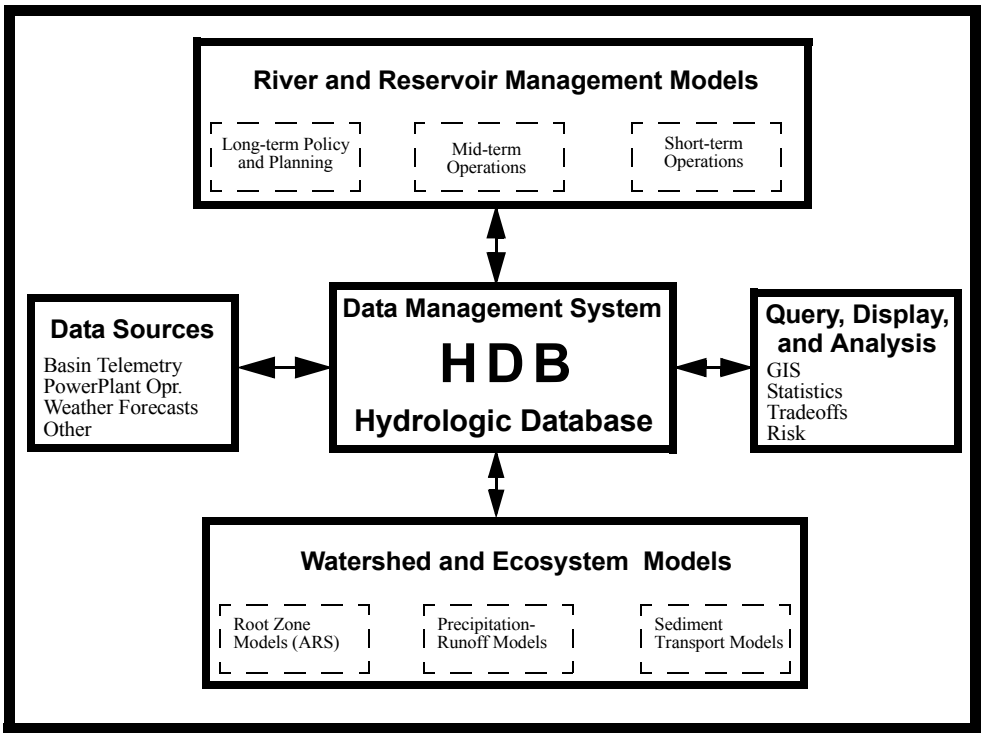
Hydrologic Database (HDB): HDB Schema User Guide

Next Page



The Center for Advanced Decision Support for Water
and Environmental Systems (CADSWES)
University of Colorado at Boulder
College of Engineering and Applied Science
Department of Civil, Environmental and Architectural Engineering
Campus Box 421
Boulder, CO 80309-0421
Phone: (303) 492-3972 Fax: (303) 492-1347
E-mail: inquiries@cadswes.colorado.edu
http://cadswes.colorado.edu

CADSWES

Section 1 Intended Audience

This document is intended for any user of HDB, or its applications, who desires an understanding of:

- the underlying schema, including tables, columns, triggers, procedures, constraints and permissions;
- naming conventions;
- design concepts for complex portions of the schema, including examples of use.

Next Page

Previous Page

Section 2 Application Overview

The Hydrologic Database (HDB) is an Oracle-based relational database which supports the storage of river basin data. The database was designed with flexibility and extensibility in mind:

- The data storage schema is generic , that is, it is not tied to the operations or structure of any particular organization.
- New types of time series data and new sites can be stored without any impact to the schema.
- Because HDB is a data-centered (rather than application centered) project, new data storage, retrieval and processing applications can easily be added to the system.
- With the addition of the XML data loading application, data from any source can easily be stored in HDB.
- HDB is essentially a non-distributed database, meaning it can run as a standalone data server in any organization.
- The Meta Data Application, and the concept of a data supersystem (comprised of a master and one or more snapshot databases), makes it possible to use HDB where data must be shared between two or more organizations.

Next Page

Previous Page

In general, the Hydrologic Database holds three types of data:

1. *Time series data* is divided according to the interval it represents (i.e., daily, monthly, etc.), and whether it is observed or modeled data.

2. Physical characteristic data is (in general) divided according to the type of object the data pertains to (i.e., reservoir, diversion, etc.).

3. Meta data is data used to identify data in the database, and to help external applications access HDB. It is divided according to whether or not it is installation-specific, or global to all coordinated HDB installations.

Section 3 Permissions

Next Page

Previous Page

Section 4 Invoking the Application

There is no one interface from which to access all of HDB. Rather, a few applications exist to give access to varying parts of the database. Following is a brief guide to HDB data accessing and maintenance applications.

Next Page

TABLE 1.HDB Data Access Applications

Application	Purpose
Brio Query	**Query and view time series data**
Hydrologic Update Facility (HUF)	**Maintain time series data. This application is accessed from Brio.**
Meta Data Application	**View and maintain HDB meta data.**
SQLPlus	**View and maintain all HDB data. For advanced users only.**
Oracle Designer	**View and maintain HDB Schema**

Previous Page

Section 5 Inputs

Command Line

Next Page

Files

Previous Page

Dialog Boxes

Database Driver Tables

<high level description only, with pointer to technical_reference>

Section 6 Outputs

Data Files

NA

Next Page

Log and Error Files

Previous Page

NA

Database

The database schema is documented via Oracle Designer. Specific information on tables, columns, triggers, procedures, constraints and permission can be found in the Designer Repository reports published on the HDB Users' Web Page.

Section 7 Using Application Functionality

Introduction

This section is used to explain standards and complex design concepts in HDB which impact the user.

Next Page

Naming Conventions

Previous Page

A set of naming conventions has been developed for HDB to provide user friendliness, and also to simplify the process of extending the database (either its schema or its contents) when appropriate. These conventions encompass not only database objects (e.g., tables, columns, views), but also data stored in the database (e.g., data type names, object type names, etc.).

These conventions should be adhered to when possible to ensure the most consistent naming scheme possible.

Note: New conventions should be documented as they are established.

Objects to Which Conventions Apply

- databases
- tables
- views
- columns
- data types
- object types

Next Page

Previous Page

Databases

Each separate installation of HDB has a unique name. The name is of the format:

<office>hdb

where <office> is a short identifier (2 - 4 letters) of the office in which the database resides (e.g., UC, LC or YAK).

