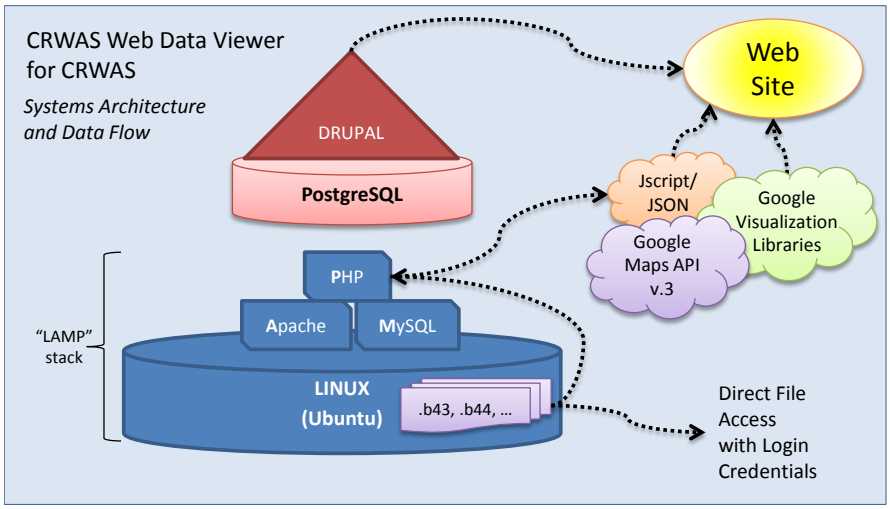
Technical Memorandum – Final

To:	Ray Alvarado (CWCB), Blaine Dwyer (AECOM)
From:	Kelly Close (Leonard Rice Engineers, Inc.)
Subject:	CRWAS Phase I – Task 2.9 Public Data Posting – Documentation of Subtask 1 (Binary File Reader) and Subtask 2 (Metadata)
Date:	February 28, 2012

Introduction

CRWAS Phase I included scope (Task 2.9) to develop an online Data Viewer to allow user-friendly public access to CRWAS results. Subtasks 1 and 2 of the Data Viewer scope included a deliverable that documents a binary file reader and metadata associated with development of the Data Viewer.

A dedicated database and web server has been configured to house the CRWAS Data Viewer system. This server has its own fixed IP address, where users of the system can interact with the web site via a domain name dedicated for the project. The illustration below shows a conceptual layout of the server and the flow of data through the system.



The CRWAS Data Viewer has been developed in Drupal on a standard "LAMP" stack Linux server (also supported by a PostgreSQL database server, a number of custom Javascript, PHP, and HTML files) and calls to Google API and visualization libraries (for Data Viewer maps and graphs). The Drupal software provides a secure web site framework supporting user accounts and convenient built-in page layouts. The PostgreSQL database includes custom scripting to process user requests, store user selections, and contain model metadata that can be used in the Drupal site. PHP scripts read the StateMod output files and display data in graphs using Google visualization libraries with custom Jscript code. HTML code combines the individual components with embedded Javascripting and calls to the PHP scripts and the Google API and visualization libraries to display map pages and graph displays.

This memo describes the technical specifications of the server machine and the server contents. It outlines the contents the PostgreSQL database and describes each custom script and code file.

Server Specifications

IP Address: 64.25.233.142

Dedicated Domain Name: www.dataviewer.info

Google Apps account: USER NAME: crwas@dataviewer
PASSWORD: Crw@s_google

Storage Locations of Data, Code, and Scripts

Model output files are stored on the server are located in /data/CRWAS/ModelData.

Custom code and scripts used in the web interface are located in /var/www/CRWAS and include:

- ***gmap_api.html***: Displays a Google Map with KML overlays showing selectable model nodes for a specific (user selected) basin.
- ***statemod_reader_graph.html***: Displays final report graphs.
- ***cdss_reader.php***: Called by the statemod_reader_graph file to pull user specified content into report graphs.
- ***gvds_pgtable.php***: Called by Jscript embedded in the drupal site that displays a complete list of model nodes from all basins. This file converts the data pulled from PostgreSQL into JSON formatted data for use by the google API libraries.

Additional custom content not part of the Drupal site:

- ***statemod_param_descrips.htm*** *(and associated folder)*: Displays in the parameter descriptions popup.
- ***traces_disclaimer.htm*** *(and associated folder)*: Displays in the traces disclaimer popup.
- ***climate_model_descrips.htm*** *(and associated folder)*: Displays in the climate model descriptions popup.

Backup scripts run each night to back up the server contents and are located under home/administrator/bin/backup_*. Backup files are stored on the server as well as on an external device connected to the server:

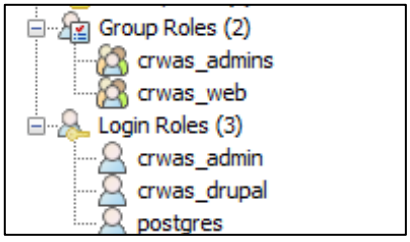
- /data/backups/… (on the server)
- /media/lre250raid1/backups/… (on a connected external device)

PostgreSQL Database Contents and Organization

Users:

The PostgreSQL Database is named WVToolCRWAS, and in addition to the standard “postgres” user login, includes two login users and two group roles. These additional logins keep the database contents more secure, allowing non-super user access by the external processes (Drupal, Javascript, and PHP). Login credentials for the three PostgreSQL users include:

User: postgres Password: Crw@s_pgsql
User: crwas_admin Password: crwasadmins
User: crwas_drupal Password: crwasdrupal

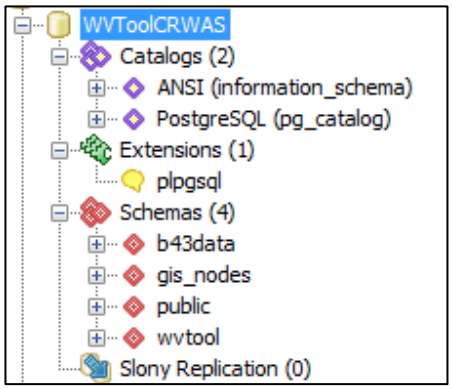


All external processes connecting to the database use the *crwas_drupal login*, which belongs to the *crwas_web group*. The *crwas_web group* has connect and usage privileges only on the database and schemas and only specific permissions on tables, views, and functions needed to support the external processes.

