include copy, no-op (use for instantaneous and computed datatypes), unknown.

Note that for methods of any *kind, the intelligence is coded into the application.*

Datatypes

The following definitions are for the "types" of datatypes (as represented in the datatype_type field of hdb_datatype).

Simple Datatypes
A simple datatype is any datatype that represents either:

- a datatype which is not computed and which has had no methods (time aggregation or calculated) applied to it.

- essentially, it is any datatype that is not a method-applied or computed datatype.

Method-Applied Datatypes
Method-applied datatypes include those datatypes for which:

- a single time-aggregation method is applied to a series of instantaneous values; or
- a single time-aggregation method is applied to a series of values which have had the same time-aggregation method applied to them one or more times (for example, the average of an average is still just an average). See the definition of compound datatypes to distinguish this from applying an aggregation method to a series of data which has been dis-similarly aggregated.
- a single calculation method (change, accumulation, percent of normal) is applied

Computed Datatypes
A computed datatype is any datatype which is composed of multiple pieces of data (datatypes or attributes) combined arithmetically, using the operators known in HDB. For each computed datatype, there may be many ways of combining components to arrive at the desired data. The component pieces of data and the resulting data are always at the same time step. Computed datatypes include "all releases" and "consumptive use".

Computed datatypes differ from calculation methods in that the hydrologic data being computed (i.e., all releases) can't easily be separated from the mathematics required to arrive at the data. Calculated methods, however, such as percent of normal, are easily separated from the hydrologic data (e.g., precipitation) to which the method is applied.

Other Datatype Definitions
The following definitions are not for "types" of datatypes, as are the preceding three, but as they do impact the representation of the datatype in hdb_datatype, they are worth mentioning.

Compound Datatypes
A compound datatype is one where a time-aggregating method class is applied to a datatype which has already had a method applied (either time-aggregation or calculation) in some *other* way; the result always represents a larger interval than the source data represented. For instance, when a daily maximum air temperature is averaged over the month, the result is a compound datatype. Note that for all compound datatypes, it is important to know the interval of the source data. In the above example, *daily* maximum air temperatures, averaged over the month, might produce a different value than *hourly* maximum air temperatures averaged over the month. So, for compound datatypes, the compound_ interval must be specified. Different compound_ intervals, for the same kind of source data and with the same aggregating method being applied, result in *different* datatypes.

To restate, a compound datatype is one which has had 2 or more different time-aggregating method classes applied to it, or which has had a time-aggregating method class applied to a calculation method class. A calculation method class applied to a time-aggregating method class does *not* result in a compound datatype as defined here, as the calculation method class always operates within a single interval; the interval of the result determines the interval of the source data, hence the compound_interval need not be specified.

Note that the Derivation Application may need to functionally treat these datatypes as compounds. In other words, it must ensure that the source data is computed before the data which depends on it.

As an example of a compound datatype, see Row 14 in the datatype table. (Daily maximum air temperatures, averaged). Note that the two methods applied are different – first maximum, then average. An average applied to an average would not result in a compound datatype; rather, it would just be the same averaged datatype at a larger interval.

Cumulative Datatype

A cumulative datatype is any datatype representing a value which has been accumulated since some specified starting time. For instance, precipitation accumulated since the beginning of the water year is a cumulative datatype. When the reset point for the accumulation is at the start of an HDB business interval (such as water year) that interval can be defined in the cumulative_interval field of the hdb_datatype table; in this way, the meaning of the datatype is without question. If the accumulation reset point is not on a business interval, it can be noted as a comment in hdb_site_datatype.

Note that all cumulative values are, by definition, instantaneous readings. If, for business reasons, a cumulative value needs to be stored in an interval table, the end-of-period method must be applied to the instantaneous values.

For an example, see Row 7 in the datatype table.

Statistical Datatype

A statistical datatype is any datatype which represents a time-aggregation (with a time-aggregation method) of multiple data points representing *the same time interval* (wc), over some period of record, into a single data point. A statistical data point is never associated with a specific point in time (February 15, 2002 21:18:02) but rather with an index that indicates the interval which the value is associated with. For instance:

- A value representing February 15 over multiple years would be stored with a day-of-year index of 46. (table r_dayofyearstat)
- A value representing the 15^{th} day of the month over multiple months would have a day-of-month index of 15. (table r_dayofmonthstat)
- A value representing the 6pm hour of the day, over multiple days, would be stored with an hour-of-day index of 18.