We currently have LCHDB, UCHDB and YAKHDB.

Note: Capitalization is unimportant. No underscores ("_") are used in database names.

Tables

Consistency in database table names is essential. If tables in HDB are well-named, it is easy to determine what data is stored in the table, and to intuitively determine the name of a table if you know the type of data it contains.

General Table Naming Conventions

There are several kinds of tables within HDB and slightly different naming conventions apply to each. There are some general conventions which apply to all table names:

- All table names are singular (e.g., "div", not "divs");
- Table names are constructed of various components (which will be described specifically for each table type). Underscores ("_") are used only to delineate components. They are not used within components.

- Table names use modifiers. In addition, Modifiers are used in constructing data type names. Modifiers are explained in more detail in this section.

Time Series Data Tables

Components

1. Source of data

Time series data is either real (based on measurements taken at a physical site) or model-generated. The codes used to represent source of data in time series data tables are:

Next Page

- real: "r"
- model generated: "m"

Previous Page

2. Time affiliation of data

Each time stamp, or time index associated with a time series value indicates the time with which the value is associated.

2.1 Instantaneous/interval data: Data that is associated with a *date*. The possible time affiliations are:

- instantaneous: "instant"
- other: "other"
- hourly: "hour"
- daily: "day"
- monthly: "month"
- yearly: "year"
- water yearly: "wy"

2.2 Statistics data: Data that is associated with a *time index*. (**See"Date or Time Index Column Names" on page -17** for valid index ranges.)

- hour of day: "hour"
- day of year: "day"
- month of year: "month"

Note: The day of the year index can apply to either calendar years or water years however, values indexed by day of calendar year must be stored separately from values indexed by day of water year. Consequently, if 'day of water year' values become important, a new data table (e.g., r_daywystat) must be created to hold these values.

Note: A day of month index may also become desirable. If it does, it will be necessary to create another table for day of month statistics.

3. Modifier for time series data

Statistics and range data (see ??) are the only time series data requiring a modifier for the table name.

- statistics: "stat"
- range: "range"

They can be used together, for instance, in naming a table that holds statistical range data.

Note: Time series tables hold all types of data (e.g., flow for stream gages, contents of reservoirs, etc.) so the type of data does not play a part in the table name.

Next Page

Previous Page

Naming Convention

The convention for naming time series data tables is:

<source>_<time affiliation>[<modifier>]

There is never a "_" between a modifier and the component which precedes it, nor between modifiers.

Examples

- m_month: holds model-generated data associated with monthly intervals
- r_hour: holds real, observed data associated with hourly intervals
- r_instant: holds real, observed data where each value is associated with an instant in time
- r_monthstatrange: holds statistical monthly range data arrived at from examining several years of monthly range data
- r_hourstat: holds statistical values computed for every hour of the day arrived at from examining several days, months, or years of real data.

Attribute and Physical Characteristic Data Tables

Attribute and physical characteristic data tables hold values which are not measured or computed, but which are inherently associated with the site (i.e., physical characteristics). Many physical characteristics are a single value per site (e.g., elevation). Other physical characteristics may vary by time of year (e.g., month) or by time of day (e.g., hour of the day). The following naming convention takes these situations into account.

Components

1. Reference table indicator "ref": Indicates that this table is not synchronized with the data of any other HDB installation rather, the data is pertinent to, and owned by the local installation.
2. Object type: Some attributes are common to all object types. These attributes are stored in the non-type, specific site table (hdb_site). Some attributes apply only to objects of a specific type; in this case, the object type must be evident in the table name. No abbreviations, underscores, or plurals will be used in identifying object types within table names. (?? **See"Object Types" on page -18** for a list of valid object type abbreviations.)
3. Time affiliation of data: The unit of time for which the attribute is effective, only if the attribute varies with time. (**See"Date or Time Index Column Names" on page -17** for valid index ranges.)
   - day of year: "day"
   - hour of day: "hour"
   - month of year: "month"

Next Page

Previous Page

Naming Convention

The convention for naming physical characteristic and attribute tables is:

ref_<object type>[_<time index>]

If the data does not vary with time (e.g,. if the attribute is a single value), it will be stored in the object type table (ref_res, for instance). Any attributes which do vary with time will be stored in an object-specific table with the appropriate time index (e.g., ref_hyd_month).

Examples

ref_res: holds all single value attributes for reservoirs (i.e., dead storage, area, etc.)

1. ref_hyd_month: holds (or would hold) all attributes for hydro plants which vary by month (i.e., baseflow).

Coefficient Tables

Coefficient tables manage any site-specific coefficients. Coefficients are stored at the installation responsible for the site. Different types of objects can share a coefficient table. A different coefficient table exists for each time interval over which a coefficient is effective (e.g., daily coefficients, monthly coefficients, etc.).

Components

The components of a coefficient table name are:

1. Coefficient prefix, including reference table ("ref") indicator. Like physical characteristic tables, the data in coefficient tables is not shared between installations. It is completely local.
2. Time index of coefficient: the unit of time for which the coefficient is effective, only if the coefficient varies with time. (**See"Date or Time Index Column Names" on page -17** for valid index ranges.)
   - day of year: "day"
   - hour of day: "hour"
   - month of year: "month"

Next Page

Previous Page

Naming Conventions

The convention for naming coefficient tables is:

hdb_site_coef_<time index>

Examples

ref_site_coef: holds coefficients which do not vary with time (i.e., discharge, tailwater, etc.) for all object types.

ref_site_coef_month: holds coefficients which vary by month (i.e., reservoir evaporation) for all object types.

Lookup Tables

Lookup tables manage site-specific data where one value is dependent upon some other value measured at the site. The dependent value is "looked up" using the supplied measured value. For example, given the water surface elevation at some reservoir, the volume of water in storage can be looked up using the elevation volume table for the reservoir.

Naming Components

1. Reference table indicator ("ref").
2. Object type: The lookup tables are object-type specific The object type must be indicated in the table name. (?? **See"Object Types" on page -18** for Object Types.)
3. Datatype code: Only the measured (independent) data type should be indicated in the table name. Any number of variables whose values depend on the measured value may then be stored in the table. For example, in the current elevation volume table, only the code for water surface elevation is used in the table name (hdb_res_wselu), although area, content,

release and spill can all be looked up based on the water surface elevation.

