SWAT+ Editor Documentation

Texas A&M AgriLife Research and the USDA-ARS

Feb 26, 2019

User Docs

Desktop interface to SWAT+ allowing the user to import a project from GIS, modify SWAT+ input, write the text files, and run the model.

CHAPTER 1

SWAT+ Resources

- SWAT+ Editor source code repository
- SWAT+ source code repository maintained by Blackland

CHAPTER 2

Contents:

2.1 Get Started Using SWAT+ Editor

SWAT+ Editor is still in the early phases of testing and development, so please proceed with caution.

2.1.1 What you'll need

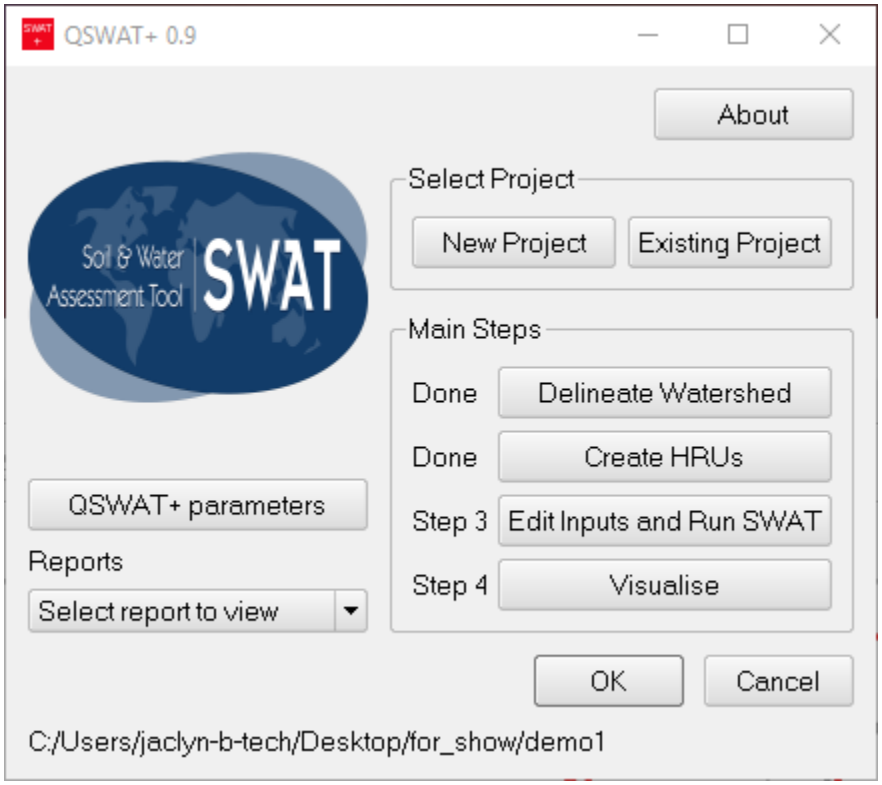
- QSWAT+
- SWAT+ Databases
- SWAT+ Editor

Download from the SWAT website

2.1.2 Step 1: Set up your project in QSWAT+

First get your project set up in QSWAT+. After your project is set up, create a back up copy; SWAT+ Editor manipulates the database, so during these early stages it is good to have a copy of your project before opening it in the editor so that you may start over if needed.

To open SWAT+ Editor, click the button for Step 3: Edit Inputs and Run SWAT from QSWAT+.



2.1.3 Step 2: Watch the MP4 video

Please watch the guide video before launching SWAT+ Editor. This video will quickly walk you through the steps needed to bring your QSWAT+ project into the editor, as well as show you how to catch and report errors.

2.1.4 Current functionality available in SWAT+ Editor

- Import weather generator data, add/edit/delete them
- Import weather stations using SWAT2012 or SWAT+ formats, add/edit/delete them
- Edit simulation times and print settings
- Add/edit/delete channels, aquifers, reservoirs, as well as some routing unit and hru files
- Write SWAT+ input files
- Run SWAT+
- Import output files to SQLite for use in QSWAT+ visualizer

2.1.5 Help

If you encounter an error, please follow the steps at the end of the guide video for capturing them fully. Contact Jaclyn by email or report the issue to the SWAT+ Editor issue tracker.