The *crwas_admin user* belongs to the *crwas_admins group* and owns the database and all contents. This is a non-super user with full rights throughout the database so it can be used to administer the database without allowing super user database configuration rights. The *postgres user* has super user privileges to the database server.

Schemas

The database is organized into four schemas. The standard "public" schema is not used.



The "b43data" schema contains the model file metadata read in from the header of each type of model file (currently b43, b44, and xdd). **If the model output file formats change, these tables should be rebuilt.** Also included in this schema are several metadata tables (see next section) and the view used by the PHP reader code to display selected series for the report graphs.

The "gis_nodes" schema includes a metadata table that associates model nodes with districts and basins and provides alternate names for nodes that do not have adequate naming in the model files themselves (see next section).

The "wvtool" schema includes remaining data and logic supporting the Data Viewer web site, including several metadata tables (see next section), user selection tables, views, and functions. An entity relationships diagram of the database is included in **Appendix A** detailing more of this schema's contents and how it relates to the other schemas and to the overall Data Viewer processing.

Metadata Tables

The PostgreSQL database stores information about the models and model nodes beyond information stored in the model output files themselves. These data make it possible to display more information for the user in the web site so users that are less familiar with the models can still use the site with ease. The metadata tables are listed here by database schema name and table name (schema.table):

- ***b43data.basin_models***: Lists the five basins by name and includes the model file name abbreviation used in the mode output file naming conventions.
- ***b43data.climate_models***: Includes the list of all climate models (11) by scientific name and common description and the name of the folder on the server where model files for those climate models are located. The table also includes a sorting field for ordering the climate models in lists in the web interface.
- ***b43data.statemod_output_filetypes***: Lists the filetypes that can be processed by the PHP reader code and describes them. It also includes filename convention information needed to find specific files based on user selections.
- ***b43data.statemod_runs***: Comprehensive table that displays all available combinations of basins and climate models. Though currently, all basins include the same set of climate models, this table will support a different set of climate models in different basins.

- ***gis_nodes.districts_lookup***: Comprehensive list of all node ID's from all model files that associates them with a district number and a basin. It also includes an alternate name for the node that will override the name in the model file when the node is displayed in the web interface.
- ***wvtool.climate_models_to_basins***: Associates each climate model listed in the *climate_models* table with a basin listed in the *basin_models* table.
- ***wvtool.list_district_names***: List of districts represented in the models and a name for each district, for display in the district drop-down box in the web interface.
- ***wvtool.list_parameter***: List of all parameters available in any of the .b43, .b44, or .xdd files and which parameter should be displayed in the web interface for which node type. An alternate name for the parameter may also be specified. **This table should only be edited through the drupal web interface (with an administrator login).**

Appendix A includes more detailed documentation of database tables, views, and functions.

Drupal Site Administration

The Data Viewer user interface was developed in Drupal 6.22. Drupal is a popular, open source content management framework. The Drupal site may be administered by logging into the site with admin login:

User Name: admin
Password: Crw@s_drupal

This login provides access to all administration menus, pages, and settings familiar to a Drupal programmer. Most of the Drupal site has been built with either core functionality or with the addition of some very popular add-in modules available and documented on the drupal.org website. Add-in modules installed on the Data Viewer Drupal site include:

- admin_menu
- cck
- date
- block_edit
- better_perms
- permission_select
- front (front page)
- iquery_ui
- node_clone
- nodeaccess
- quicktabs
- google_analytics

In addition, two custom modules have been added to the site to support the unique PostgreSQL interaction functionality of the CRWAS Data Viewer. These include a utilities module (*wvtool_utils*) which contains custom functions that are called from code in Drupal pages to interact with the PostgreSQL database; and Fetchit2, a custom module developed by Leonard Rice Engineers, Inc. to provide integrated support in Drupal for connecting to an external database (in this case PostgreSQL).

Please see **Appendix B** for details about *wvtool_util* and **Appendix C** for details about Fetchit2.

APPENDIX A

PostgreSQL Database Annotated Entity Relationships Diagram (ERD)

The CRWAS Data Viewer system includes a PostgreSQL database that houses data needed to support the Data Viewer that cannot be pulled from the StateMod binary files. This includes metadata about the models and a host of functions and views which support the Drupal site and PHP scripts. This document includes Entity Relationship Diagrams for the database objects plus explanation and annotation to help fully document the Data Viewer database and is organized as follows:

1. Binary File Metadata ([b43data] schema)
2. Node Lists ([gis_nodes] schema and portions of the [wvtool] schema)
3. Parameter Lists (portions of [wvtool] the schema)
4. Basin, District, and Climate Model Lists (portions of the [wvtool] schema)
5. User Selections (objects and code in [wvtool] schema that store and process user selections)
    - Map Selections
    - Adding and Modifying Selections
    - Removing Selections

The following page includes a key for symbols used throughout the ERDs. Table A1 includes a list of all database objects cross-referenced with the corresponding Figure number for the ERD containing each object.

ERD SYMBOL KEY

The database is organized into three schemas. While these schemas are largely self-contained, there are some relationships that cross schemas. The help the reader understand the cross-schema relationships, the three database schemas are color coded:

- GREEN background for the [b43data] schema,
- BLUE background for the [wv_tool] schema and
- PURPLE background for the [gis_nodes] schema.

Database TABLES within each schema are shown as boxes with the same color coding.

PostgreSQL Views in all schemas are represented as 3D light orange boxes.

PostgreSQL Functions are dark blue.

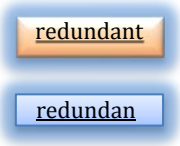
Primary Key fields in tables are noted with this symbol.

Flow of data is shown with thin blue arrows

Processes that modify database contents with code (either internal to the database by PostgreSQL functions or external by PHP scripts) are noted with heavy red arrows.

Red circles identify database objects which are either used by or modified by the Drupal interface or PHP scripts.

The External code in Drupal and PHP scripts directly using or affecting the database objects are shown as light blue flags.