Note: Note that datatype codes are no longer a standard part of the datatype representation in HDB. However, they are still used as needed, as they are in naming lookup tables.

4. Lookup modifier: All lookup tables end with the modifier "lu".

Naming Convention

The convention for naming lookup tables is:

Next Page

hdb_<object type>_<measured data type code>lu

Previous Page

Examples

hdb_res_wselu: lookup table for all reservoir elevation versus <data type>pairs.

Meta Data Tables

Every meta table should be immediately identifiable by its name. All meta table names will use a 3-letter prefix which indicates that it is a meta (i.e., reference or lookup) table.

Several meta-data tables will operate under the supersytem model, where one office (e.g., Upper Colorado) is the master location for the meta data, and any coordinated HDB installations have read-only copies of the data. All updates, additions, etc. to the data take place only on the master database. (See the Meta Data User Guide for more information on this protocol.) When the database in question is an island (not part of a supersystem), the data in those tables is not shared, but it is also not office-specific.

The data in all remaining meta data tables is office-specific. For example, while both UC and LC have meta data tables to support the Hydromet data-loading applications, the data in these tables is not identical across the UC and LC installations; site codes and pcodes differ between the regions. (The previously-mentioned physical characteristic, coefficient and lookup tables fall into the office-specific category of meta data tables.)

Naming Conventions

- The names of all meta tables which operate under the supersystem model will be prefixed with 'hdb_' to indicate that there is a single master copy of the data which is shared throughout all coordinated subsystems of HDB. This convention applies on island installations also, where the data is not actually shared.
- The names of all meta tables which hold installation-specific data will be prefixed with 'ref_' to indicate that this is office-

specific meta (or reference) data.

A few other conventions should be adhered to in naming meta tables. Because the content of meta tables varies widely, the conventions are not entirely strict.

- Keep table names short, intuitive, without unnecessary underscores, and without cryptic abbreviations.
- For meta tables which are used only by a specific application, follow the "ref_" with a brief indicator of the application (plus an underscore), and use the application abbreviation/name consistently for all meta tables specific to that application. For instance, all tables which hold Hydromet meta data might begin with "ref_hm_".

Next Page

Previous Page

Examples

hdb_agen: lookup table for all agencies referenced in HDB. Agencies are coordinated using the supersystem protocol.

ref_hm_site: lookup table for all Hydromet site codes used in the Hydromet applications.

Note: The "hm" is an application indicator.

Summary Notes

If these table naming conventions are followed, all tables in any HDB subsystem should begin with:

- "r_"
- "m_"
- "hdb_" or
- "ref_"

Views

All views will follow the naming conventions for tables explained above, with the addition that each view name will be suffixed with "_v". For instance, a view on all single value site attributes would be called "hdb_site_v". Views that join elements of various tables should be named accordingly.

Columns

The following conventions should be used in naming table

columns:

- Use only singular column names (i.e., "value", not "values"). The one exception to this is if the value in the column can actually represent a plural. For instance, a column which holds the number of units at a hydro power plant may well have the value 4 and should probably be called "nounits", not "nounit".
- Do not prefix column names with the table name or some abbreviation of it, unless this is necessary to make the column name unique within the table.
- Use underscores in column names only to delineate the various components of the name.
- Be consistent. Do not reinvent the wheel. If you need to create a new table; and if a column exists elsewhere in the database which represents an attribute which you must incorporate in the new table, name the new column the same as the existing one. (Re-phrased: columns which result from foreign key relationships should be named the same as the column which is the key source.) The exception to this is when a different name is required to make clear the meaning of the column as it is used in the table.
- If the column will hold values for a specific data type or attribute (e.g., reservoir dead storage, power plant capacity, etc.), the column should be given the name of the assigned code for the data type or attribute. For instance, a column which holds water surface elevation values should be called "wse".

Next Page

Previous Page

Note: Datatype codes are no longer used in HDB. Any time series "datatype" that needs to be used in a column name is really functioning as an attribute, and should be added to the HDB_Attr table and given a code.

- If the column can hold values of any data type (e.g., if the table has a datatype_id or site_datatype_id column to indicate the type of the corresponding value), then the column holding the data value , should be named "value".
- Columns which hold time indicators (either dates or time indexes) must be named consistently so that accessing applications always know the name of the time column.
- The name of the time column must clearly indicate whether the value will be a date (e.g., '03-JAN-1994'), or an index (1 - 12).

Date or Time Index Column Names

For actual dates:

All tables holding time series data must have both a begin and end date to indicate the entire interval represented by the value. Single date/time columns (such as date_hour, date_day, etc.) are included for backward compatibility with existing applications, but will eventually be phased out.

The appropriate column names for begin and end dates, regardless of the interval represented, are:

Next Page

start_date_time

end_date_time

Previous Page

For date indices:

The following table shows the appropriate date-index column name given the time component represented. Date indices are used in statistical series tables.

TABLE 2. Date column names for date indices

Time Component Represented		Column Name
hour of the day (1 - 24)	**use**	**hour**
day of the year (1 - 366)	**use**	**day**
month of the year (1 - 12)	**use**	**month**

For date ranges or date index ranges:

For columns which must indicate date ranges or date index ranges, prefix the column name with "start_" or "end_". For instance, monthly range values will have date columns of "start_date_month" and "end_date_month". Monthly statistical range values will have date columns of "start_month" and "end_month":

Note: The "day of the year" index can be used for both calendar and water year; however, values which represent day of the water year must be stored in a table separate from those values representing day of the calendar year (e.g., not r_daystat) (**See " Naming Convention" on page 11.**)

Data Types

A data type in HDB corresponds to a type of information held in the database. Applications manipulating time series data can easily identify datatypes using the explicit supporting attribute information in the HDB_Datatype table. However, it is important that users of HDB can easily recognize a datatype by the text name

given to it. Each HDB datatype has two names:

- datatype_name: The full name of the datatype. This name should be as detailed and explicit as possible, leaving no question as to the meaning of the datatype. Conventions to adhere to in assigning the full name:
    - If the data is instantaneous, indicate this clearly in the datatype_name.
    - If the data represents an interval, indicate clearly the type of aggregation (beginning or end of period, average, maximum or minimum, sum total, accumulation, etc.) used in aggregating the data.