Note: Subscribe to our mailing list to be notified of new releases

2.2 How to Use SQLite

As mentioned in the *SWAT+ Editor Design* doc, SWAT+ Editor uses uses a SQLite database to hold model input data to allow easy manipulation by the user. The database is structured to closely resemble the SWAT+ ASCII text files in order to keep a clean link between the model and editor.

2.2.1 Opening the SQLite Database

We recommend users stick to the SWAT+ Editor program provided to browse and edit SWAT+ input data. However, if you need to access the database, we recommend using:

- SQLite Studio

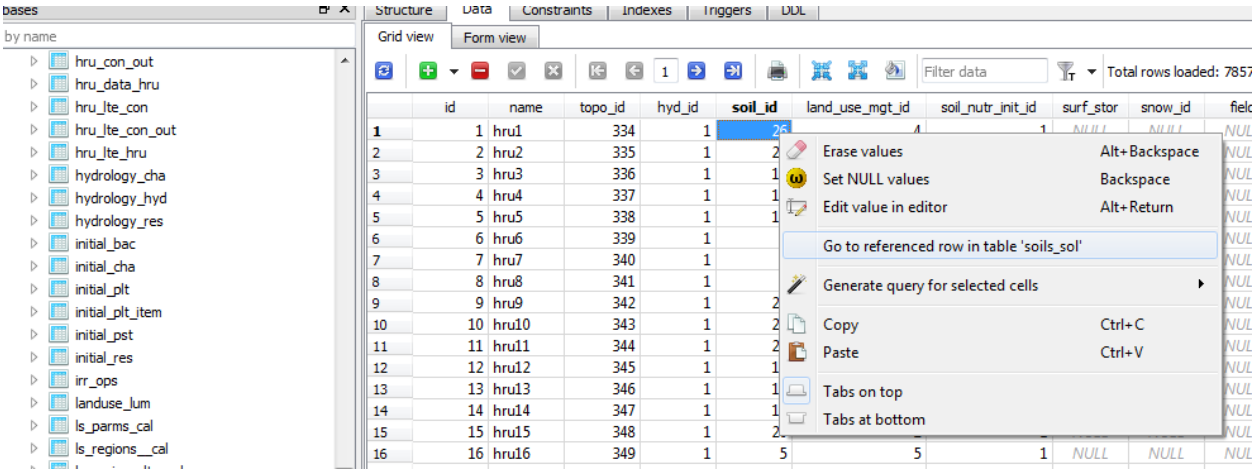
There are many other alternatives out there. A few of them are:

- DB Browser for SQLite
- SQLite Manager, Firefox Add-On

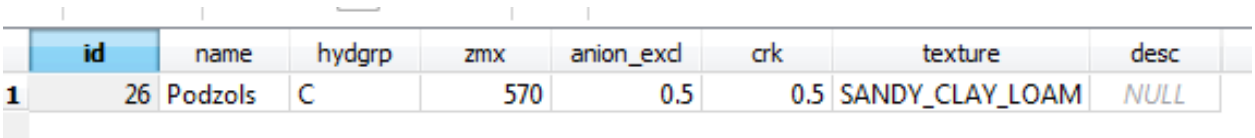
2.2.2 Understanding Table Relationships

SWAT+ contains many links between files, and the database follows suit by creating foreign key relationships where applicable. In the SWAT+ text files, you will see links reference object names. In the database however, these are done with an integer id. Luckily, relational databases make it easy to view the referenced row.

In SQLite Studio, simply right-click a foreign key id in a row of data, and select "Go to referenced row in table ..." as shown in the image below. This will open the referenced row of data in a new tab.



Results in:



	id	name	hydgrp	zmx	anion_excl	crk	texture	desc
1	26	Podzols	C	570	0.5	0.5	SANDY_CLAY_LOAM	NULL

2.3 SWAT+ Editor Design

SWAT+ Editor is a program that allows users to modify SWAT+ inputs easily without having to touch the SWAT+ input text files directly. The editor will import a watershed created in QSWAT+ or ArcSWAT+, or allow the user to create a SWAT+ project from scratch. The user may write input files and run the SWAT+ model through the editor.