Some tables and views are repeated on multiple ERD's. These are shown in detail only the first time they appear. On subsequent diagrams they are shown without the fields listed and with a blue glow.

TABLE A1. DATABASE TABLES, VIEWS AND FUNCTIONS CROSS-REFERENCED WITH ERD FIGURES

Database Object	Object Type	ERD Figure(s)
append_selections(integer)	function	A7
append_selections_quick_builder(integer)	function	A7
delete_selections(integer)	function	A8
district_filter_post_submit(integer)	function	A7
modify_selections(integer)	function	A7 (A8)
param_admin_submit()	function	A4
rebuild(integer)	function	A7
climate_models_to_basins	table	A5
climate_models_to_groups	table	A5
list_climate_group	table	A5 (A7)
list_districts_names	table	A3 (A5)
list_parameter	table	A4 (A7, A8)
list_parameter_drupaledit_b43	table	A4
list_parameter_drupaledit_b44	table	A4
return_paths	table	A6
selected_basin	table	A3 (A3, A5, A7, A8)
selected_call_page	table	A6
selected_climate_group	table	A7
selected_climate_model	table	A7
selected_district	table	A3 (A5, A7)
selected_node	table	A6 (A7)
selected_parameter	table	A7 (A8)
selected_to_delete	table	A8
selected_trace	table	A7
selected_ts	table	A7 (A8)
selected_ts_edit	table	A7 (A8)
filtered_list_*	view	A3
list_all_nodes_for_gviz	view	A3
list_baseflows	view	A3
list_basin	view	A5
list_climate_models_by_basin	view	A5 (A8)
list_district	view	A5
list_diversions	view	A3
list_flowstations	view	A3
list_isfreaches	view	A3
list_nodes_advanced_builder	view	A3
list_parameters_*	view	A4
list_reservoirs	view	A4
master_list_base	view	A3 (A6)
master_list_final	view	A3 (A7, A8)
node_type_returnpath	view	A6
quick_builder_selections	view	A7
selected_nodes_info	view	A8

1. Binary File Metadata

The database stores information extracted from the header of the .b43 and .b44 files. This information provides the foundation for all node and parameter related information and logic. These header data are stored in the schema [b43data], in the table [b43metadata_record6x_rivernodes]. There are many other [b43metadata_...] tables in this schema that also store data from the binary file headers but they are redundant and not used by any other functions or views in the database (and not listed in this ERD).

Five additional tables in the [b43data] schema store information about the model runs that have been loaded into the CRWAS Data Viewer to date. When new model runs are to be incorporated, these tables must be modified to reflect the new data.

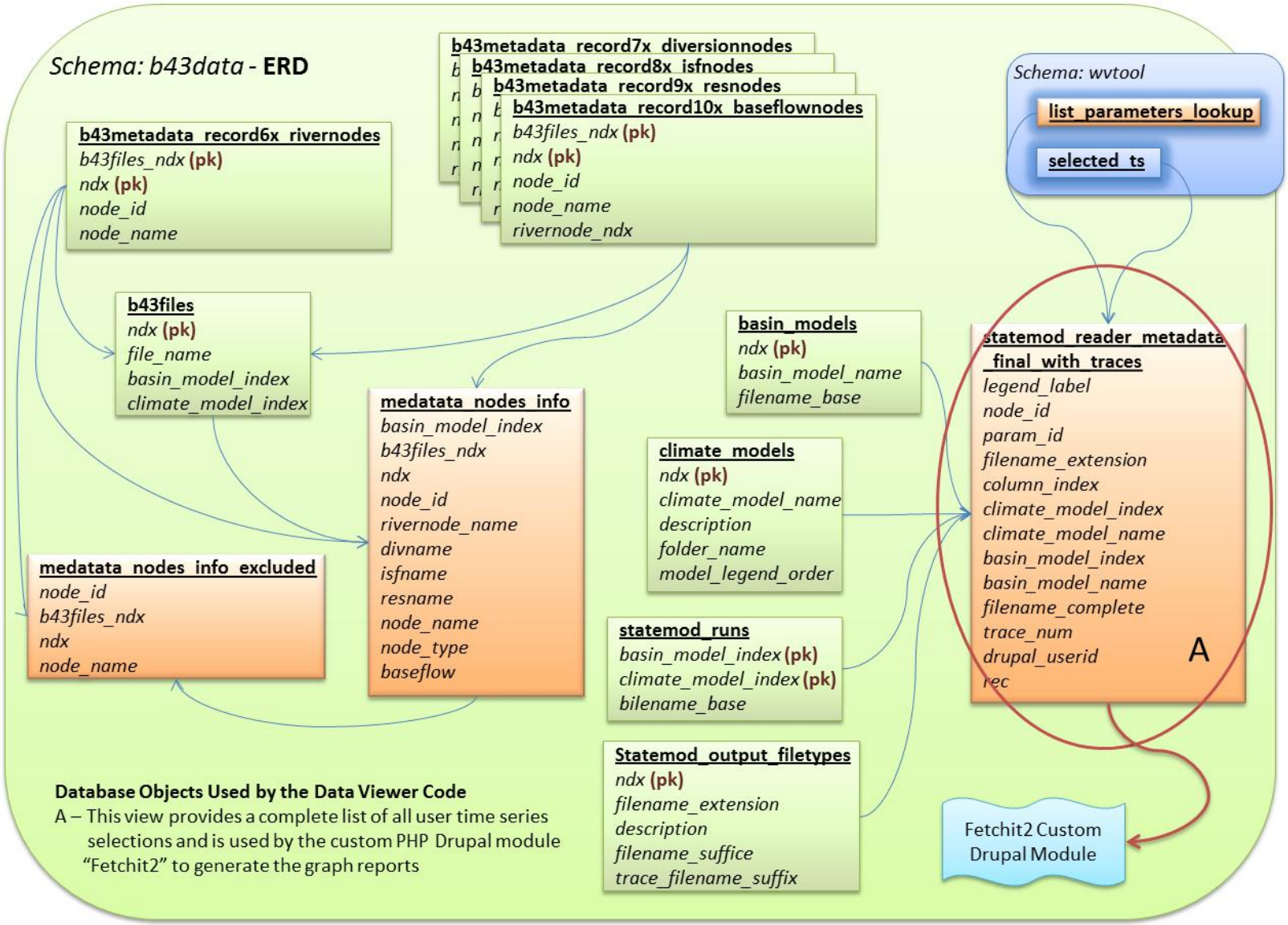
- ***[b43files]*** lists the binary files used to for header data reference. The process of loading the header data from the files listed in this table into [b43metadata-record6x_rivernodes] must be done manually.
- ***[basin_models]*** lists the basin model included in the Data Viewer.
- ***[climate_models]*** lists the climate models included in the Data Viewer (the table that associates climate models to basin models is in the [wvtool] schema and discussed below).
- ***[statemod_runs]*** is a comprehensive list of every model run available through the Data Viewer, which basin and climate model it is from and the path to where the model output files live on the server.
- ***[statemod_output_filetypes]*** provides additional critical naming convention strings for the different types of model files so that the full path to each file can be built in code at the time a user selects it.