For any of these values to be meaningful, the period of record must be known.

Note that currently, all statistics tables in HDB are assumed to be for data aggregated over more than a single year, which is why there is only one “day” table, r_daystat, which is used to hold day-of-the-year data. If day-of-the-month data was needed, a new table would have to be created to hold it.

Note also that the schema for representing the "index" to the value may change. Day of year may need to begin with March 1, to handle leap years; also, a character string may be stored to make searching for "March 20" easier.

For an example, see Rows 9 and 10 in the datatype table.

Statistics datatypes will not be stored in hdb_datatype, but rather in another (as yet uncreated) table, hdb_statistic_datatype.

Derivation vs. Calculation

The Derivation Application takes one type of data, applies some sort of method to it, and gets data (of the same datatype, or a different one) *at a higher interval*. The calculated datatypes take 2 or more types of data from the same interval, apply some sort of method or operation to them, and get data *at the same interval*. The first process is derivation; the second is calculation. Graphically, you might represent derivation as an up arrow (^) – moving to a higher interval, while calculation is right arrow (➔) – staying in the same interval.

For derivations, the following hold:

- only one type of data is required as source data;
- only non-arithmetic (time-aggregation) operations are used to create derivations;
- the source data is always pulled from a smaller interval than the interval to which the result applies

For calculations, the following hold:

- a calculation can show up in either a calculated method (such as percent of normal, cumulative, change) or in a computed datatype;
- multiple datatypes (2 or more) are required as source data; (percent of normal, cumulative and change are exceptions)
- currently, all datatypes are assumed to be on the same site;
- only known operators (such as +,-,*,/) -- not time aggregation methods such as min, max, and ave – are used in calculations (including computed datatypes);
- the source data is always pulled from the same interval table as the interval to which the result applies

Proposed Changes to the Schema

We believe that the following schema meets all of the requirements for datatype representation while cleanly and consistently managing the distinctions between instantaneous and interval data. The examples provided show how the requirements are met; namely, that hydrologic concepts of data are clear, and that exact calculation methods are stored with every data point.

NOTE: This representation *precludes* storing what we call *parallel time series*. A parallel time series is two sets of data for the same site, same time period, same *conceptual* type of data (e.g.

end of period) but with the exact calculation methods differing. For example, if you wanted to store Lake Mead's daily end-of-period water surface elevations for the period from Jan 1, 2000 through Dec 31, 2000, with one complete time series representing last-value-in-period end-of-period calculation, and another complete time series representing a linearly interpolated end-of-period, the following proposed representation would not allow that.

HDB_Method_Class

The method_class represents the *conceptual method* for deriving a piece of data. "EOP" means what hydrologists think of when they think of "end-of-period", without implying anything about how, exactly, that end-of-period value was calculated for a particular site. (Exact calculations are implied by the method_code, not the method_class_code.) The following table shows the classes of methods we need to include in HDB. Note that "copy" is not a "hydrologic concept", but is needed by the derivation application. "Unknown" is needed for data points loaded into the base area from external sources whose method_code is not known. In addition, some methods (plus, minus, times, per) represent strictly arithmetic operations which are (or could be) needed to define datatypes.

Method_Class_ID	Method_ Class_Name	Method_Class_Type	Comment
1	accumulation	calculated	Accumulation of data from a specified reset time.
2	average	time-agg	Average over an interval.
3	beginning of period	time-agg	Value at instant beginning the period.
4	change: delta of two values	calculated	Difference of two values, for a single site_datatype_id, representing same type of interval at different time steps.
5	derivation application copy	N/A	Needed for the Derivation Application.
6	end of period	time-agg	Value at instant ending the period.
7	flow-to-volume	time-agg	Flow converted to volume for an interval.
8	maximum	time-agg	Maximum value in period.
9	minimum	time-agg	Minimum value in period.
10	no-op	N/A	Needed in hdb_datatype for defining some instantaneous data and also computed datatypes.
11	percent of normal to date, stored as a fraction	calculated	An instant or interval cumulative value is divided by the corresponding statistics value. The statistics value represents the same site_datatype_id, but over some larger period of record.

