The Emissions Inventory Maintenance Application

Jo Ellen Brandmeyer, Aaron Parks, Sunil Rao, and Robert Zerbonia
RTI International
3040 Cornwallis Rd, PO Box 12194, Research Triangle Park, NC 27709-2194
brandmeyer@rti.org

ABSTRACT

The Air Quality Management Decision Support System (AQMDSS) is a comprehensive set of software tools that includes the complete data path from emission inventory development through emissions processing, air quality modeling, and data analysis. The AQMDSS uses the Multimedia Integrated Modeling System (MIMS) as the user interface for setting up and executing the Sparse Matrix Operator Kernel Emissions (SMOKE) processing system, the AMS/EPA Regulatory Model (AERMOD) and its AERMAP and AERMET preprocessors, the Models-3/Community Multiscale Air Quality (CMAQ), and other data preparation and analysis tools.

This paper is an update to one that was presented at the 14th Annual Emission Inventory Conference in Las Vegas, NV in 2005.[1] At that time, the design phase was complete and implementation was proceeding. The AQMDSS is now installed at the Beijing Municipal Environmental Protection Bureau (BMEPB) in Beijing, China.

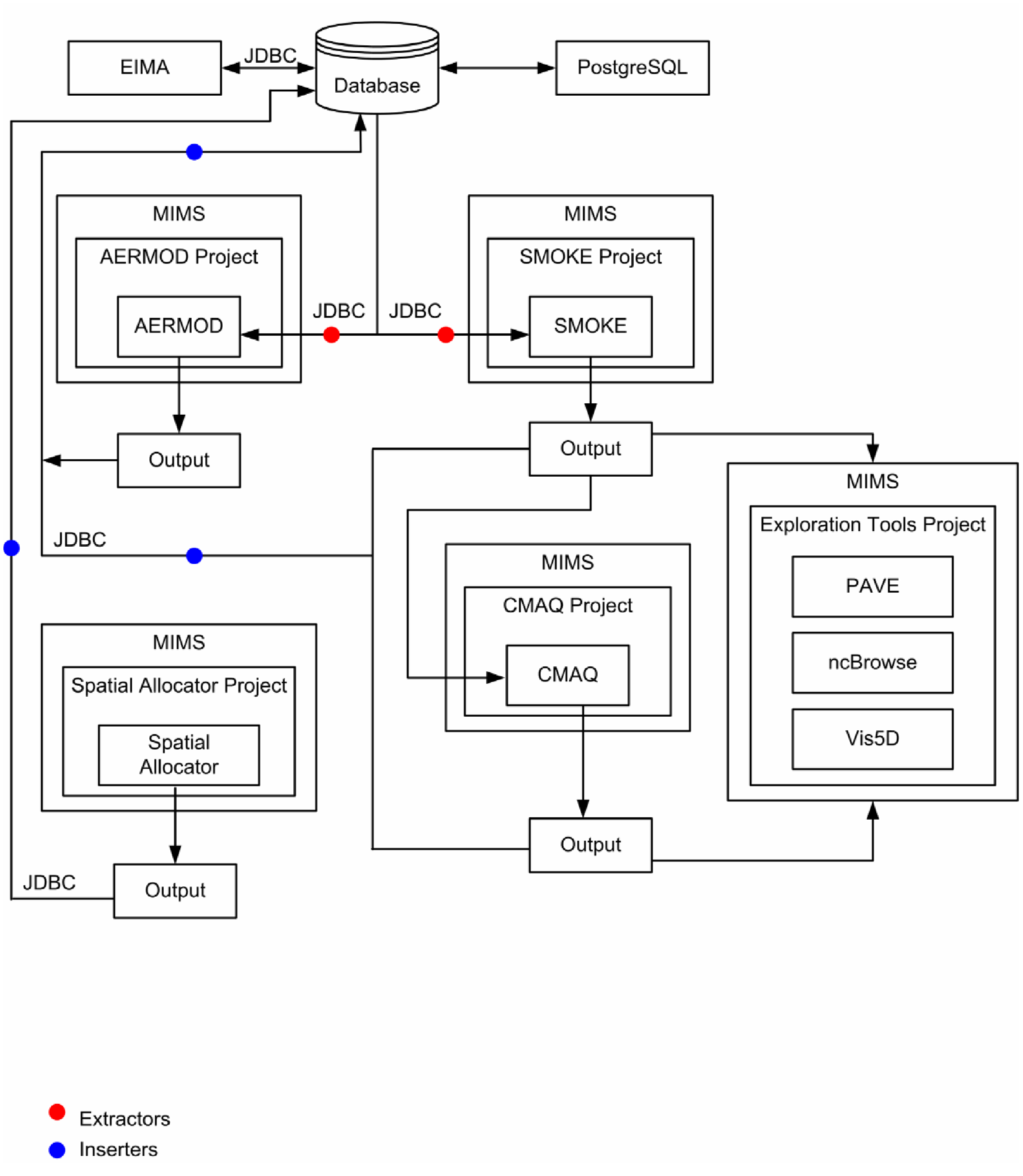
First, this paper focuses on the Emissions Inventory Maintenance Application (EIMA) of the AQMDSS. Key features include a relational database implementation of an extended National Emission Inventory (NEI) Input Format (NIF) version 3; menu-driven, multilingual user interface that supports data integrity; security for user access of data; and direct access to data tables via Structured Query Language (SQL).