The [b43data] schema also includes three views which are fundamental to the operations of the Data Viewer.

- ***[metadata_nodes_info]*** pulls the information from the header data (in the record6x table) and prepares it for use by subsequent views and functions.
- ***[metadata_nodes_info_excluded]*** lists the nodes that are present in the model file metadata but are not being exposed through the Data Viewer (these are generally nodes that do not represent physical locations on the ground are not baseflow nodes).
- ***[statemod_reader_metadata_final_with_traces]*** builds from information in the b43data schema AND information in the [wvtool] schema (see the next section) to provide a list of all model data sets chosen by all users for display in the Data Viewer graph reports. This list is used by the external graphing PHP scripts.

Figure A1 on the following page includes a complete Entity Relationships diagram for the [b43data] schema.

FIGURE A1. ENTITY RELATIONSHIPS DIAGRAM FOR THE [B43DATA] SCHEMA

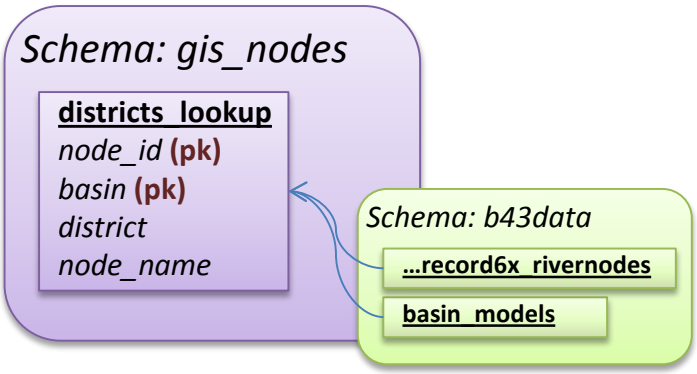


2. Node Lists

Information about model nodes starts in the [b43data] schema (see previous section) and then is passed to the [wv_tool] schema for processing in views and functions to prepare the data for the variety of ways the lists are used by the Data Viewer. Information is also stored in the [gis_nodes] schema describing the which nodes fall into which districts. **It's important to note that controlling the node information in the [b43data] and [gis_nodes] schemas is a manual process.** Once these data sets are edited correctly, the node information in wv_tool updates automatically by virtue of the functions and views.

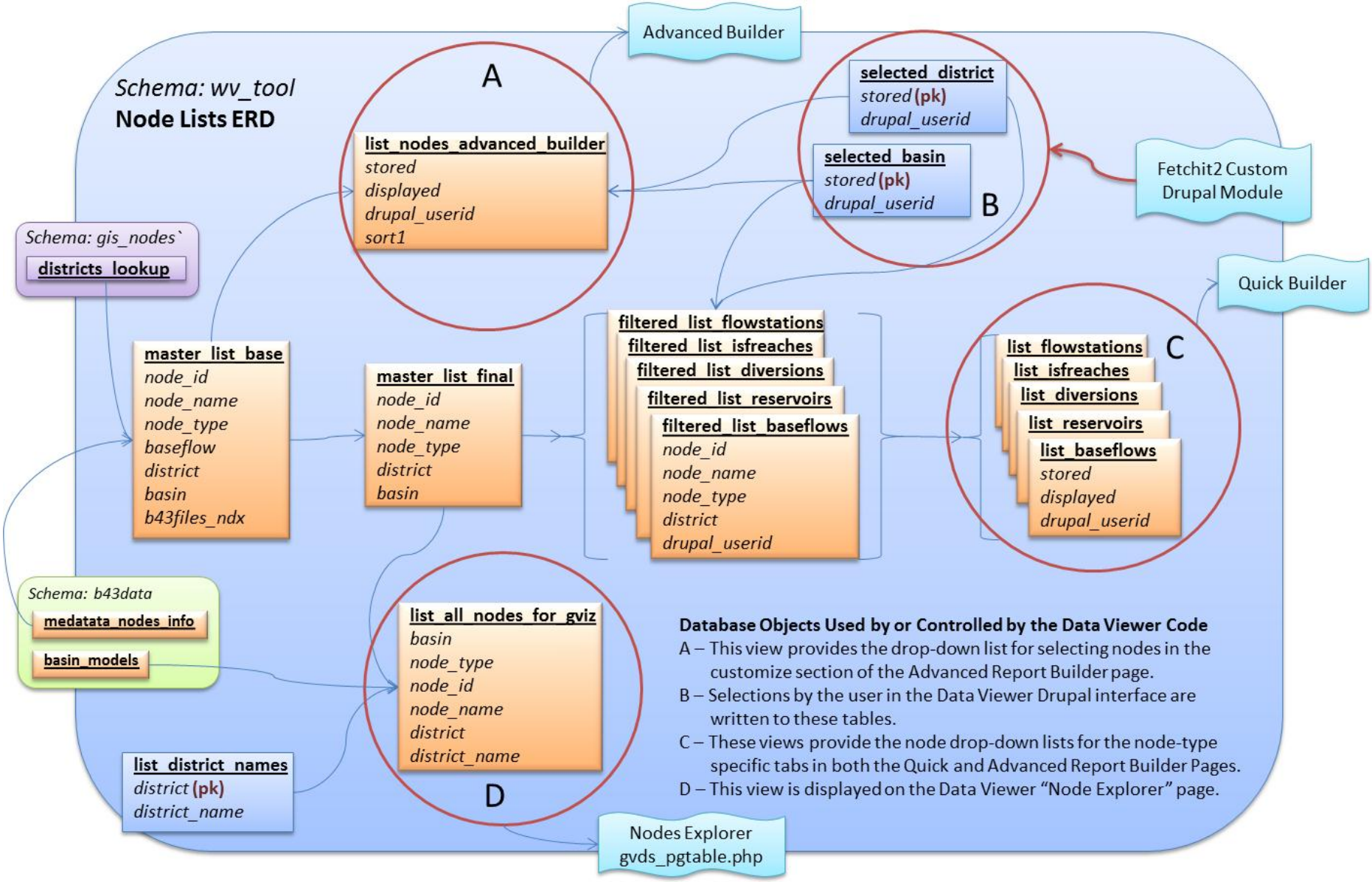
Figure A2 below shows the [gis_nodes] schema entity relationships diagram. This schema has one table with foreign keys to tables in the [b43data] schema.