2.3.1 Technologies

The following software is used to create and build SWAT+ Editor:

- Node.js 8.x
- Electron 1.8.x
- Vue.js 2.x
- Bootstrap 4
- Python 3.x
- PyInstaller
- SQLite
- Peewee ORM
- Git repository hosted on Bitbucket
- Read the Docs

2.3.2 Database Design

SWAT+ Editor uses a SQLite database to hold model input data to allow easy manipulation by the user. The database is structured to closely resemble the SWAT+ ASCII text files in order to keep a clean link between the model and editor. The following conventions are used in the project database:

- The table names will match the text file names, replacing any "." or "-" with an underscore "_".
- The table column names will match the model's variable names. All names use lowercase and underscores.
- Any text file with a variable number of repetitive columns will use a related table in the database. For example, many of the connection files contain a variable number of repeated outflow connection columns (obtyp_out, obtyno_out, hytyp_out, frac_out). In the database, we represent these in a separate table, basically transposing a potentially long horizontal file to columns.
- All tables will use a numeric "id" as the primary key, and foreign key relationships will use these integer ids instead of a text name. This will allow for easier modification of these object names by the user and help keep the database size down for large projects.

A separate SQLite database containing common datasets and input metadata will be provided with SWAT+ Editor. (This is a replacement for the SWAT2012.mdb packaged with SWAT2012 versions of ArcSWAT, QSWAT, and SWATeditor.)

In addition, reformatted SSURGO and STATSGO soils databases will be available for download. The structure of the soils database has been split into two tables: a soil table and soil_layer table.

Similarly, the global weather weather generator database will be available for download in SQLite format. The structure of the wgn database has been split into two tables: a wgn table and wgn_monthly_value table.

A tool will be provided to convert old soils and wgn tables to the new format.

2.3.3 SWAT+ Documentation

SWAT+ is the latest version of the SWAT model, which is a command line executable written in Fortran. It uses several ASCII text files as inputs to the model. The documentation for these files is still under development.

- SWAT+ code repository on Bitbucket

2.4 Database Documentation

2.4.1 Project Database

The project database in SWAT+ Editor is made in SQLite. SWAT+ Editor uses the Peewee ORM (object-relational mapping) to represent and work with the tables in Python. The use of an ORM provides a layer of abstraction and portability in hopes of streamlining future SWAT+ development projects.

Download a SQLite file containing the project table structure.

Note: This page is incomplete and under development.

Foreign Keys

Relationships are defined in a Peewee ORM python class as a `ForeignKeyField`. In the python class, the field will be named after the object it is referencing. In the database, this name will automatically be appended by the referencing table's column name, which is usually `id`.

For example, we have two tables representing soils: soils (`soils_sol`) and layers (`soils_sol_layer`). The layer table has a foreign key to the main soils table, so we know to which soil the layer belongs. In the python class, this field is named `soil`, and in the database it is called `soil_id`. See the *Soils module* for reference.

Modules and Tables

A Python module is created for each input classification, based on the SWAT+ file.cio input file:

- *Simulation*
- *Climate*
- *Connect*
- *Channel*
- *Reservoir*
- *Routing Unit*
- *HRU*
- *Delivery Ratio*
- *Aquifer*
- *Herd*
- *Water Rights*
- *Link*

- *Basin*
- *Hydrology*
- *Exco*
- *Bacteria*
- *Structural*
- *Parameter Database*
- *Operations*
- *Landuse Management*
- *Change*
- *Init*
- *Soils*
- *Decision Table*
- *Constituents*
- *Regions*

Simulation

Section label in file.cio	simulation
Python module	database/project/simulation.py

Tables

Text file name	Database table name	Related tables
time.sim	time_sim	
print.prt	print_prt	print_prt_aa_int, print_prt_object
object.prt	object_prt	
object.cnt	object_cnt	

Time_sim

```
class Time_sim(BaseModel):
    day_start = IntegerField()
    yrc_start = IntegerField()
    day_end = IntegerField()
    yrc_end = IntegerField()
    step = IntegerField()
```

time_sim		
PK	id	int
	day_start	int
	yrc_start	int
	day_end	int
	yrc_end	int
	step	int

Print_prt

```
class Print_prt(BaseModel):
    nyskip = IntegerField()
    day_start = IntegerField()
    yrc_start = IntegerField()
    day_end = IntegerField()
    yrc_end = IntegerField()
    interval = IntegerField()
    csvout = BooleanField()
    dbout = BooleanField()
    cdfout = BooleanField()
    soilout = BooleanField()
    mgtout = BooleanField()
    hydcon = BooleanField()
    fdcout = BooleanField()
```