12	percent of normal total, stored as a fraction	calculated	An instant or interval value cumulative is divided by a total statistics value. The statistics value represents the same site_datatype_id, but the average total of the accumulation period, over some larger period of record.
13	total percent of normal, stored as a fraction	calculated	A total value for an interval is divided by a total statistics value. The statistics value represents the same site_datatype_id, but the average total of the interval, over some larger period of record.
14	total	time-agg	Sum total of multiple values for same site_datatype_id, within an interval.
15	unknown	N/A	Used in r_base only.
16	7 day running average	calculated	Calculated because source and destination are from same interval; this is included just as an example; to show how this type of method would be represented.

HDB_Method

The HDB_method table represents all of the *particular algorithms* that might be used to arrive at a specific piece of data. Each particular algorithm (known as a method) is associated with a method_class. Each method_class can have multiple methods associated with it: for instance, method_class eop is associated with both the last value method and the linearly interpolated method. Many method_classes currently have only one associated method.

Method_ ID	Method_Name	Method_ Class_ID	Comment
1	Data-point accumulation of values	1	
2*	Data-point average of values	2	
3*	Time-weighted average using linear interpolation	2	
4*	Data-point beginning of period	3	
5	Data-point delta of two values	4	
6*	copy of data	5	
7*	Data-point end of period or last value in period	6	
8*	Linearly interpolated end of period	6	

9*	Data-point flow–to-volume	7	
10	Time weighted flow-to-volume	7	
11*	Data-point maximum	8	
12*	Data-point minimum	9	
13	No-op	10	
14	Fraction of normal to date	11	
15	Fraction of normal total	12	
16	Total fraction of normal	13	
17*	Data-point total or sum	14	
18	Calculation method not known	15	

- *Method previously discussed for the Derivation Application to support

Determining, for "percent-of-normal" values, what interval of statistical data to use.

The appropriate statistical interval table to use as the source for statistics data as the denominator (in a "nor" calculation), can be determined by the numerator. Consider the following examples

- Row 13 is defined as "fraction of normal to date". In the case of precipitation data, this would be calculated by dividing pcpcmlwyeop by pcpcmlwyeopave. If the numerator time series is coming from r_day, it only makes sense to pull the statistics from r_daystat. You would *not* want to take Feb 18's precip, accumulated from the start of the WY, and compare it to the precip normally accumulated (from Oct 1) at the *end* of February (which would be in r_monthstat). Likewise, if your pcpcmlwyeop values come from r_month (for instance, precip accumulated at end of Feb 2000, starting Oct 1), it makes no sense to compare these to the average precip normally accumulated by a certain day in that month (which would be in r_daystat).

  Similarly, if there were a datatype that represented precip accumulated from the beginning of the month (pcpcmlmoneop), and you were to divide it by pcpcmlmoneopave, you would always look to the same interval of statistics data as that of the source data. (Precip accumulated from Feb 1 to Feb 18, 2002, would be divided by the average precip accumulated on Feb 18, starting Feb 1.)

  Two things can be said here:
  - Cumulative divided by statistical cumulative implies a "percent-of-normal to date" reading.
  - When cumulative values are compared to statistical cumulative values, they are always compared to the statistical cumulative value from the *same interval*.

  So, "percent-of-normal to date" is a calculation that can be coded into an application and applied to any type of data with no further information.

- Row 14 is defined as "fraction of normal total". In the case of precipitation data, this would be calculated by dividing pcpcmlwyeop by pcptotave. Regardless of whether the numerator time series is coming from r_day or r_month, the statistical value must come from

r_wystat. It makes no sense to compare a value accumulated from the beginning of the water year (pcpcmlwyeop) to a value representing the average total precipitation for a given day, or month. The only measurement of interest is current wy-to-date precip, compared to what we normally get over an entire water year.

Similarly, pcpcummoneop would always be compared to pcptotave from the r_monthstat table, giving "precip accumulated from Feb 1 to Feb 18, compared to what we normally get for the whole month of February." (It would make no sense to compare pcpcummoneop to what we normally get for the entire year or water year.)

From this, we can conclude:

- Cumulative divided by statistical total implies a "percent-of-normal for the accumulation interval" reading.
- When cumulative values are compared to statistical totals, they are always compared to the statistical total for the interval which matches the accumulation interval.

So, "percent-of-normal total" is a calculation that can be coded into an application and applied to any type of data with no further information.

Note that the pcpcmlwyeop reading for September 30, compared to the pcpcmlwyeopave value for all September 30s, would essentially be the same as comparing pcpcmlwyeop for September 30 to the the pcptotave reading in r_wystat. But, this representation of the current day as a percent of the normal total for the interval is useful only on the last day of the interval; therefore, using "totave" in the denominator is more effective than using "cmleopave".