FIGURE A2. ENTITY RELATIONSHIPS DIAGRAM GIS DERIVED NODE INFORMATION



The Data Viewer requires the user to first select a model basin. Optionally, the user may also filter nodes by district. The database stores selected basin and district filters for each user and the Data Viewer displays the appropriate set of nodes to the user. Please see Figure A3 on the following page for a diagram of the database objects related to processing and displaying lists of nodes.

FIGURE A3. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS RELATED TO NODE LISTS



3. Parameter Lists

The main database table storing model parameters, [list_parameter], contains the master list of parameters available to the Data Viewer interface. New parameters added to StateMod should be added to this table by hand, and also to the appropriate temporary table:

- [list_parameter_drupaledit_b43] or
- [list_parameter_drupaledit_b44].

The custom Drupal module "Fetchit2" makes it possible to edit the master parameter list (except for adding new records), in coordination with a stored PostgreSQL function [param_admin_submit()], which is called by the Drupal code to write changes made by the user to the master parameter list.

The view [list_parameters_lookup] provides a sorted list of parameters helpful as a reference to database administrators (it is not used by the Data Viewer interface). The rest of the views shown in Figure A4 are used to support the drop-down selections boxes in the data Viewer interface. Please see Figure A4 for a diagram of the database objects related to processing and displaying lists of nodes.

4. Basin, District and Climate Model Lists

Database tables also store lists of

- The five model basins associated with the CRWAS models
- The districts contained in these basins, and
- The Climate Model runs included in the CRWAS Phase I project.

The association of nodes to basins is inherent in the b43data metadata view [metadata_nodes_info] and passed to the [wvtool] schema in the view [master_list_base] (see Figure A3). Figure A5 includes an ERD showing the PostgreSQL database objects associated with display and processing these lists.

Please note that the Figure A3 (Nodes Lists) includes the original table illustrations for the tables [list_district_names], [selected_district] and [selected_basin] showing the table fields.

FIGURE A4. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS RELATED TO PARAMETER LISTS