Next, this paper describes how the EIMA was implemented at the BMEPB. We will report our experience with performing bulk data loads into the NIF 3.0 data format using SQL.

INTRODUCTION

The Air Quality Management Decision Support System (AQMDSS) was built for the Beijing Municipal Environmental Protection Bureau (BMEPB) of Beijing Municipal Government (BMG) in the People’s Republic of China. Figure 1 illustrates the schematic of the primary components in the completed system.[2]

Figure 1. Primary system schematic of the Air Quality Management Decision Support System. (From RTI International, “System Analysis and Design Report for the Air Quality Management Decision Support System – Revised Version,” Prepared for PA Consulting Group by RTI International, Research Triangle Park, NC 2004.)



The AQMDSS was designed and implemented by RTI International as an integrated software system to support air quality modeling at multiple scales. The initial design was to integrate existing software – models and associated tools – into a common framework, which would provide users with a friendly, graphical user interface (GUI).[3] Also, data would be entered and stored in a database, from which data files would be generated as needed by the models. No changes would be made to the models to ease upgrading as newer versions of models were released in the future.

When gathering the models and tools, however, we were not able to locate a publicly available application for building and maintaining the emissions inventory data. Instead of a readily available tool, we found that a wide variety of tools had been created by various organizations (e.g., States) over the years. BMEPB, however, needed a way to build a new emissions inventory and store it in the database. To fill this need we created the Emissions Inventory Maintenance Application (EIMA).

The next section discusses the main features of the EIMA. Then, we present our experience with installing the EIMA on the computer system in Beijing and performing bulk data loads into the database. At the end we present our conclusions.

MAIN FEATURES OF THE EIMA

System requirements of the AQMDSS impacted the design and implementation of the EIMA. These included a user interface in both Simplified Chinese and English, data access limited to authorized users, and data integrity for processing the data with the Sparse Matrix Operator Kernel Emissions (SMOKE) processing system. It also needed to support the parent/child relationships among data tables in the database, while being flexible to support future additions to the data structure. The remainder of this section discusses the main features of the EIMA.

Flexibility

The components of the AQMDSS are applications that are periodically updated. For example, SMOKE typically has at least one new release each year. To accommodate updates as quickly as possible, the AQMDSS and the EIMA must be extremely flexible.

Maximizing flexibility requires minimizing the changes required in the AQMDSS when one model is changed. To meet this need the AQMDSS was built using object-oriented design of reusable components. For example, if the data structure for one model input file is changed, then the only components that must be changed are the one extractor for that data file, the data elements in the database that support that data file, and the EIMA screens for those data elements.

User Interface

The BMEPB required the user interface to be in Simplified Chinese. Contractors and other consultants from outside China, however, primarily used English as their first language. An early requirement, therefore, was that the EIMA have a multilingual user interface.

The Java programming language includes support for internationalization/localization (i.e., i18n/l10n). This construct allows the text displayed by the program to be stored separately from the program itself. The EIMA was constructed using this construct, so text such as the titles of frames and the descriptions of input fields are stored in text files.

Currently, the EIMA is distributed with 2 sets of these files – English and Simplified Chinese. The language that is displayed is determined by the user's environment, as defined for his user ID in the operating system. English is the default language, so if the user ID is set up for Simplified Chinese then that is how the screens are displayed, otherwise English is used. To support another language, such as Spanish, a new set of text files need to be created for that language. The screens would then be displayed in that new language for any user ID that uses it. No changes are required for the program itself, and only one version of the program must be maintained.