- Row 15 is defined as "total fraction of normal". In the case of precipitation data, this would be calculated by dividing pcptot by pcptotave. What makes the most sense here is for the real and statistical intervals to match: total precip for the month, compared to what we usually get, total, for the same month. However, it might also make sense to take the total for the month, and compare it to what we usually get for the year, or the water year. ("This month's total precip as a percentage of what we normally get for the whole year.") This is not all that far-fetched. However, we have decided (for now) that it is not necessary to allow this. Therefore, the representation of calculation methods does not have to be extended to include representation of the statistical interval.

  From this, we can conclude:

  - When total values are compared to statistical totals, they are always compared to the statistical total for the *same interval*.
  - Total divided by statistical total implies "percent of the normal total for ths interval.

  So, "total fraction of normal" is a calculation that can be coded into an application and applied to any type of data with no further information.

HDB_Datatype

The HDB_datatype table holds the datatypes defined in HDB. The new version of this table explicitly represents the *method class* used to arrive at each datatype. We use the method_class so that the same conceptual data (i.e., end-of-period water surface elevation) does not generate two separate datatype_ids when more than one particular method has been used to arrive at that data. This table also now makes explicit the interval of the source data used to derive compound and cumulative datatypes.

In addition, note that:

- Unless a datatype has a method_class_id indicating no-op, the method_source_datatype_id must be a valid, already- existing datatype_id in hdb_datatype. (I.e., all non-instantaneous and non-computed data is derivable from a more basic set of data.)

ID	Datatype Name	Unit ID	Type	Method Source Datatype_ID	Method Class ID	Compound Source Data Interval	Cml Interval
1	Instantaneous flow	cfs	S	1 (flow)	10 (no-op)		
2	Average flow	cfs	M	1 (flow)	2 (ave)		
3	Instantaneous Water Surface Elevation	feet	S	3 (wse)	10 (no-op)		
4	EOP Water Surface Elevation	feet	M	3 (wse)	6 (eop)		
5	Instantaneous precipitation	inches	S	5 (pcp)	10 (no-op)		
6	Total precipitation	inches	M	5 (pcp)	14 (tot)		
7	Cumulative Water Year Precipitation	inches	M	5 (pcp)	1 (cml)		wy
8	Cumulative Water Year Precipitation – End of period	inches	M	7 (pcpcmlwy)	6 (eop)		
9	% Normal Precip to Date	frac.	M	<calc>	11 (% normal)		
10	Instantaneous Air Temperature	deg. F	S	12 (airtemp)	10 (no-op)		
11	Maximum Air Temperature	deg. F	M	12 (airtemp)	8 (max)		
12	Average Daily Maximum Air Temperature	deg. F	M	13 (airtempmax)	2 (ave)	day	
13	All releases	cfs	C	<calc>	10 (no-op)		
14	Consumptive Use	cfs	C	<calc>	10 (no-op)		

The following shows a couple of potential rows in the hdb_statistic_datatype table (whose schema is by no means complete or well-examined in this example):

ID	Datatype Name	Unit ID	Method Source Datatype_ID	Method Class ID
50	Average of Cml Water Year Precipitation – End of period	inches	8 (pcpcmlwyeop)	2 (ave)
51	Average of total precipitation	inches	6 (pcptot)	2 (ave)

Note: it is not clear if datatype_ids for statistics datatypes need to be mutually exclusive from those used for other hdb_datatypes.

HDB_Computed_Datatype

The HDB_computed_datatype table represents the identifying information (names, ID, associated datatype) of computations used to create computed datatypes. For instance, there are at least 2 ways to compute consumptive use; each of these ways will have a row in the hdb_computed_datatype table.

Computation_ ID	Name	Datatype_ID	Cmnt
1	All releases: spill, bypass and power	13	
2	All releases: spill and power only	13	
3	Consumptive use: diversions – return flows	14	
4	Consumptive use: acreage * evap rate	14	

HDB_Computed_Datatype_Component

The HDB_computed_datatype_component table represents the specific information (operands and operators) which must be specified in order for a calculation application to compute the right type of data.