print_prt		
PK	id	int
	nyskip	int
	day_start	int
	yrc_start	int
	day_end	int
	yrc_end	int
	interval	int
	csvout	bool
	dbout	bool
	cdfout	bool
	soilout	bool
	mgtout	bool
	hydcon	bool
	fdcout	bool

```
class Print_prt_aa_int(BaseModel):
    print_prt = ForeignKeyField(Print_prt, on_delete='CASCADE', related_name='aa_ints
↪')
    year = IntegerField()
```

print_prt_aa_int			
PK	id	int	
FK	print_prt_id	int	REFERENCES print_prt (id) ON DELETE CASCADE
	year	int	

```
class Print_prt_object(BaseModel):
    print_prt = ForeignKeyField(Print_prt, on_delete='CASCADE', related_name='objects
↪')
    name = CharField()
    daily = BooleanField()
    monthly = BooleanField()
    yearly = BooleanField()
    avann = BooleanField()
```

print_prt_object			
PK	id	int	
FK	print_prt_id	int	REFERENCES print_prt (id) ON DELETE CASCADE
	name	text	
	daily	bool	
	monthly	bool	
	yearly	bool	
	avann	bool	

Object_prt

```
class Object_prt(BaseModel):
    ob_typ = CharField()
    ob_typ_no = IntegerField()
    hyd_typ = CharField()
    filename = CharField()
```

object_prt		
PK	id	int
	ob_typ	text
	ob_typ_no	int
	hyd_typ	text
	filename	text

Object_cnt

Note: If the integer fields are set to 0, use the total - calculated programmatically.

ls_area and tot_area from the text files are not included as they will be calculated.

```
class Object_cnt(BaseModel):
    name = CharField()
    obj = IntegerField(default=0)
    hru = IntegerField(default=0)
    lhru = IntegerField(default=0)
    rtu = IntegerField(default=0)
    mfl = IntegerField(default=0)
    aqu = IntegerField(default=0)
    cha = IntegerField(default=0)
```

(continues on next page)

(continued from previous page)

```
    res = IntegerField(default=0)
    rec = IntegerField(default=0)
    exco = IntegerField(default=0)
    dlr = IntegerField(default=0)
    can = IntegerField(default=0)
    pmp = IntegerField(default=0)
    out = IntegerField(default=0)
    lcha = IntegerField(default=0)
    aqu2d = IntegerField(default=0)
    hrd = IntegerField(default=0)
    wro = IntegerField(default=0)
```

object_cnt		
PK	id	int
	name	text
	obj	int
	hru	int
	lhru	int
	rtu	int
	mfl	int
	aqu	int
	cha	int
	res	int
	rec	int
	exco	int
	dlr	int
	can	int
	pmp	int
	out	int
	lcha	int
	aqu2d	int
	hrd	int
	wro	int

Climate

Section label in file.cio	climate
Python module	database/project/climate.py

Tables

Text file name	Database table name	Related tables
weather_wgn.cli	weather_wgn_cli	weather_wgn_cli_mon
weather_sta.cli	weather_sta_cli	
wind_dir.cli	wind_dir_cli	
atmo.cli	atmo_cli	atmo_cli_sta, atmo_cli_sta_value
	weather_file	

Weather_wgn_cli

The weather generators are populated from an external table formatted in a similar structure to the one below. US and global CFSR SQLite databases are available on the SWAT+ Editor repository.

```
class Weather_wgn_cli(BaseModel):
    name = CharField()
    lat = DoubleField()
    lon = DoubleField()
    elev = DoubleField()
    rain_yrs = IntegerField()
```

weather_wgn_cli		
PK	id	int
	name	text
	lat	real
	lon	real
	elev	real
	rain_yrs	int

Weather_wgn_cli_mon

Each wgn has a set of values for each month. In SWAT2012, the months were represented in a single table along with the wgn definition. In SWAT+ we have opted for a normalized approach.

```
class Weather_wgn_cli_mon(BaseModel):
    weather_wgn_cli = ForeignKeyField(Weather_wgn_cli, related_name='monthly_values',␣
↪on_delete='CASCADE')
    month = IntegerField()
    tmp_max_ave = DoubleField()
    tmp_min_ave = DoubleField()
    tmp_max_sd = DoubleField()
    tmp_min_sd = DoubleField()
    pcp_ave = DoubleField()
    pcp_sd = DoubleField()
    pcp_skew = DoubleField()
    wet_dry = DoubleField()
    wet_wet = DoubleField()
    pcp_days = DoubleField()
    pcp_hhr = DoubleField()
    slr_ave = DoubleField()
    dew_ave = DoubleField()
    wnd_ave = DoubleField()
```

weather_wgn_cli_mon			
PK	id	int	
FK	weather_wgn_cli_id	int	REFERENCES weather_wgn_cli (id) ON DELETE CASCADE
	month	int	
	tmp_max_ave	real	
	tmp_min_ave	real	
	tmp_max_sd	real	
	tmp_min_sd	real	
	pcp_ave	real	
	pcp_sd	real	
	pcp_skew	real	
	wet_dry	real	
	wet_wet	real	
	pcp_days	real	
	pcp_hhr	real	
	slr_ave	real	
	dew_ave	real	
	wnd_ave	real	