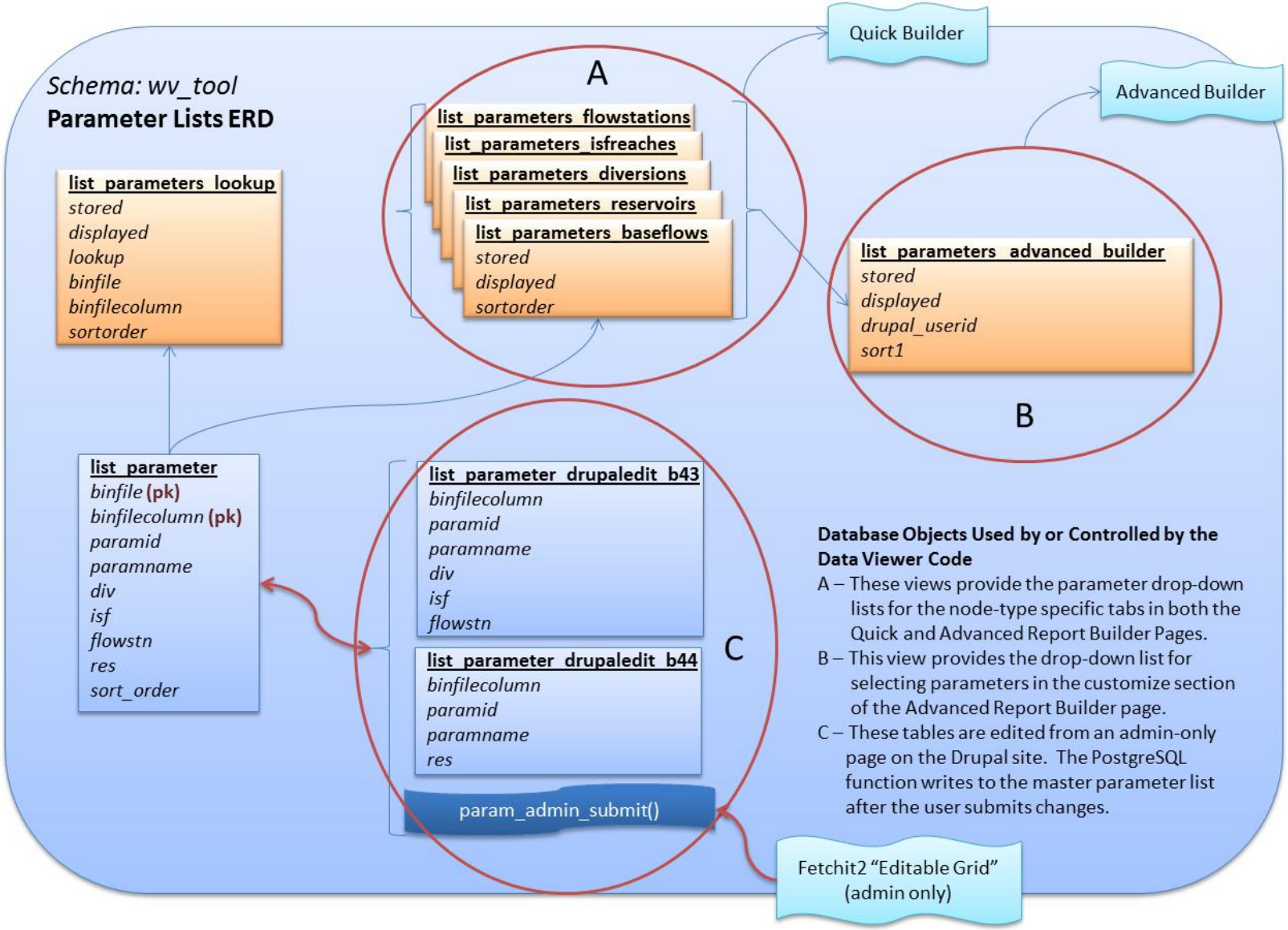
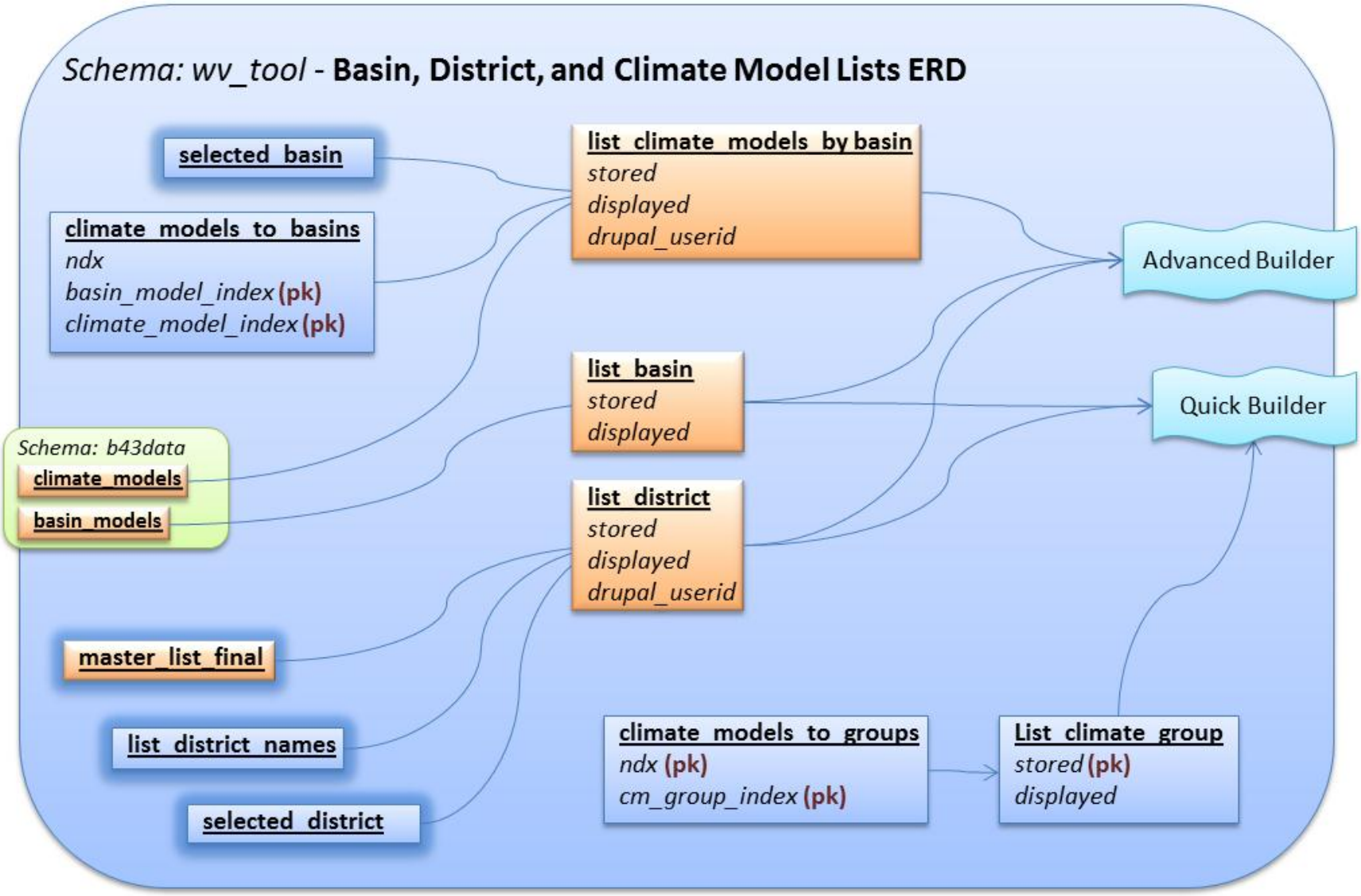


FIGURE A5. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS RELATED TO BASIN, DISTRICT, AND CLIMATE MODEL LISTS



5. User Selections

From the Data Viewer Drupal interface, the user is able to make a variety of selections. When submitted, custom PHP scripts call PostgreSQL scripts to write the user selections to tables in database tables. The following selections are stored in the following tables:

- Basin – [selected_basin]
- District – [selected_district]
- APPLY SELECTIONS – The combination of node, parameter, climate model, and trace making up one time series selection are stored in [selected_ts] and each individual selection is also stored:
  - Node – [selected_node]
  - Parameter – [selected_parameter]
  - Planning Horizon (Quick Builder Only) – [selected_climate model group]
  - Climate Model (Advanced Builder Only) – [selected_climate_model]
  - Trace – [selected_trace]
- MODIFY SELECTIONS – [selected_ts_edit] modified first and then code writes changes to [selected_ts]
- REMOVE SELECTIONS – the record numbers selected for removal are written to the table [selected_to_delete]. The table [selected_ts_edit] is modified first and then code writes changes to [selected_ts]

See Figures A6, A7, and A8 for details on the database organization supporting user selections.

Map Selections

When the user clicks PICK FROM MAP, PHP code writes a flag to database table [selected_call_page] to store whether the users clicked from the Quick Builder or the Advanced Builder. The code then returns the user to the flagged page after leaving the map. The table [return_paths] stores the Drupal page addresses for each time series selection tab by node type. After selecting a node from the map, the type of the selected node is used by the code to determine which tab to make active for the user.