Next Page

- datatype_common_name: The display name of the datatype. This name should be shortened for user-friendliness, as it will be used as the display name of the datatype in interfaces to the data.

Previous Page

Datatype names should *not* include:

- References to specific sites or objecttypes (unless, as with water surface elevation, the data can only be associated with a single type of object).
- References to specific time intervals, unless some sort of interval association is implied by the data. For instance, if a datatype represents an accumulation of precipitation since the beginning of the water year, it is important to indicate this in the datatype name. Similarly, if the data is an average of daily maximum air temperatures, it is important to indicate the affiliation with the daily interval in the datatype name.

Object Types

Each object type has a database tag (e.g., "str" for a stream gage object) as well as a full name. Only the tags will be used to refer to specific object types in table or column names. All existing tags are three characters and will not incorporate underscores.

HDB object types and database tags identified to date are:

- acoustic velocity meter: "avm"
- basin: "bas"
- canal: "can"
- climate site: "cli"
- confluence: "con"

- diversion: "div"
- hydro power plant: "hyd"
- hydro power plant unit: "unt"
- reach: "rch"
- reservoir: "res"
- snotel site: "sno"
- stream gage: "str"
- water quality site: "wtrql"

Next Page

Previous Page

For new object types added to HDB in the future, the tags should not incorporate underscores, and they should not use cryptic abbreviations. If possible, the database tag should be three characters in length, but must first be unique. Both object type names and database tags should use all lower-case letters. Objecttype tags cannot exceed 5 characters.

Site Names

Unfortunately, there is not one definitive source of "official" site names for physical sites monitored by the Bureau of Reclamation. For sites added in the future, the following rules should be followed in naming the site.

- Determine the official site name used by the agency responsible for the site.
- Use first-letter capitalization rules. Sites are proper names. Do not use all capital letters!

Note: The site name will merely be attribute data; it will not be used to reference the site anywhere within the database. However, consistent site names will be easier to read and sort throughout the life of HDB. Official USGS stream gage and climate site names contain state names as well. This is necessary to fully identify the site, and the state name should be included in the site_name field in HDB.

Other Data

The following miscellaneous conventions apply to other data stored in HDB.

- Use first-letter capitalization rules for all proper names (e.g., sites, user names, data source names, state names, etc.)
- All other names (e.g., unit names, data type names, time

step names, etc.) should use lower-case.

- All table and column names which are stored as data should use lower case. Oracle stores its own catalog data (concerning tables and columns) in upper case; if joins are necessary between Oracle catalog tables and HDB, string manipulation will be necessary. Convert either the HDB or the Oracle value using the upper or lower SQL*Plus string manipulation function.

The following table shows miscellaneous abbreviations which have been adopted.

Next Page

Naming Conventions — Miscellaneous Abbreviations

Table: 3

Previous Page

Word	Abbreviation
attribute	**attr**
coefficient	**coef**
index	**idx**
interval	**intv**
number	**no**
water year	**wy**

Instantaneous and Interval Data

There are 2 fundamental categories of data in HDB.

Instantaneous Data

This is 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.

Interval Data

This is 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.)

Next Page

Previous Page

It is up to each installation to decide which data is instantaneous and which is interval. One guideline for distinguising instantaneous and interval data is this: if we know the start and end times, and they're different, the value must be an interval value.

Instantaneous and Interval Tables in HDB

All categories of data (instant, other, and regular interval) are stored in individual 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.

Two tables, r_instant and r_other, exist despite the fact that they hold non-aggregated data which can be found in r_base. The reason for having these tables is that all data 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, and it is important to have instantaneous and other data available in that quality-checked form.

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.

Special Kinds of Interval Data

In addition to the "standard" interval data, that is, data which represents a defined interval such as an hour, day, month, etc., there is also statistics and range data. Both of these kinds of data are affiliated with an interval; they are described below.

Next Page

Statistics Data

Previous Page

Statistics data are computed over a number of years (e.g., for a site's entire period of record). For instance, a monthly statistic could be computed by averaging the values for each month of the year over multiple years. In this way, you would calculate a "January average", a "February average", etc. These resulting averages are associated with a time index, such as "month of the year", rather than a specific date. That is, the value is meaningful "for all Januaries".

Statistics can be calculated using any aggregating method (e.g., ave, min. or max), and for any time index. Therefore, a statistics table could hold maximum values by day of year, or minimum values by hour of the day.

Note: To be completely accurate, the representation of statistical data must include a descriptor of the period of record used as source data for the statistic. Currently, this information is not included in the statistical data representation of HDB.

Range Data

Range data are time series data which represent not a single business interval, but a specific range of consecutive business intervals. April through July volumes (of interest for the runoff season) are the most common example of range data. Range values can be associated with either a specific start and end date (01-APR-1999, 01-JUL-1999), or they can be statistics associated with multiple years, in which case a start month and end month would be used to index the data (i.e., 4 and 7 for April and July). The latter is an example of statistical range data.

Datatypes

Overview of Representation

Advantages

The representation of datatypes in HDB is a fairly rigorous

representation with the following advantages:

- Users can 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, or whether it was interpolated to fall exactly at the end of the period.
- Users can 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?
- Users and applications can:
  - look at a datatype and see how it is generated;
  - know how to generate the datatype given the source data.

  Therefore, the integrity of the data which supports the derivation application is guaranteed, and the entering of derivation specifications is highly automated based on information provided in the datatype representation.
- The datatype's type (simple, method-applied, computed, or a combination) is explicitly stored, indicating where to look for more information about the datatype.
- Compound datatypes are handled specifically. If a datatype represents an aggregation of an already-aggregated piece of data (for instance, the average of daily maximum temperatures), the representation specifies that the source data is daily. This eliminates the question (left by earlier representations) as to what interval of data was used as the source.
- Cumulative datatypes are handled specifically. The interval over which a value is accumulated is specified in the datatype definition, leaving nothing to the imagination.
- Computed datatypes can be represented thoroughly using a Reverse Polish Notation paradigm.

Next Page

Previous Page

Inclusions and Exclusions

- Data types are independent of the unit in which data is represented. For example, the data type "air temperature" does not denote a temperature in degrees Centigrade. Rather, it represents a temperature measurement, the units for which are specified elsewhere in the data type definition.
- Data types are not specifically tied to a single object type. The datatype "instantaneous flow" can be used for a diversion,

a stream gage, or for any other type of object for which this data is measured. This convention makes it possible to reduce the number of datatypes in HDB.

