AMP Database Tool

User Manual  
Bruce Taylor, Rocket Science Consulting

## Starting and Stopping

The AMP Database tool is a Java application, and can be started in one of two ways:

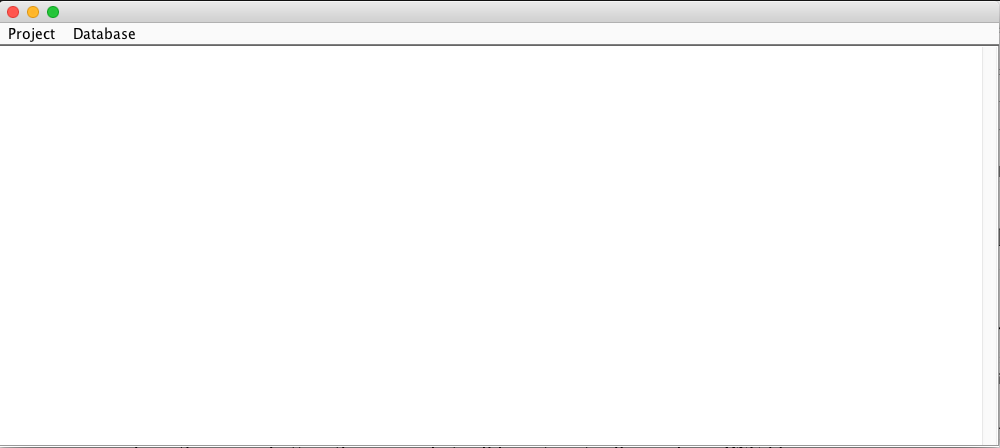
1. Double-clicking on the “AMPDBTool.jar” icon 
2. From the command line: java –jar AMPDBTool.jar

The AMP Database tool can be stopped by closing (not minimizing) its window, or using the File->Quit menu item.

## General Operation

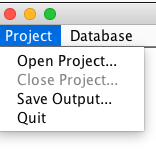
The AMP Database Tool’s window is shown below. It comprises three sections:

1. The File menu controls selection of AMP projects to be imported into the database
2. The Database menu controls the selection, creation, and population of databases with AMP data.
3. The Progress Window that shows progress of population operations and any errors that occur during those operations.

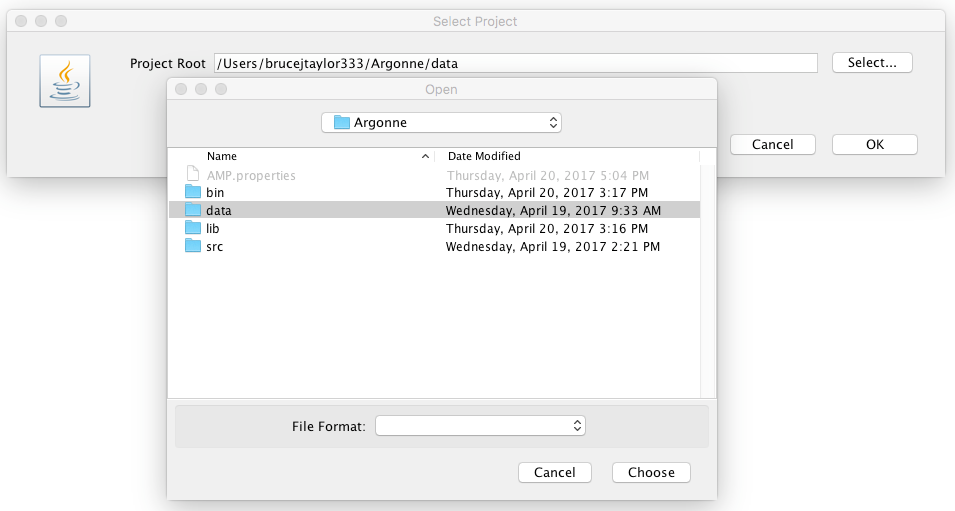


## Managing Projects

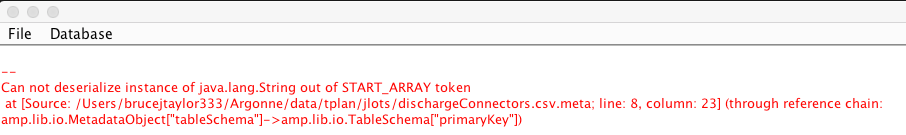
An AMP DB “project” is a root directory of an AMP output directory tree. All Metadata files accessible from the root directory will be included in schema generation and database population. The project is controlled from the “File” menu as shown:



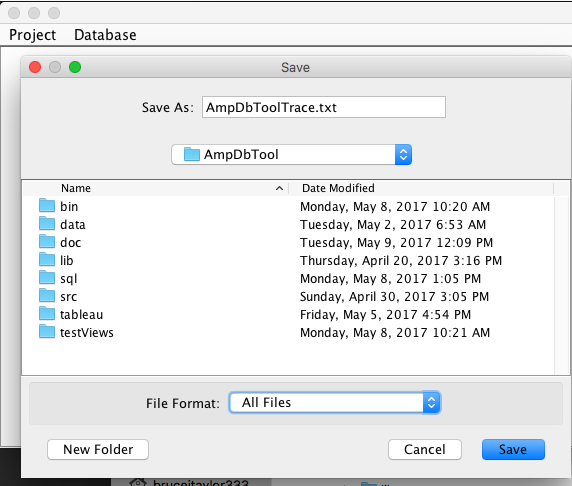
The “Open Project” menu item opens the following dialog that allows the user to select the project root either by entering its path directly or be using the standard file chooser widget. The “Close Project” menu item removes the current project path from consideration.



When a project is opened, the AMP Database tool immediately reads and parses all metadata files (those whose file extension is “.meta”) and reports errors on the progress window as shown below.



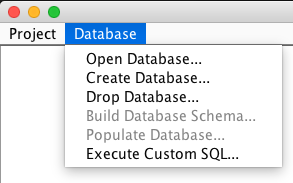
To save the contents of the progress window to a text file for later analysis, select the “Save Output…” menu item and use the subsequent file selector window to indicate where the text output should be saved:



## Managing Databases

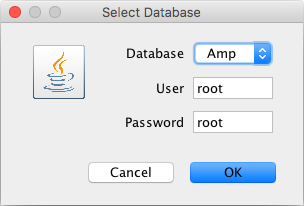
The AMP Database Tool supports the basic operations on databases, as shown in the menu illustrated below:

1. Opening a database  
   The user must select an available database and supply a user name and password.
2. Creating a database  
   The user must supply a database name, a user name, and an appropriate password.
3. Dropping a database  
   The selected database is deleted and removed from the available database list
4. Build Database Schema  
   The tool will clear all tables from the specified database and create new tables and indices as determined by the metadata files in the project.
5. Populate Database  
   The tool will empty all tables in the database and reload their contents from the CSV files in the project, as directed by the metadata files.
6. Execute Custom SQL  
   The tool will read SQL statements from a directory of your choice and execute them against the open database. You must know the syntax and semantics of the SQL language to use this feature.



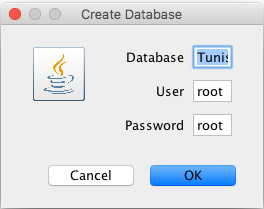
## Opening a Database

To open a database containing AMP data, choose the “Open Database…” menu item from the Database menu and fill out the resulting information about the desired database. The “Database” dropdown field contains the names of all the non-system databases available on the server.



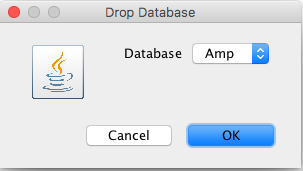
## Creating a New AMP Database

To create a fresh database, select the “Create Database…” menu item from the Database menu and fill out the fields in the resulting dialog. If the user has appropriate privileges, the database will be created on the server.