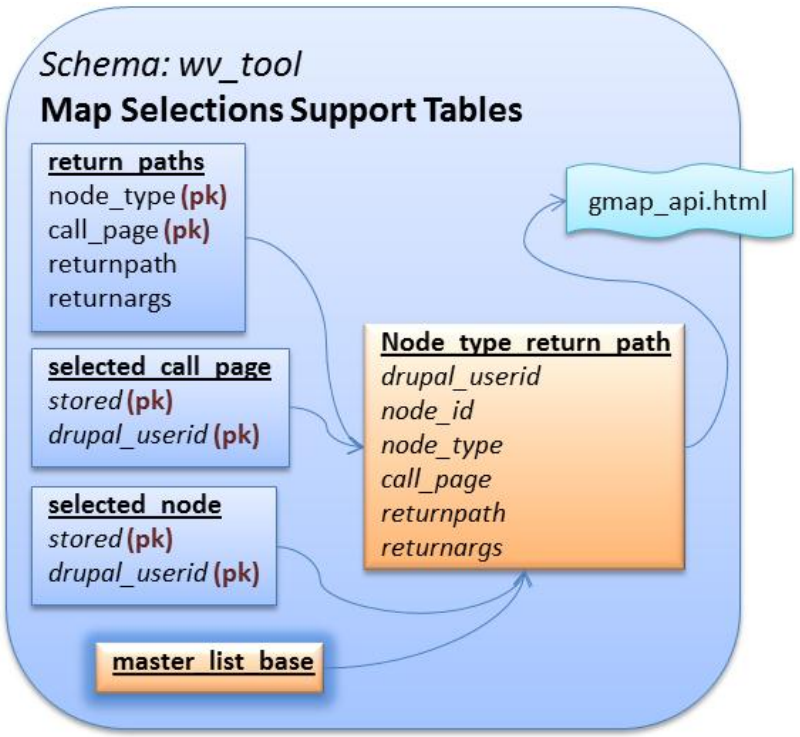


FIGURE A6. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS SUPPORTING USER MAP SELECTIONS

FIGURE A7. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS SUPPORTING ADDING AND MODIFYING SELECTIONS

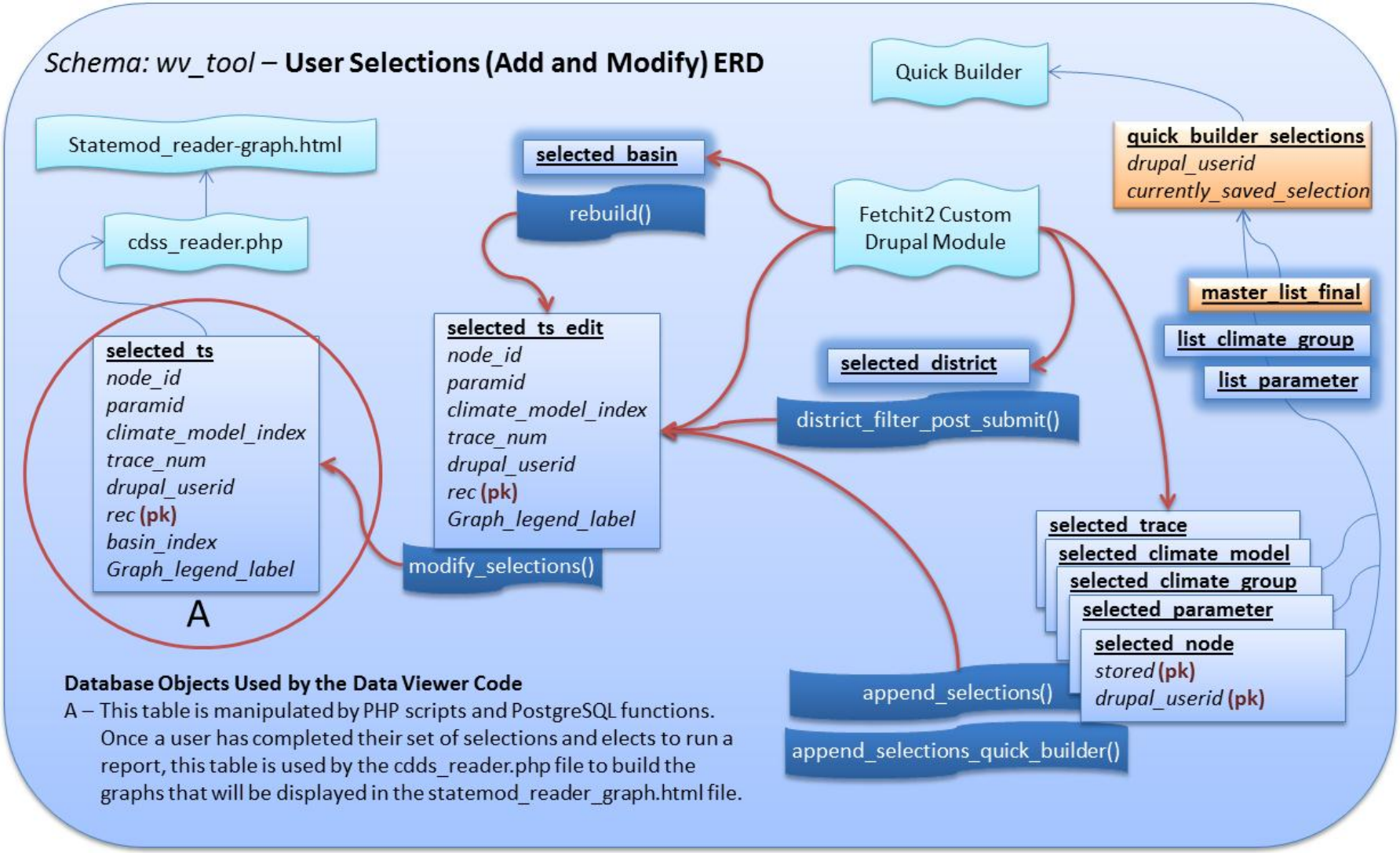
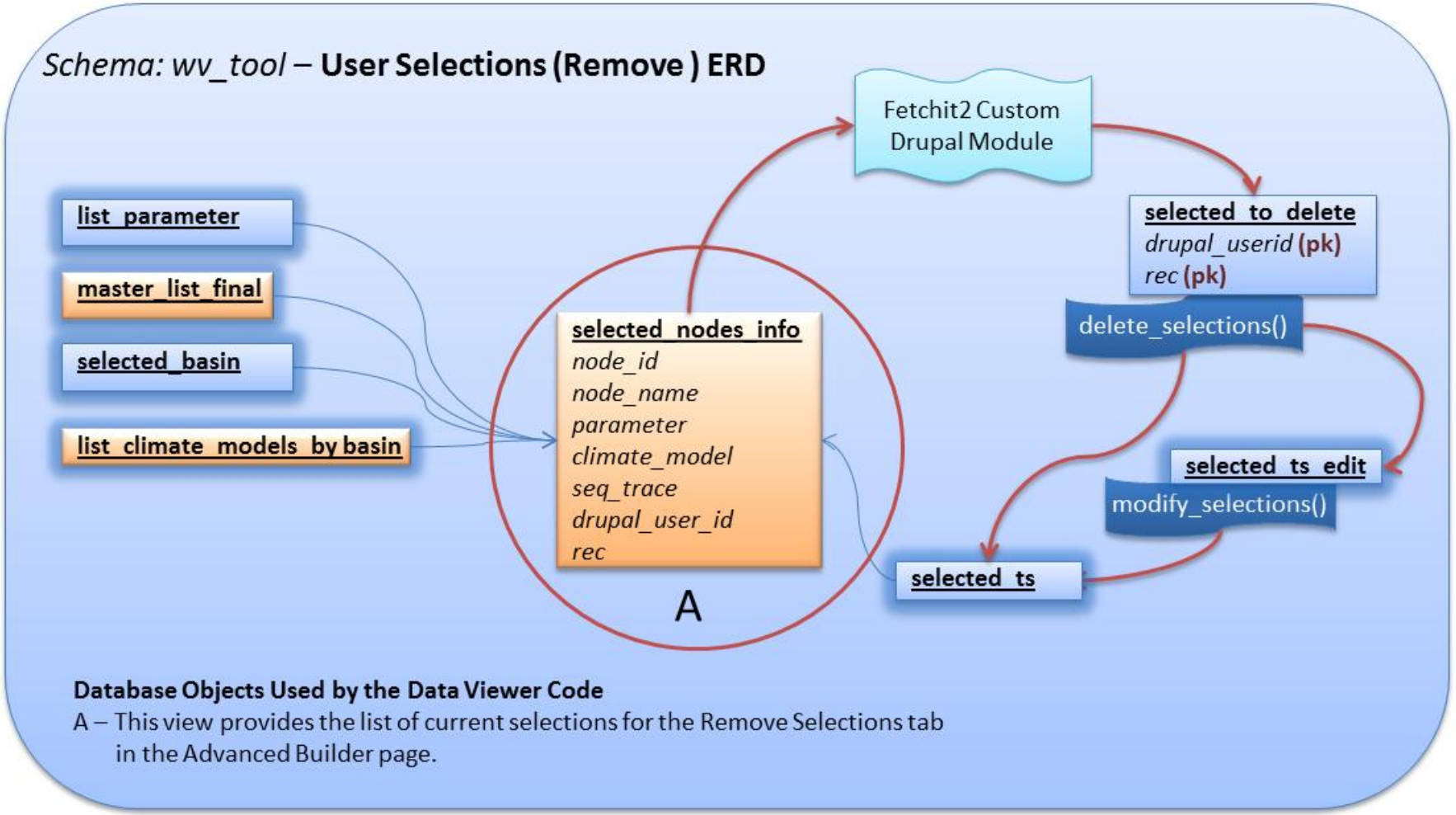