- Data types are never tied to a single site. It would be inappropriate to create a datatype for the water surface elevation of Lake Powell; rather, this data would be described with a datatype of water surface elevation, and it would be associated with Lake Powell through a site_datatype_id.

Definitions

Next Page

This section defines the concepts integral to the representation of datatypes and data in HDB. This information is intended as a supplement to the schema documentation provided in the Appendices. ??

Previous Page

Method Classes

The following definitions are for the "types" of method classes (as represented in the method_class_type field of hdb_method_class).

Time Aggregation Method Class

A time 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: average, beginning of period, end of period, maximum, minimum, sum over time, to volume.

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 related datatype at the same interval table. Percent of normal readings (a current reading 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 of the accumulation interval 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 proposed

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 (N/A) Method Classes

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

Operator

Next Page

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

Previous Page

Datatype Types

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 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. Simple datatypes always represent instantaneous data.

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.

Computed datatypes are defined using a Reverse Polish Notation in the HDB_Computed_Datatype table.

Next Page

Other Datatype Definitions

Previous Page

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.

Next Page

Previous Page

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 unit (e.g., day), 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 15th 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.

Ultimately, statistics datatypes will not be stored in HDB_Datatype, but rather in another (as yet uncreated) table, HDB_Statistic_Datatype.

Hdb_Datatype: Use of Columns

Next Page

Previous Page

This section is intended to augment the column descriptors supplied in Oracle Designer.

Method Class ID

This column indicates the method class which would be used, either outside HDB or by the Derivation Application, to generate this datatype from its source data. If the data is simple or computed, use N/A. Note that the particular algorithm is not indicated by the method class; rather, the hydrologic concept of the operation performed (e.g., end of period) is implied.

Method Source Datatype ID

This column indicates the source data for the method-applied datatype being represented. Select the datatype_id to which the method_class will be applied to arrive at the datatype you are defining. For instance, for "average flow", use the datatype_id for "instantaneous flow" as the method_source_datatype_id.

Compound Interval

For compound data only. This is the interval of source data to be used in generating the compound data. For instance, if you're creating average daily maximum temperatures, source_data_interval = "day". Daily maximums will be used to create the average values over another, larger interval (i.e., month).

Cumulative Interval

The interval for which accumulations and percent-of-normal values apply. Typically water year ("wy") but can be anything. Note that many datatypes previously termed "totals" (e.g., total precipitation) are really accumulated datatypes. For most such datatypes, the appropriate cumulative_interval is "table interval," which indicates that the data is accumulated from the beginning of the interval represented by the table holding the cumulative data. So, the cumulative_interval for total precipitation data might be "day" if it's accumulated from the beginning of the day. This is in

contrast to water year accumulations, which are typically recorded on a daily basis and stored in the r_day table.

Business Rules

This section identifies *conceptually* the constraints that are in effect when defining a new datatype. For a complete list of the foreign keys, triggers, and procedures which go into implementing these constraints, see the HDB Administration document. ???

Next Page

Note: Data which is in violation of these constraints will always be caught at the database level. However, the MetaData Application is implemented in such a way that the user cannot even attempt to enter violating data for the majority of these constraints. Therefore, we recommend using the MetaData Application to maintain this data.

Previous Page

- Unit_id must be present in hdb_unit table.
- Datatype_type must be present in hdb_datatype_type table.
- Method_class_id must be present in hdb_method_class table. Use N/A for datatypes which are simple or computed.
- Method_source_datatype_id must be present as a datatype_id in hdb_datatype. This field can only be filled in if the datatype_type includes "method-applied".
- Compound_interval must be present in hdb_interval table, and can only be filled in when the datatype is a compound datatype.
- If compound_interval is not null, cumulative_interval must be null, and vice-versa. (A datatype cannot be compound and cumulative.)

Percent-of-Normal Data

This section describes how the various percent-of-normal methods should be calculated by a calculation application. Specifically, this section describes how to determine what interval of statistical data to use in the calculation: The appropriate statistical interval table to use as the source for statistics data as the denominator, can be determined by the numerator.

Next Page

Fraction of normal to date accumulation

Previous Page

In the hdb_method table on ???, the 15th row is for "fraction of normal to data accumulation. In the case of precipitation data, this would be calculated by dividing precipitation accumulated from the beginning of the water year (the current water year's value) by average precipitation accumulated from the beginning of the water year, for many water years (the statistical value). 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 precipitation, accumulated from the start of the water year, and compare it to the precipitiation 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 precipitation accumulated from the beginning of the month, and you were to divide it by average precipitation accumulated from the beginning of the month, you would always get statistics data from the same interval 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.

From this, we can conclude:

- 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*.

Fraction of Normal Total for Accumulation Period

In **Table 6, "HDB_Method," on page 45**, the 16th row is "fraction of normal total for accumulation period". In the case of

precipitation data, this would be calculated by dividing precipitation accumulated from the beginning of the water year (the current water year's value) by average total precipitation accumulated over the entire water year, for many water years (the statistical value). 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 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.

Next Page

Similarly, precipitation accumulated from the beginning of a month would always be compared to average total precipitation accumulated over the entire month, for many of these months, giving, for example, "precipitation 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 a partial February accumulation to what we normally get for the entire year or water year.)

Previous Page

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.

Fraction of Normal for the Interval

In the hdb_method table on ???, the 17th row is "fraction of normal for the interval". In the case of precipitation data, this would be calculated by dividing precipitation accumulated over the entire interval by average precipitation accumulated over the entire interval, for multiple intervals. What makes the most sense here is for the real and statistical intervals to match: precipitation accumulated 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 accumulated precipitation 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, at this time, 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 this interval.

Next Page

Previous Page

Time Series Data Tables

Business Rules for R_Base

This section identifies *conceptually* the constraints that are in effect when inserting a row into r_base. These checks 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_id of each piece of data. For a complete list of the foreign keys, triggers, and procedures which go into implementing these constraints, see the HDB Administration document.

Next Page

Previous Page

Note: Note that the datatype is implied by the site_datatype_id.