Weather_sta_cli

```
class Weather_sta_cli(BaseModel):
    name = CharField()
    wgn = ForeignKeyField(Weather_wgn_cli, null=True, on_delete='SET NULL')
    pcp = CharField(null=True)
    tmp = CharField(null=True)
    slr = CharField(null=True)
    hmd = CharField(null=True)
    wnd = CharField(null=True)
    wnd_dir = CharField(null=True)
    atmo_dep = CharField(null=True)
    lat = DoubleField(null=True)
    lon = DoubleField(null=True)
```

weather_sta_cli			
PK	id	int	
	name	text	
FK	wgn_id	int	REFERENCES weather_wgn_cli (id) ON DELETE SET NULL
	pcp	text	null
	tmp	text	null
	slr	text	null
	hmd	text	null
	wnd	text	null
	wnd_dir	text	null
	atmo_dep	text	null
	lat	real	null
	lon	real	null

Wind_dir_cli

Note: This table may be incomplete / inaccurate.

```
class Wind_dir_cli(BaseModel):
    name = CharField()
    cnt = IntegerField()
    n = DoubleField()
    nne = DoubleField()
    ne = DoubleField()
    ene = DoubleField()
    e = DoubleField()
    ese = DoubleField()
    se = DoubleField()
    sse = DoubleField()
    s = DoubleField()
    ssw = DoubleField()
    sw = DoubleField()
    wsw = DoubleField()
    w = DoubleField()
    wnw = DoubleField()
    nw = DoubleField()
    nnw = DoubleField()
```

wind_dir_cli		
PK	id	int
	name	text
	cnt	int
	n	real
	nne	real
	ne	real
	ene	real
	e	real
	ese	real
	se	real
	sse	real
	s	real
	ssw	real
	sw	real
	wsw	real
	w	real
	wnw	real
	nw	real
	nnw	real

Atmo_cli

```
class Atmo_cli(BaseModel):
    filename = CharField()
    timestep = CharField()
    mo_init = IntegerField()
    yr_init = IntegerField()
    num_aa = IntegerField()
```

atmo_cli		
PK	id	int
	filename	text
	timestep	int
	mo_init	int
	yr_init	int
	num_aa	int

```
class Atmo_cli_sta(BaseModel):
    atmo_cli = ForeignKeyField(Atmo_cli, on_delete='CASCADE', related_name='stations')
    name = CharField()
```

atmo_cli_sta			
PK	id	int	
	atmo_cli	int	REFERENCES atmo_cli (id) ON DELETE CASCADE
	name	text	

```
class Atmo_cli_sta_value(BaseModel):
    sta = ForeignKeyField(Atmo_cli_sta, on_delete='CASCADE', related_name='values')
    timestep = IntegerField()
    nh4_wet = DoubleField()
    no3_wet = DoubleField()
    nh4_dry = DoubleField()
    no3_dry = DoubleField()
```

atmo_cli_sta_value			
PK	id	int	
	sta	int	REFERENCES atmo_cli_sta (id) ON DELETE CASCADE
	timestep	int	
	nh4_wet	real	
	no3_wet	real	
	nh4_dry	real	
	no3_dry	real	

Weather_file

The purpose of this table is to keep track of observed weather files and the lat/lon coordinates associated with each for easy pairing with weather stations.