Computation ID	Order	Component_ Type	Component_ Token	Timestep_ Offset
1	1	datatype	39 (power release)	
1	2	datatype	1001 (bypass release)	
1	3	operator	+	
1	4	datatype	46 (spill)	
1	5	operator	+	
2	1	datatype	39 (power release)	

2	2	datatype	46 (spill)	
2	3	operator	+	
3	1	datatype	94 (diversion)	
3	2	datatype	93 (return flow)	
3	3	operator	-	
4	1	attribute	1002 (acreage)	
4	2	attribute	1003 (evap rate)	
4	3	operator	*	

Note that the structure of this table is not set in stone, nor will it be created at this time. Rather, it makes sense to wait until requirements for a calculation application are specified. It is likely, however, that the schema for this table will need to include additional supporting information for the kinds of computations being done. For instance, the timestep_offset could be used to indicate that a value should be pulled from some number of timesteps previous to or after the current one.

R_Base

The structure of r_base, the entry-point for all data coming into HDB, is as follows:

SITE_DATATYPE_ID:	The identifier for this site and datatype
INTERVAL:	The interval which the piece of data represents (see below)
START_DATE_TIME :	The beginning of the interval represented
END_DATE_TIME:	The end of the interval represented.
VALUE:	Value
AGEN_ID:	ID of Agency responsible for data
OVERWRITE:	Overwrite flag, used to tell the Derivation Application that this value overrides other values already-derived at the same interval (e.g., USGS final data). The overwrite values is always at a larger interval than the provisional value.
DATE_TIME_LOADED:	Date and time the value was loaded into HDB
VALIDATION:	Validation flag, such as a Dayfiles quality flag
METHOD_ID:	External method used to arrive at the data; can point to unknown.
COMPUTATION_ID:	External computation used to arrive at the data;
APPLICATION_ID:	Identifier representing the application which loaded the data.
COLLECTION_SYS_ID:	Identifier representing the system from which the data came (e.g., SCADA)

Use of Interval in R_Base

The interval is stored explicitly in r_base, in a separate column, instead of being deduced from the start_date_time and the end_date_time. This is for ease-of-use by HDB applications. It is simpler to have the interval specified explicitly at time of data loading, than to have the application deduce the interval by the start and end dates. Note that, currently, the valid values for interval are instant, other, hour, day, month, year, wy.

Integrity Checks to be Performed when Loading Data into R_Base

Several checks can be performed automatically, by the database, as data is being loaded into r_base. These checks will ensure the highest level of integrity in the data, ensuring where possible that there is consistency between the datatype, interval, start_date_time and end_date_time, and method_code of each piece of data.

Note that the datatype is implied by the site_datatype_id.

The proposed checks are as follows:

1. If interval = instant
    a. start_date_time *must equal exactly* end_date_time
    b. datatype must have a method_class_id which denotes instantaneous data; the method_class_codes which qualify are:
        i. instant
        ii. cumulative : accumulation of data up to an instant in time
        iii. some calculation method classes
2. If interval != instant
    a. start_date_time *must not equal* end_date_time
    b. datatype must have a method_class_id which denotes non-instantaneous data; the method_class_ids which qualify are:
        i. average: average over an interval
        ii. beginning of period: value at the beginning of the interval, representing the interval
        iii. change: change of 2 values over an interval
        iv. end of period: value at the end of the interval, representing the interval
        v. maximum: max value for the interval
        vi. minimum: min value for the interval
        vii. some calculation method classes
        viii. tovol: volume of flow over an interval
3. Copy is never a valid method_class_id for datatypes in r_base.
4. There must be some sort of integrity check between the method_class_id (implied by the site_datatype_id), and the method_id. (If method_id does not represent 'unknown', then method_id must have method_class_id consistent with the datatype.)

Note: We will enforce the interval/method_class_id checks proposed above to the best of our ability. It may not be possible to do the checks for method classes that do calculations. (See questions at end of document.)

The gist is ➔ if you're loading interval data into the base area, the method_class_id must reflect what the data represents *with respect to that interval*.

Example 1

Given the rules above, a value being loaded into r_base which represents the midnight water surface elevation reading at Lake Mead could be stored in one of 2 ways:

SDI / method_class_id	Interval	Start Date Time	End Date Time	Value	Method_ID
4000 / wse	instant	01-feb-02 00:00:00	01-feb-02 00:00:00	1924	12 (no-op)
1930 / wseeop	day	31-jan-02 00:00:00	01-feb-02 00:00:00	1924	7 (eopdata)

The point is, the way the value is loaded into r_base is up to the user, determined by how they want to use the value. If an entire time series of instantaneous water surface elevation readings is available, and the user would like to use them in calculating averages, etc., then the first row might make sense. (The Derivation App spec would provide the proper window to select the appropriate – close to midnight – value, for the eop value.) If only the eop value is of concern, it might seem simpler or more intuitive to mark the initial value as an end of period, rather than instantaneous, reading (the second row).