The proposed checks are as follows:

1. If the timestamp on the value is more than 2 hours ahead of the current time, do not allow the value into r_base. (Future values cannot be loaded into r_base.)
2. If the inserted value is marked as an overwrite value, there cannot also be a source derivation specification for this site_datatype_id and interval.
3. If interval equals instant
   - **3.1** start_date_time *must equal exactly* end_date_time
   - **3.2** datatype must be instantaneous data; the ways to determine this are:
     - **3.2.1** the method_class is N/A
     - **3.2.2** the method_class is accumulation
     - **3.2.3** the method_class is a percent-of-normal method_class (there are 3) and the denominator is instantaneous data
4. If interval does not equal instant
   - **4.1** start_date_time *must be less than* end_date_time
   - **4.2** datatype must have a method_class_id which denotes non-instantaneous data (the converse criteria of 1.2 above)
5. Copy is never a valid method_class_id for datatypes in r_base.

6. If method_id for this data point does not represent 'unknown', then verify that the supplied method_id has the same method_class_id as the datatype implied by the site_datatype_id. This means, for instance, that an end of period elevation value cannot have a method_id for data point average.

Next Page

Previous Page

Parent Site and Parent Objecttype Data

The parent_site_id field is used to identify multiple HDB sites which are located at the same geographic (physical) location. While HDB is not intended to fill the role of a geographic information system (GIS), knowing which sites are in the same physical place is often useful.

Each site in HDB can have one associated parent site. The parent site is the site which:

Next Page

1. is located at the same geographic location as its child, and
2. is of the objecttype which has the highest ranking in the parent order defined in hdb_objecttype.

Previous Page

For any group of sites at the same geographic location, all sites must have the same parent; a site will be its own parent if it is of the objecttype highest in the ranking.

Setting Parent Site and Parent Object type IDs

As of now, only a few sites have their parents defined in HDB. These sites are climate and snotel sites which reside at the same geographic location; the climate site is always the parent. When defining the parent sites for other sites in HDB, do the following for each geographic location:

1. Determine all of the sites which are at that location.
2. Determine the object type of each of these sites.
3. Determine which of these object types is highest in the parent order:

   SELECT objecttype_id FROM hdb_objecttype

   WHERE objecttype_parent_order = (SELECT max (objecttype_parent_order)FROM hdb_objecttype);

This will be the parent_objecttype_id for all sites at this geographic location.

4. Of all of the sites at this location (Step 1) determine which one is of the objecttype highest in the ranking for this geographic location (Step 3). This site is the parent_site_id for all sites at this geographic location.
5. Set all of the parent_objecttype_id and parent_site_id values to those ID values obtained in Steps 3 and 4, respectively. The parent site will have itself as a parent site.

Note: Because basins are not point features, they do not rest at the same geographic location as any other sites. Hence, they cannot be given parent site or parent object type IDs.

Setting the Parent Order for New Object Types

If a new type of object is added to HDB, in the HDB_Objecttype table, it must be placed in the parent order. If the object type goes last in the parent order, then no other changes need to be made: the addition will not affect the parent sites of any existing sites in HDB_Site.

Next Page

If, however, the new object type needs to be placed in the middle of the objecttype_parent_order, then the following steps must be taken:

Previous Page

1. For every object type which will now be lower in the parent ranking, increase the rank by 1 (that is, add one to the current objecttype_parent_order value). If an objecttype_id did have an objecttype_parent_order of 8, it should now be 9.
2. Insert a record for the new object type and assign its proper parent rank.
3. For every object type (referred to from here on as lower_objecttype_id) which is now lower in the parent ranking than the new object type (new_objecttype_id):

    3.1 Determine if any objects of this "lower" type are parents in hdb_site, and note the site_ids returned by the query:

    *SELECT * FROM hdb_site WHERE parent_objecttype_id = <lower_objecttype_id>;*

    3.2 If the above query returns any rows, determine if any sites of the **new** object type are located at the same place as sites returned in 3a:

        3.2.1 SELECT site_id FROM hdb_site WHERE objecttype_id = <new_objecttype_id>;

        3.2.2 Manually compare resulting set of site_ids (Set B) to those site_ids from Step 3a (Set A), and determine if any sites in Set B are at the same geographic location as sites in Set A. If so, these Set B sites become the parents of the Set A sites.

        3.2.3 If needed, reset each parent_site_id and parent_objecttype_id of the sites in Set A which have a parent in Set B:

3.2.4 UPDATE hdb_site SET parent_site_id = <corresponding Set B site_id>,parent_objecttype_id = <new objecttype_id> WHERE site_id = <Set A site_id which has new parent>;

Next Page

Previous Page

Unit Representation

Introduction

In the interest of simplifying the unit representation in HDB, as well as ensuring that its conversion results more closely match those of Riverware, HDB has adopted a unit representation similar to that used in Riverware.

Next Page

This representation is based on having:

Previous Page

- a set of unit dimensions, such as flow, length, volume, time, etc.;
- a set of units for each dimension, such as acre feet, cubic inches, etc. for volume;
- a single stored unit for each dimension, i.e., the unit which serves as the reference unit for conversions between units in this dimension, such as acre feet for volume; and
- a set of conversions, to go from the stored unit for a dimension to each unit within that dimension.

Unit Conversion

The Unit Conversion Factor

Most unit conversions are represented as simple multiplication factors; e.g., what factor is required to go from cubic feet per second (the stored unit) to acre feet per day (an alternate unit). The following table shows pertinent columns in the HDB_Unit table for units in the flow dimension.

Conversion factors can be used to go from the non-stored unit to the stored unit simply by taking the reciprocal of the factor.

Table: 4

unit_id	unit_name	dimension_id	stored _unit_id	month_year	over_month _year	is_factor
46	**acre feet per day**	**2**	**2**			**1**
43	**acre feet per month**	**2**	**2**		**M**	**1**
47	**acre feet per year**	**2**	**2**		**Y**	**1**

Table: 4

unit_id	unit_name	dimension_id	stored _unit_id	month_year	over_month _year	is_factor
2	**cubic feet per second**	**2**	**2**			**1**
9	**cubic meters per second**	**2**	**2**			**1**

Scaling Non-Constant Time-Dependent Units

Next Page

Previous Page

In the above table, the conversion from CFS to AFD is represented as a constant factor, 1.983 (roughly). This makes sense: the number of acre feet in a cubic foot is constant, and the number of days in a second is constant. The conversion from CFS to AFM is also represented as a constant factor, 61.488 (roughly). But the number of months in a second (or vice-versa) is not constant.This factor is based on a 31-day month, and when values representing non-31-day months are being converted, the factor is scaled according to the number of days really in the month. The same thing applies to the unit AFY. This year-dependent conversion factor is based on a 365 day year; in leap years, the conversion factor is scaled.