The type field should be one of the following: hmd, pcp, slr, tmp, wnd

```
class Weather_file(BaseModel):
    filename = CharField()
    type = CharField()
    lat = DoubleField()
    lon = DoubleField()
```

weather_file		
PK	id	int
	filename	text
	type	text
	lat	real
	lon	real

Connect

Section label in file.cio	connect
Python module	database/project/connect.py

Tables

Text file name	Database table name	Related tables
hru.con	hru_con	hru_con_out, hru_data_hru
hru_lte.con	hru_lte_con	hru_lte_con_out, hru_lte_hr
rout_unit.con	rout_unit_con	rout_unit_con_out, rout_unit_rtu
modflow.con	modflow_con	modflow_con_out
aquifer.con	aquifer_con	aquifer_con_out, aquifer_aqu
aquifer2d.con	aquifer2d_con	aquifer2d_con_out, aquifer_aqu
channel.con	channel_con	channel_con_out, channel_cha
reservoir.con	reservoir_con	reservoir_con_out, reservoir_res
recall.con	recall_con	recall_con_out, recall_rec
exco.con	exco_con	exco_con_out, exco_exc
delratio.con	delratio_con	delratio_con_out, delratio_del
outlet.con	outlet_con	outlet_con_out
chandeg.con	chandeg_con	chandeg_con_out, channel_lte_cha

Each connect file has basically the same structure and is represented by two tables in the database: the connect table and the outflow parameters table. Because of the shared structure, we use the following base classes in Python from which each connect table above is inherited.

```
class Con(BaseModel):
    """Inheritable base class for all connect files."""
    name = CharField()
    area = DoubleField()
    lat = DoubleField()
    lon = DoubleField()
    elev = DoubleField(null=True)
    wst = ForeignKeyField(climate.Weather_sta_cli, null=True, on_delete='SET NULL')
    cst = ForeignKeyField(constituents.Constituents_cs, null=True)
    ovfl = IntegerField()
    rule = IntegerField()

class Con_out(BaseModel):
    """Inheritable base class for all outflow parameters in many of the connect files.
↪"""
    order = IntegerField()
```

(continues on next page)

(continued from previous page)

```
    obj_typ = CharField()
    obj_id = IntegerField()
    hyd_typ = CharField()
    frac = DoubleField()
```

con			
PK	id	int	
	name	text	
	area	real	
	lat	real	
	lon	real	
	elev	real	null
FK	wst_id	int	REFERENCES weather_sta_cli (id) ON DELETE SET NULL
FK	cst_id	int	REFERENCES constituents_cs (id)
	ovfl	int	
	rule	int	

con_out		
PK	id	int
	order	int
	obj_typ	text
	obj_id	int
	hyd_typ	text
	frac	real

Specific code implementations of each connect table are defined below:

```
class Hru_con(Con):
    hru = ForeignKeyField(hru_db.Hru_data_hru, null=True)


class Hru_con_out(Con_out):
    hru_con = ForeignKeyField(Hru_con, on_delete='CASCADE', related_name='hru_con_outs
↪')


class Hru_lte_con(Con):
    lhru = ForeignKeyField(hru_db.Hru_lte_hru, null=True)


class Hru_lte_con_out(Con_out):
    hru_lte_con = ForeignKeyField(Hru_lte_con, on_delete='CASCADE', related_name='hru_
↪lte_con_outs')


class Rout_unit_con(Con):
    rtu = ForeignKeyField(routing_unit.Rout_unit_rtu, null=True)


class Rout_unit_con_out(Con_out):
    rtu_con = ForeignKeyField(Rout_unit_con, on_delete ='CASCADE', related_name='rout_
↪unit_con_outs')
```

(continues on next page)

(continued from previous page)

```
class Modflow_con(Con):
    mfl = IntegerField()  # Should be FK to something, but no modflow object yet that␣
↪I can find.


class Modflow_con_out(Con_out):
    modflow_con = ForeignKeyField(Modflow_con, on_delete='CASCADE', related_name=
↪'modflow_con_outs')


class Aquifer_con(Con):
    aqu = ForeignKeyField(aquifer.Aquifer_aqu, null=True)


class Aquifer_con_out(Con_out):
    aquifer_con = ForeignKeyField(Aquifer_con, on_delete='CASCADE', related_name=
↪'aquifer_con_outs')


class Aquifer2d_con(Con):
    aqu2d = ForeignKeyField(aquifer.Aquifer_aqu, null=True) # Some doubt in␣
↪documentation about this link


class Aquifer2d_con_out(Con_out):
    aquifer2d_con = ForeignKeyField(Aquifer2d_con, on_delete='CASCADE', related_name=
↪'aquifer2d_con_outs')


class Channel_con(Con):
    cha = ForeignKeyField(channel.Channel_cha, null=True)


class Channel_con_out(Con_out):
    channel_con = ForeignKeyField(Channel_con, on_delete='CASCADE', related_name=
↪'channel_con_outs')


class Reservoir_con(Con):
    res = ForeignKeyField(reservoir.Reservoir_res, null=True)


class Reservoir_con_out(Con_out):
    reservoir_con = ForeignKeyField(Reservoir_con, on_delete='CASCADE', related_name=
↪'reservoir_con_outs')


class Recall_con(Con):
    rec = ForeignKeyField(exco.Recall_rec, null=True)


class Recall_con_out(Con_out):
    recall_con = ForeignKeyField(Recall_con, on_delete='CASCADE', related_name=
↪'recall_con_outs')
```