## Dropping a Database

When an AMP database is no longer useful, you can drop (delete) it by choosing the “Drop Database…” menu item from the Database menu and selecting its name from the resulting dialog. If the user has appropriate privileges, the database will be removed from the server.



## Building Database Schemas

When the user selects the “Build Database Schema” menu item, the AMP Database Tool uses each metadata file to create a new table in the database. The following illustrations show the “aircraftTypes.csv.meta” file and the table definition derived from it.

Table definitions have the following features:

1. Table name taken from the metadata “title” field
2. Columns taken from the metadata “columns” array
3. Column datatypes translated from metadata “datatype” field
4. Column comments taken from metadata “description” field
5. Primary key constraint taken from metadata “primaryKey” field
6. Foreign key constraints taken from metadata “foreignKey” array
7. One index per primary and secondary key.
8. MySQL keywords are escaped everywhere by back-tick (`) characters

{

    "id": "/fedResults/aircraftType.csv.meta",

    "title": "AircraftType",

    "scenario": "metadata-scenario",

    "classname": "ampfed.domain.entity.AircraftType",

    "ampversion": "14.13.0 alpha1",

    "tableSchema": {

        "columns": [

            {

                "name": "Model",

                "titles": "Model",

                "description": "Enumerated type enumerating the models supported by this federation",

                "datatype": "string"

            },

            {

                "name": "FedTime",

                "titles": "FedTime",

                "description": "The federation time of this interaction",

                "datatype": "float"

            },

            {

                "name": "BlockSpeed500",

                "titles": "BlockSpeed500",

                "description": "500 nm block speed in knots.",

                "datatype": "float"

        ],

        "primaryKey": "Model",

        "foreignKeys": [

            {

                "columnReference": "Model",

                "reference": {

                    "schemaReference": "/fedResults/modelTypes.csv.meta",

                    "columnReference": "ModelType"

                }

            },

            {

                "columnReference": "Size",

                "reference": {

                    "schemaReference": "/fedResults/aircraftTypeSize.csv.meta",

                    "columnReference": "AircraftTypeSize"

                }

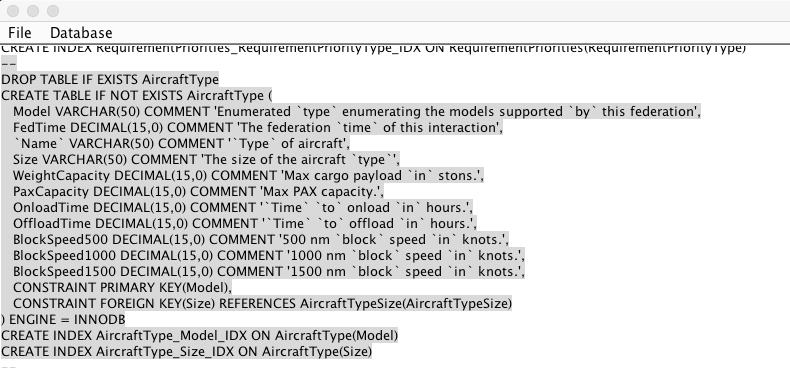
            }

        ]

    }

}

Some Columns Removed for Clarity



Metadata files that contain a “viewSchema” entry, as shown below, lead to creation of a database view. View definitions have the following features:

1. The name of the view is taken from the metadata “title” field.
2. The columns of the view are defined in the “references” list.
3. Each column is a reference to a metadata table (“schemaReference”) and a column within that table (“columnReference”).
4. A columnReference of “\*” implicitly means all the columns of the referenced table.
5. Tables can be joined within a view through the “joins” list. Each “join” element contains the two join column references (“column1” and “column2”) and an optional join comparison operator. The default comparison operator is equality (“=”).