This scaling is why the month_year and over_month_year fields are needed: to tell the converter when and how to do its scaling.

Conversion Expressions

Conversion expressions are used only when absolutely necessary. Converting an expression is more time-consuming, and more conversion information must be stored in the database.

The only dimension for which expression conversion is required, so far, is temperature. This is because temperatures are not zero-based values; you have to add and subtract 32’s and 459.67’s to convert between temperature units.

Unlike factors, the reciprocal of an expression cannot be used to convert from a non-stored unit to a stored unit. Hence, expressions must be stored for doing the conversion in both directions. (This was preferable to writing an algebraic expression converter to solve the equation in the other direction.)

Data Validation Flags

The validation field of HDB real time series tables can hold two types of information:

- one-character data quality flags, which come from the source of the data (e.g., Hydromet dayfiles);
- one-character external validation flags, indicating the results of any external validation procedures which might be run on some time series data.

Next Page

Quality flags from the data source are more important than external validation flags; any data which has a quality flag will not, with its initial loading, receive a flag which represents external validation procedures.

Previous Page

However, few sources provide data quality flags, and, in those cases, external validation can be performed, to verify that values fall within a specified range. Such validation procedures will be defined as part of the Database of Record project.

For validation flags which are data quality indicators from outside HDB, two processing options are available.

1. Such rows could be ignored by the validation application; or
2. the site_datatype_id could be checked, and, if it is one which requires external validation, the validation could be performed and the quality flag overwritten. These are semantics which can be determined at the time the validation application is implemented.

Default Date Format

Next Page

Previous Page

This section describes how to work with dates in HDB. Most date handling is made invisible through data query and maintenance applications, but users of SQL*Plus do need to understand date formatting concepts.

Unless the date which you supply in an insert, update, delete or select statement is of the default format that Oracle has been set to recognize, you must call a format function and supply a format string to tell Oracle in what format you're presenting your date.

For HDB, we have chosen to set the parameter NLS_DATE_FORMAT to "DD-MON-YYYY". This implies that all dates which are queried without explicit formatting will be presented in this format (e.g., "01-JAN-1999"). Also, this is the only format that is accepted in inserts, updates and deletes without using the to_date function to specify the format. For instance, the statement:

insert into r_day values (2021, '10-mar-1999', 2752, 5, 'Z');

will be accepted by Oracle, because the format of the data matches the default set in NLS_DATE_FORMAT. It will insert the value with a date of 10-mar-1999 00:00:00 (hh/mm/ss portion is automatically set to 0 if none is supplied).

But the statement

insert into r_hour values (2021, '10-mar-1999 13:00:00', 2752, 5, 'Z');

will not be accepted, as the date format is longer than the default. Use of the to_date function is required here:

insert into r_hour values (2021,

to_date ('10-mar-1999 13:00:00', 'dd-mon-yyyy hh24:mi:ss'), 2752, 5, 'Z');

The to_char function can be used query dates from HDB and display the hh/mm/ss portion:

select site_datatype_id,to_char (date_hour,'dd-mon-yyyy hh24:mi:ss'),value from r_hour;

Obviously, the NLS_DATE_FORMAT is more conducive to working with daily and monthly values, not hourly or instantaneous. If, within a session, you will mainly be querying dates for which you need the hour, you can set your default date format for the session:

alter session set nls_date_format 'dd-mon-yyyy hh24:mi:ss';

Throughout your session, all dates that are in this format will not

require formatting, and all dates queried will be displayed in this format.

Next Page

Previous Page

Section 8 Examples

Datatypes

Next Page

Previous Page

This section gives sample data for all the tables involved in representing datatypes in HDB. Along with the sample data, descriptions are given to assist the user in choosing datatype attributes appropriately.

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 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.

TABLE 5.HDB_Method_Class

Method_Class_Name	Method_Class_Type	Comment
accumulation	calculated	Accumulation of data from a specified reset time.
average	time-agg	Average over an interval.
beginning of period	time-agg	Value representing beginning of period.
change	calculated	Difference of two values, for a single site_datatype_id, representing same type of interval at different time steps.
copy	N/A	Derivation Application copy.
end of period	time-agg	Value representing end of period.
to-volume	time-agg	Conversion of flow to volume or volume to volume for the interval.

TABLE 5.HDB_Method_Class

Method_Class_Name	Method_Class_Type	Comment
maximum	time-agg	Maximum value in period.
minimum	time-agg	Minimum value in period.
N/A	N/A	For datatypes which do not have a method associated with them (simple or computed)
percent of normal to date accumulation, 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.
percent of normal total for the accumulation period, stored as a fraction	calculated	An instant or interval value cumulative is divided by the statistical normal total value for the instant or interval. The statistics value represents the same site_datatype_id, but the average total of the accumulation period, over some larger period of record.
percent of normal for the interval, stored as a fraction	calculated	Non-accumulated value for the interval divided by the statistical normal non-accumulated value for the same interval.
sum over time	time-agg	Sum of multiple values for same site_datatype_id, within an interval.
Method not known	N/A	Used for unknown_method in r_base only.

Next Page

Previous Page

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 have only one associated method.

TABLE 6.HDB_Method

Method_Name	Method_Class_Name
Data-point accumulation of values	accumulation
Data-point average of values	average
Time-weighted average using linear interpolation	average
Data-point beginning of period	beginning of period
Data-point delta of two values	change
copy of data	copy
Data-point end of period or last value in period	end of period
Linearly interpolated end of period	end of period
Data-point flow–to-volume	to volume
Time weighted flow-to-volume	to volume
Volume to volume	to volume
Data-point maximum	maximum
Data-point minimum	minimum
N/A	N/A
Fraction of normal to date accumulation	% of normal to date cml, stored as fraction
Fraction of normal total for accumulation period	% of normal total for the cml period, stored as fraction
Fraction of normal for the interval	% of normal total for the interval, stored as fraction
Data-point sum over time	sum over time
Unknown	method not known

Next Page

Previous Page

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. The following table gives examples of most unusual cases of datatype definitions that might arise. The identifiers in this table are used in subsequent tables referencing HDB_Datatype.