Example 2

Here we present sample rows in hdb_site and hdb_site_datatype to support the following time series data examples for r_base and r_day. Neither hdb_site or hdb_site_datatype are under review for restructuring.

HDB_Site

Site_ID	Site_Name
921	Lake Mead
748	Colorado River Below Hoover Dam

HDB_Site_Datatype

Site_Datatype_ID	Site_ID	Datatype_ID
99	921	3
100	921	4
199	748	1
200	748	2

So, 100 = Lake Mead's end-of-period water surface elevation, 99 is the instantaneous reading; 200 = Colorado River Below Hoover Dam's average flow, 199 is the instantaneous flow.

R_Base

SDI	Interval	Start Date Time	End Date Time	Value	Method_ID
99	instant	02-Jan-2002 23:33	02-Jan-2002 23:33	121.8	12 (no-op)
99	instant	02-Jan-2002 23:57	02-Jan-2002 23:57	123.2	12 (no-op)
					12 (no-op)
99	instant	05-Jan-2002 23:40	05-Jan-2002 23:40	124.7	12 (no-op)
99	instant	05-Jan-2002 23:55	05-Jan-2002 23:55	125.0	12 (no-op)
200	hour	01-Jan-2002 00:00	01-Jan-2002 01:00	800.80	2 (avedata)
200	hour	01-Jan-2002 04:00	01-Jan-2002 05:00	800.84	2 (avedata)
200	hour	01-Jan-2002 08:00	01-Jan-2002 09:00	800.90	2 (avedata)
200	hour	01-Jan-2002 12:00	01-Jan-2002 13:00	800.94	2 (avedata)
200	hour	01-Jan-2002 16:00	01-Jan-2002 17:00	801.00	2 (avedata)
200	hour	01-Jan-2002 20:00	01-Jan-2002 21:00	801.40	2 (avedata)

R_Day

All of the real time series tables have been extended to support the Derivation Application and the Database of Record concept. The following version of r_day shows all relevant columns (some columns not pertinent to this discussion are left out), with sample rows based on the data above.

Site_Datatype_ID	Start_Date_Time	End_Date_Time	Value	Method_ID
100	01-Jan-2002 00:00:00	02-Jan-2002 00:00:00	122.8	7 (eopdata)
100	02-Jan-2002 00:00:00	03-Jan-2002 00:00:00	123.2	7 (eopdata)
100	03-Jan-2002 00:00:00	04-Jan-2002 00:00:00	123.8	8 (eoplint)
100	04-Jan-2002 00:00:00	05-Jan-2002 00:00:00	124.4	8 (eoplint)
100	05-Jan-2002 00:00:00	06-Jan-2002 00:00:00	125.0	7 (eopdata)
100	06-Jan-2002 00:00:00	07-Jan-2002 00:00:00	124.8	7 (eopdata)
200	01-Jan-2002 00:00:00	02-Jan-2002 00:00:00	800.98	2 (avedata)
200	02-Jan-2002 00:00:00	03-Jan-2002 00:00:00	803.1	2 (avedata)
200	…	…		2 (avedata)
200	18-Sep-2002 00:00:00	19-Sep-2002 00:00:00	1100.9	2 (avedata)
200	19-Sep-2002 00:00:00	20-Sep-2002 00:00:00	1098.5	2 (avedata)
200	20-Sep-2002 00:00:00	21-Sep-2002 00:00:00	1093.6	3 (avetwlint)
200	21-Sep-2002 00:00:00	22-Sep-2002 00:00:00	1095.0	3 (avetwlint)

200	…	…		3 (avetwlint)
200	30-Dec-2002 00:00:00	30-Dec-2002 00:00:00	902.5	3 (avetwlint)
200	31-Dec-2002 00:00:00	31-Dec-2002 00:00:00	846.4	3 (avetwlint)

The Lake Mead date (SDI=100) shows what the data would look like if a couple of days of eopdata data were missing and had to be filled in through linear interpolation.

The Colorado Below Hoover data shows what the data would look like if average flows were calculated with a data-point average method until September 20, and then the calculation method was switched to a linearly interpolated time weighted average.