FIGURE A8. ENTITY RELATIONSHIPS DIAGRAM FOR DATABASE OBJECTS SUPPORTING REMOVING SELECTION



APPENDIX B: *wvtool_utils* Module

This module contains custom functions for the CRWAS Data Viewer Project. These functions can be called from code in a drupal page, a web page, or from another drupal module. The module resides in the following location on the server: /var/www/CRWAS/site/all/modules/custom/wvtool_utils

The functions in this module include:

function wvtool_rebuild($user_id): Populates default set of selections in the database for a specified user (used during testing).

function wvtool_check_district_filter($user_id): Runs postgresql code after user filters by district to filter available nodes by selected district.

function wvtool_get_basin($user_id): Runs postgresql code after user selects a basin to update other selection lists for the selected basin.

function wvtool_set_node($node_arg, $call_page_arg): Runs SQL and Postgresql code after user selects a node from either the Quick Builder or Advanced Builder page and stores the selection in the form. Returns user to the Builder page they started from.

function wvtool_quickbuilder_submit($user_id): Runs postgresql after user submits the Quick Builder form to store selections and display selections on the page.

function wvtool_advancedbuilder_submit($user_id): Runs postgresql after user submits the Advanced Builder form to store selections and display selections on the page.

function wvtool_advancedbuilder_delete($user_id): Runs postgresql after user submits a DELETE request on the Advanced Builder form to store selections and display selections on the page.

function wvtool_advancedbuilder_modify($user_id): Runs postgresql after user submits a MODIFY request on the Advanced Builder form to store and display modified values.

APPENDIX C: Fetchit2 Module

The Fetchit2 module was developed by Leonard Rice Engineers, Inc. to provide an interactive connection between Drupal and a remote server database and tools for Drupal programmers to create interactive web site content live-linked to the remote database. Please see the fetchit2.module PHP code for more detailed documentation of the code itself.

Fetchit2 functions create custom drupal forms that allow the creation of:

- HTML tables linked to tables in the remote database,
- Google Visualization (GViz) tables and charts linked to data in the remote database,
- Editable grids of data that when modified will change content in the remote database,
- Parameter Selection controls, including check boxes, option buttons, select buttons, and drop-down lists that push user specified selections into tables in the remote database.

Most of the user interaction takes place through either the editable grid objects or parameter selection objects. When a user submits information through either of these controls, that information is stored in the remote database and the web page is refreshed. Any web page content linked to the database (HTML or GViz Tables and Charts) and dependent on the user specified selections will change to reflect the new information.