TABLE 7.HDB_Datatype

ID	Datatype Name	Unit ID	Type	Method Class	Method Source Datatype	Cpd Int	Cml Int
1	Instantaneous flow	cfs	S	N/A	<NULL>		
2	Average flow	cfs	M	ave	1 (flow)		
3	Instantaneous Water Surface Elevation	feet	S	N/A	<NULL>		
4	EOP Water Surface Elevation	feet	M	eop	3 (wse)		
5	Instantaneous precipitation	inches	S	N/A	<NULL>		
6	Total precipitation	inches	M	cml	5 (pcp)		
7	Cumulative Water Year Precipitation	inches	M	cml	6 (pcptot)		wy
8	Cumulative Water Year Precipitation – End of period	inches	M	eop	7 (pcpcmlwy)		
9	% Normal Precip to Date	frac.	M	% norm to date	8 (pcpcmlwyeop)		
10	Instantaneous Air Temperature	deg. F	S	N/A	<NULL>		
11	Maximum Air Temperature	deg. F	M	max	10 (airtemp)		
12	Average Daily Maximum Air Temperature	deg. F	M	ave	11 (airtempmax)	day	
13	All releases	cfs	C	N/A	<NULL>		
14	Consumptive Use	cfs	C	N/A	<NULL>		

ext Page

Previous Page

Notes:

Row 7: Even though this data may come into HDB already accumulated, it should be broken down to the most detailed possible representation: conceptually, this data can be derived from daily total precip readings. Note that this is instantaneous data, even though the readings are probably once daily.

Row 8: EOP is used to derive instantaneous accumulated readings (cumulative or percent-of-normal), up into interval data. Think of this as converting an instantaneous reading into the reading which represents the hour, day, etc. This is not a compound datatype, as the source data method (accumulation) generates instantaneous data.

Row 12: Daily maximums averaged over a month will produce a different value than hourly maximums averaged over that same month. Therefore, it is important to represent that the source data is daily.

Next Page

Statistics Datatypes

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):

Previous Page

TABLE 8.HDB_Statistic_Datatype

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

TABLE 9.HDB_Computed_Datatype

Computation_ ID	Name	Datatype_ID
1	All releases: spill, bypass and power	13
2	All releases: spill and power only	13

TABLE 9.HDB_Computed_Datatype

Computation_ ID	Name	Datatype_ID
3	Consumptive use: diversions – return flows	14
4	Consumptive use: acreage * evap rate	14

HDB_Computed_Datatype_Component

Next Page

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. Note that the component_token IDs used in this example do not correspond to the datatype_ids in the above HDB_Datatype table.

Previous Page

TABLE 10.HDB_Computed_Datatype_Components

Comp.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.

Next Page

Previous Page

Time Series Data

User Choices in Loading R_Base

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:

TABLE 11. Loading R_Base, User Choices

SDI / datatype	Interval	Start Date Time	End Date Time	Value	Method
4000 / instant. wse	instant	01-feb-02 00:00:00	01-feb-02 00:00:00	1924	N/A
1930 / eop wse	day	31-jan-02 00:00:00	01-feb-02 00:00:00	1924	Data point EOP

Next Page

Previous Page

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).

Complete R_Base / R_Day Example

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.

TABLE 12.HDB_Site

Site_ID	Site_Name
921	Lake Mead
748	Colorado River Below Hoover Dam

TABLE 13.HDB_Site_Datatype

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

Next Page

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.

Previous Page

TABLE 14.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	N/A
99	instant	02-Jan-2002 23:57	02-Jan-2002 23:57	123.2	N/A
					N/A
99	instant	05-Jan-2002 23:40	05-Jan-2002 23:40	124.7	N/A
99	instant	05-Jan-2002 23:55	05-Jan-2002 23:55	125.0	N/A
200	hour	01-Jan-2002 00:00	01-Jan-2002 01:00	800.80	avedata
200	hour	01-Jan-2002 04:00	01-Jan-2002 05:00	800.84	avedata
200	hour	01-Jan-2002 08:00	01-Jan-2002 09:00	800.90	avedata
200	hour	01-Jan-2002 12:00	01-Jan-2002 13:00	800.94	avedata
200	hour	01-Jan-2002 16:00	01-Jan-2002 17:00	801.00	avedata
200	hour	01-Jan-2002 20:00	01-Jan-2002 21:00	801.40	avedata

TABLE 15.R_Day

SDI	Start_Date_Time	End_Date_Time	Value	Method_ID
100	01-Jan-2002 00:00:00	02-Jan-2002 00:00:00	122.8	eopdata
100	02-Jan-2002 00:00:00	03-Jan-2002 00:00:00	123.2	eopdata
100	03-Jan-2002 00:00:00	04-Jan-2002 00:00:00	123.8	eoplint

Section 8 Examples

TABLE 15.R_Day

SDI	Start_Date_Time	End_Date_Time	Value	Method_ID
100	04-Jan-2002 00:00:00	05-Jan-2002 00:00:00	124.4	eoplint
100	05-Jan-2002 00:00:00	06-Jan-2002 00:00:00	125.0	eopdata
100	06-Jan-2002 00:00:00	07-Jan-2002 00:00:00	124.8	eopdata
200	01-Jan-2002 00:00:00	02-Jan-2002 00:00:00	800.98	avedata
200	02-Jan-2002 00:00:00	03-Jan-2002 00:00:00	803.1	avedata
200	…	…		avedata
200	18-Sep-2002 00:00:00	19-Sep-2002 00:00:00	1100.9	avedata
200	19-Sep-2002 00:00:00	20-Sep-2002 00:00:00	1098.5	avedata
200	20-Sep-2002 00:00:00	21-Sep-2002 00:00:00	1093.6	avetwlint
200	21-Sep-2002 00:00:00	22-Sep-2002 00:00:00	1095.0	avetwlint
200	…	…		avetwlint
200	30-Dec-2002 00:00:00	30-Dec-2002 00:00:00	902.5	avetwlint
200	31-Dec-2002 00:00:00	31-Dec-2002 00:00:00	846.4	avetwlint

Next Page

Previous Page

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.