{

    "id": "/adHoc/auto/shipBerthView.csv.meta",

    "title": "ShipBerthInfo",

    "scenario": "metadata-scenario",

    "ampversion": "14.13.0 alpha1",

    "viewSchema": {

        "references": [

            {

                "schemaReference": "/adHoc/auto/shipCargoUte.csv.meta",

                "columnReference": "Departure Day"

            },

            {

                "schemaReference": "/adHoc/auto/shipBerthTime.csv.meta",

                "columnReference": "Ship Name"

            },

            {

                "schemaReference": "/adHoc/auto/shipBerthTime.csv.meta",

                "columnReference": "Fleet"

            },

            {

                "schemaReference": "/adHoc/auto/shipBerthTime.csv.meta",

                "columnReference": "Terminal"

            },

            {

                "schemaReference": "/adHoc/auto/shipBerthTime.csv.meta",

                "columnReference": "Berth Name"

            },

            {

                "schemaReference": "/adHoc/auto/shipCargoUte.csv.meta",

                "columnReference": "\*"

            }

        ],

        "joins": [

             {

                "column1": {

                    "schemaReference": "/adHoc/auto/shipBerthTime.csv.meta",

                    "columnReference": "Ship Name"

                },

                "column2": {

                    "schemaReference": "/adHoc/auto/shipCargoUte.csv.meta",

                    "columnReference": "Ship Name"

                },

                "operator": "="

            }

        ]

    }

}

## Index and Key Creation

After the AMP Database tool has created table and view definitions, it creates all the indexes needed to implement their primary and secondary keys, then updates the table definitions to add the primary and foreign key constraints to the tables.

CREATE INDEX SustRateRules\_SubClass\_IDX ON SustRateRules(SubClass);

CREATE INDEX SustRateRules\_Priority4\_IDX ON SustRateRules(Priority4);

CREATE INDEX SustRateRules\_Priority5\_IDX ON SustRateRules(Priority5);

CREATE INDEX SustRateRules\_Priority2\_IDX ON SustRateRules(Priority2);

CREATE INDEX SustRateRules\_Priority3\_IDX ON SustRateRules(Priority3);

CREATE INDEX SustRateRules\_Priority1\_IDX ON SustRateRules(Priority1);

CREATE INDEX SustRateRules\_SupplyClass\_IDX ON SustRateRules(SupplyClass);

ALTER TABLE SustRateRules ADD PRIMARY KEY(Service,SupplyClass,SubClass);

ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Service\_IDX FOREIGN KEY (Service) REFERENCES Services(`Name`);

ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Priority1\_IDX FOREIGN KEY (Priority1) REFERENCES RateLookups(RateLookupType);

ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Priority2\_IDX FOREIGN KEY (Priority2) REFERENCES RateLookups(RateLookupType);

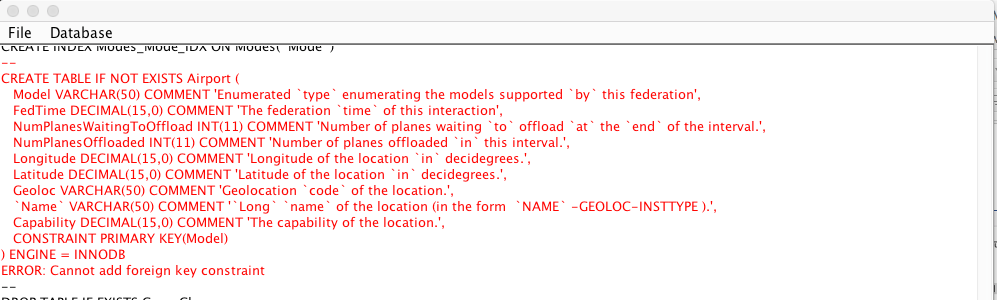
ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Priority3\_IDX FOREIGN KEY (Priority3) REFERENCES RateLookups(RateLookupType);

ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Priority4\_IDX FOREIGN KEY (Priority4) REFERENCES RateLookups(RateLookupType);

ALTER TABLE SustRateRules ADD CONSTRAINT SustRateRules\_Priority5\_IDX FOREIGN KEY (Priority5) REFERENCES RateLookups(RateLookupType);

## Error Reporting

Errors in table and view construction, and in SQL execution are reported in the progress window in red characters:

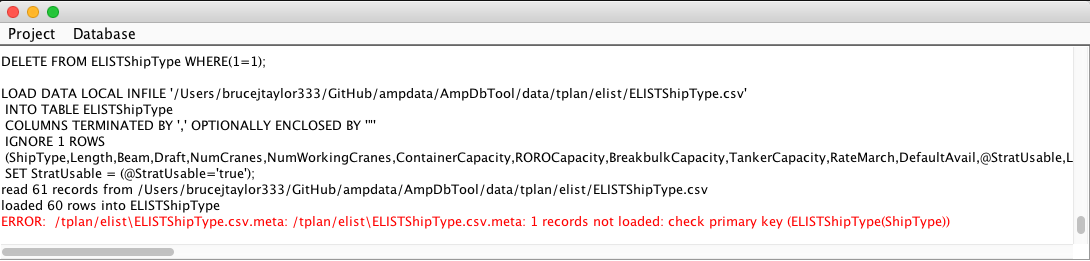


## Populating Databases

When the user selects the “Populate Database” menu item, the AMP Database Tool will attempt to populate all the tables created in schema generation. For each metadata file in the project, it attempts to locate its corresponding data (CSV) file by removing the “.meta” suffix from the metadata file name. It uses the database’s bulk load command to copy data from the CSV file into the table according to the following rules:

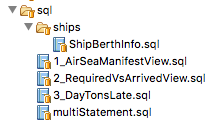
1. Referential integrity checking is suspended during data population
2. Columns in the CSV file must be in the same order as they are defined in the metadata file.
3. The CSV file is assumed to have column name definitions in the first row, but CSV column names are disregarded.
4. CSV columns are delimited by commas (,) and optionally enclosed by quotes, if the contents contain a comma.
5. Empty CSV column values translate to a null database value.
6. Boolean values (“true”, “false”) are translated into integer values (1, 0) to match the semantics of the database BOOLEAN type.
7. CSV values are taken as strings, and mapped to column datatypes according to the database’s conversion rules.

The database load operation is sensitive to primary key definition: all rows of the CSV file that have the same primary key column values will be loaded into a single row of the resulting table. This can lead to unexpected behavior, so the AMP database tool counts the rows actually loaded in the table and compares it to the number of data rows in the CSV file. If these numbers do not match, the tool reports an error and a suggestion that the user re-evaluate the primary key definition for the table to see if it is too restrictive.



## Executing Custom SQL

Some database tables, and some database operations cannot be implemented using the metadata process described above. For these situations, the AMP database tool can execute custom SQL statements stored in a directory structure, as shown below.



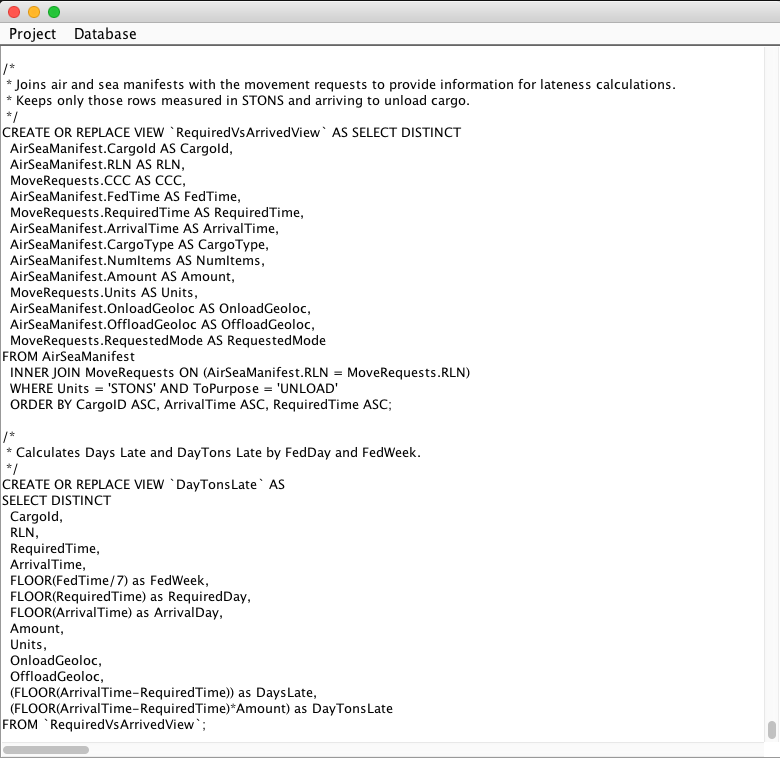
To execute the SQL statements, select the “Execute Custom SQL…” menu item of the Database menu, and select one or more SQL files or directories containing SQL files. The content of the SQL files will be evaluated according to the following rules:

1. Depth-first traversal of nested directories
2. Within a directory, in text collation order.

You can use this ordering to arrange for views to be created after their dependents, by assigning a prefix to file names. So, in the above example, the SQL files will be executed in the following order:

1. ShipBerthinfo.sql
2. 1\_AirSeaManifestView.sql
3. 2\_RequiredVsArrivedView.sql
4. 3\_DayTonsLate.sql
5. multiStatement.sql

When each SQL file is read, the AMP Database Tool will execute its contents as one or more SQL statements, separated by semicolons (“;”). The statements executed, and any error messages, will appear in the progress window, as shown below.



## Properties

For user convenience, the AMP Database Tool records the following information in the file “AMP.properties” in the user’s home directory.

1. Most recent project opened, as path to the project’s root directory
2. Most recent database opened, including user name and password.

## This information is used to prepopulate project and database selection dialogs. The AMP.properties file can safely be deleted at any time.

#

#Fri Apr 21 10:59:17 CDT 2017

db.password=root

db.user=root

db.name=Amp

proj.root=/Users/brucejtaylor333/Argonne/data

## Error Messages

1. “<metadata>: table and view definition missing”   
   The metadata file has neither a “tableSchema” nor “viewSchema” section.
2. “<metadata>: records not loaded: check primary key <key>”  
   The number of rows loaded into the table does not match the number of records in the CSV file.
3. “<metadata>: column lists do not match for load”   
   The column headers of the CSV file do not match the column definitions of the table.
4. “<metadata>: Table name <table> duplicates table name in <metadata>”  
   Table names within the database must be unique.
5. “<metadata>: foreign key column is null”  
   The column name of a foreign key is null or empty.
6. “<metadata>: foreign key column is not defined”  
   The column name of a foreign key does not appear in the referenced table.
7. “<metadata>: primary key has no columns”  
   The primary key of a table has no identified columns.
8. “<metadata>: duplicate column <column>”  
   The specified column name appears more than once in the table definition.
9. “<metadata>: primary key column is null”  
   The primary key column is null or has an empty string value.
10. “<metadata>: primary key column is not defined”  
    The primary key column does not exist in its parent table.
11. “<metadata>: foreign key column is null”  
    A foreign key has a null or empty column specification
12. “<metadata>: foreign key column <column> is not defined”  
    A foreign key has a column specification that does not exist in the referenced table.
13. “<metadata>: foreign key reference table <table> is not defined”  
    A foreign key reference table is not defined in the database.
14. “<metadata>: foreign key reference table <table> is not a table”  
    A foreign key reference is to a view, not a table.
15. “<metadata>: foreign key reference column is not defined”  
    A foreign key reference column does not exist in its referenced table.
16. “<metadata>: primary key column <column> does not exist”  
    A primary key column does not exist in its parent table.
17. “<metadata>: view reference <table> does not exist”  
    A view column references a non-existent table.
18. “<metadata>: view reference <table> is not a table”  
    A view column references a view rather than a table.
19. “<metadata>: view reference <column> does not exist”  
    A view column references a column that does not exist.
20. “Skipping creation of <metadata>”  
    Because of previous errors, the specified table or view will not be created.