(continues on next page)

(continued from previous page)

```
class Exco_con(Con):
    exco = ForeignKeyField(exco.Exco_exc, null=True)


class Exco_con_out(Con_out):
    exco_con = ForeignKeyField(Exco_con, on_delete='CASCADE', related_name='exco_con_
↪outs')


class Delratio_con(Con):
    dlr = ForeignKeyField(dr.Delratio_del, null=True)


class Delratio_con_out(Con_out):
    delratio_con = ForeignKeyField(Delratio_con, on_delete='CASCADE', related_name=
↪'delratio_con_outs')


class Outlet_con(Con):
    out = IntegerField() # Should be FK to something, but no outlet object yet that I␣
↪can find.


class Outlet_con_out(Con_out):
    outlet_con = ForeignKeyField(Outlet_con, on_delete='CASCADE', related_name=
↪'outlet_con_outs')


class Chandeg_con(Con):
    lcha = ForeignKeyField(channel.Channel_lte_cha, null=True)


class Chandeg_con_out(Con_out):
    chandeg_con = ForeignKeyField(Chandeg_con, on_delete='CASCADE', related_name=
↪'chandeg_con_outs')
```

Channel

Reservoir

Routing Unit

HRU

Delivery Ratio

Aquifer

Herd

Water Rights

Link

Basin

Hydrology

Exco

Bacteria

Structural

Parameter Database

Operations

Landuse Management

Change

Init

Soils

Section label in file.cio	soils
Python module	database/project/soils.py

Tables

Text file name	Database table name	Related tables
soils.sol	soils_sol	soils_sol_layer
nutrients.sol	nutrients_sol	

References

Table	Foreign key	Referenced table	Ref. table key
hru_data_hru	soil_id	soils_sol	id
hru_data_hru	soil_nutr_init_id	nutrients_sol	id

Soils_sol

The soils represented in this table are the soils used in the user's model. They are populated from an external table of soils formatted in a similar structure to the one below. US SSURGO and STATSGO (and potentially other global soils) SQLite databases are available on the SWAT+ Editor repository. A tool will be provided to convert SWAT2012 soil databases to the new format.

```
class Soils_sol(BaseModel):
    name = CharField(unique=True)
    hyd_grp = CharField()
    dp_tot = DoubleField()
    anion_excl = DoubleField()
    perc_crk = DoubleField()
    texture = CharField()
    description = TextField(null=True)
```

soils_sol			
PK	id	int	
	name	text	unique
	hyd_grp	text	
	dp_tot	real	
	anion_excl	real	
	perc_crk	real	
	texture	text	
	description	text	null

Soils_sol_layer

Each soil can have many layers. In SWAT2012, the layers were represented in a single table along with the soil definition. In SWAT+ we have opted for a normalized approach.

```
class Soils_sol_layer(BaseModel):
    soil = ForeignKeyField(Soils_sol, on_delete='CASCADE', related_name='layers')
    layer_num = IntegerField()
    dp = DoubleField()
    bd = DoubleField()
    awc = DoubleField()
    soil_k = DoubleField()
    carbon = DoubleField()
    clay = DoubleField()
    silt = DoubleField()
    sand = DoubleField()
    rock = DoubleField()
    alb = DoubleField()
    usle_k = DoubleField()
    ec = DoubleField()
    caco3 = DoubleField(null=True)
    ph = DoubleField(null=True)
```

soils_sol_layer			
PK	id	int	
FK	soil_id	int	REFERENCES soils_sol (id) ON DELETE CASCADE
	layer_num	int	
	dp	real	
	bd	real	
	awc	real	
	soil_k	real	
	carb0n	real	
	clay	real	
	silt	real	
	sand	real	
	rock	real	
	alb	real	
	usle_k	real	
	ec	real	
	caco3	real	null
	ph	real	null

Nutrients_sol

```
class Nutrients_sol(BaseModel):
    name = CharField(unique=True)
    dp_co = DoubleField()
    tot_n = DoubleField()
    min_n = DoubleField()
    org_n = DoubleField()
    tot_p = DoubleField()
    min_p = DoubleField()
    org_p = DoubleField()
    sol_p = DoubleField()
    h3a_p = DoubleField()
    mehl_p = DoubleField()
    bray_p = DoubleField()
    description = TextField(null=True)
```

nutrients_sol			
PK	id	int	
	name	text	unique
	dp_co	real	
	tot_n	real	
	min_n	real	
	org_n	real	
	tot_p	real	
	min_p	real	
	org_p	real	
	sol_p	real	
	h3a_p	real	
	mehl_p	real	
	bray_p	real	
	description	text	null

Decision Table

Constituents

Regions

2.5 Release Notes

2.5.1 1.1.0 - status pending

SWAT+ Editor 1.1.0 is compatible with SWAT+ rev.57.

Editor changes:

- Upgrade function available for projects made with version 1.0.0
- Packaged with SWAT+ rev. 57
- Edit -> Connections -> Channels -> Initial - page structure change to match input file changes with rev. 57
- Edit -> Connections -> Reservoirs -> Initial - page structure change to match input file changes with rev. 57
- Edit -> Hydrology - all pages now allow the edit multiple records feature already available in the other sections

Project database changes:

- d_table_dtl - add column file_name, repopulate table based on 4 new decision table files: lum.dtl, res_rel.dtl, scen_lu.dtl, flo_con.dtl
- d_table_dtl_act - add column const2
- d_table_dtl_act - rename columns application->fp and type->option
- recall_dat - drop columns sol_pest, srb_pest, p_bact, lp_bact, metl1, metl2, metl3
- exco_om_exc - drop columns sol_pest, srb_pest, p_bact, lp_bact, metl1, metl2, metl3
- exco_om_exc - rename columns ptl_n->orgn, ptl_p->sedp, no3_n->no3, sol_p->solp, nh3_n->nh3, no2_n->no2, bod->cbod, oxy->dox, sm_agg->sag, lg_agg->lg_agg
- aquifer_aqu - drop columns gw_dp, gw_ht
- aquifer_aqu - add columns dep_bot (default value 10), dep_wt (default value 5)
- aquifer_aqu - change spec_yld value from 0 to 0.05
- fertilizer_frt - drop columns p_bact, lp_bact, sol_bact
- fertilizer_frt - add column pathogens
- hydrology_hyd - drop column dp_imp
- pesticide_cha - rename column sed_conc->pst_solub
- channel_lte_cha - rename table to hyd_sed_lte_cha
- hru_data_hru - drop column soil_nut_id
- hru_data_hru - add column soil_plant_init_id
- cal_parms_cal - change column type of units from number to text

- initial_cha - drop existing columns, add new columns: org_min_id, pest_id, path_id, hmet_id, salt_id (foreign keys to new tables in init)
- initial_res - drop existing columns, add new columns: org_min_id, pest_id, path_id, hmet_id, salt_id (foreign keys to new tables in init)
- constituents_cs - drop and re-create table
- dr_om_del, dr_pest_del, dr_path_del, dr_hmet_del, dr_salt_del, delratio_del - drop and re-create tables

Drop tables:

- pest_soil_ini
- pest_soil_ini_item
- path_soil_ini
- hmet_soil_ini
- salt_soil_ini

Add new tables:

- soil_plant_ini
- om_water_ini
- pest_hru_ini
- pest_hru_ini_item
- pest_water_ini
- path_hru_ini
- path_water_ini
- hmet_hru_ini
- hmet_water_ini
- salt_hru_ini
- salt_water_ini
- temperature_cha
- channel_lte_cha (new structure; not the same as old table renamed to hyd_sed_lte_cha)

SWAT+ Datasets database changes:

- d_table_dtl - add column file_name, repopulate table based on 4 new decision table files: lum.dtl, res_rel.dtl, scen_lu.dtl, flo_con.dtl
- d_table_dtl_act - add column const2
- d_table_dtl_act - rename columns application->fp and type->option
- Replace all decision table data
- Replace all plants_plt data
- Replace all fertilizer_frt data
- Replace all var_range data
- Add new table: version

Output database changes:

- New naming structure based on SWAT+ rev. 57