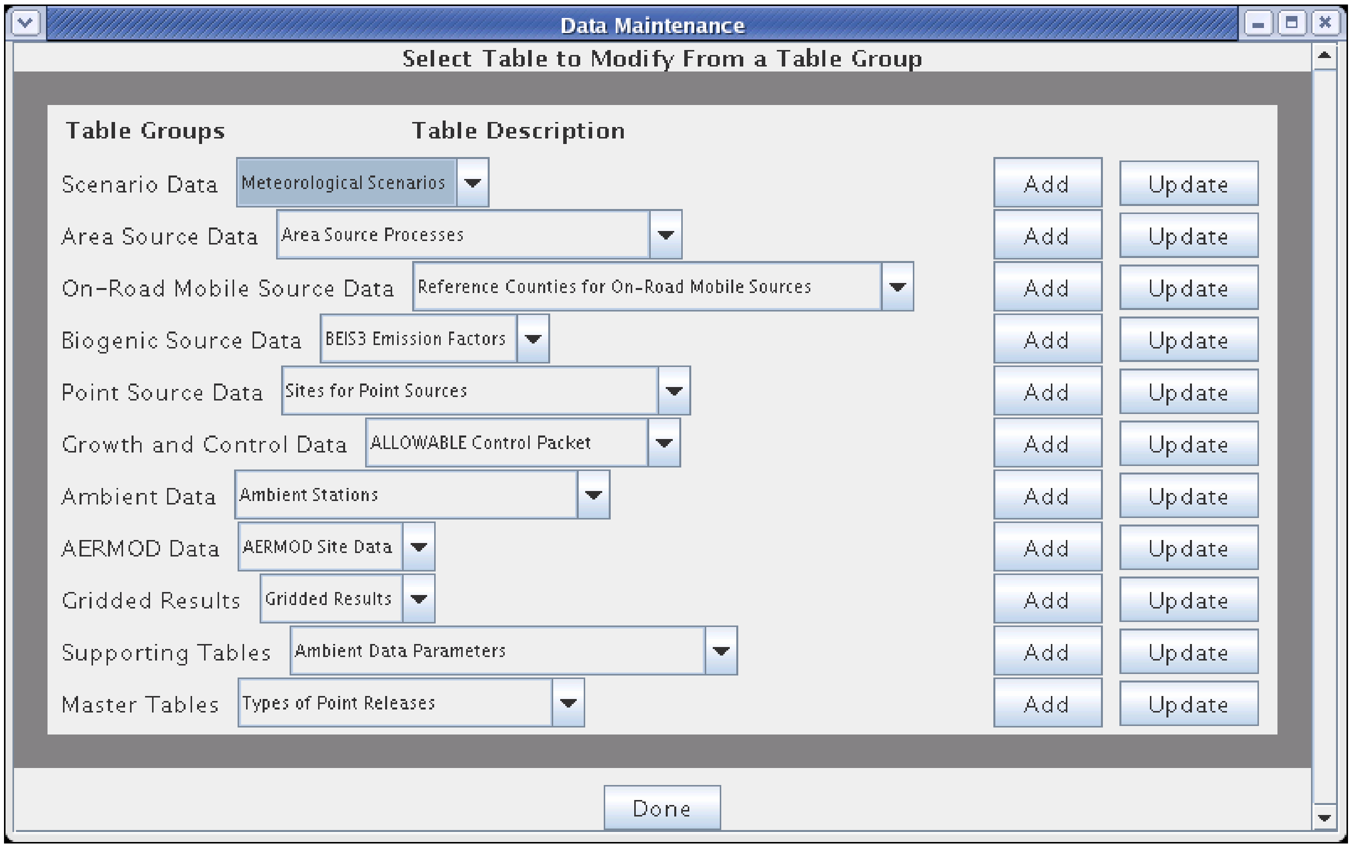
Types of Users

The AQMDSS supports many different types of users. Each type of user performs a specific job function. For example, some users may analyze model results and create visualizations but may not execute models. These users would need access to only the visualization tools and model output files. Similarly, a user who works with AERMOD for industrial-scale modeling would not need access to SMOKE.

The structure of EIMA also supports users by type. Each user ID authorized for access to the AQMDSS is assigned a user type. The most restricted user type performs only visualization work, while the most expert type has access to all functions of the entire AQMDSS. Some of these users also need access to the EIMA. So, a user ID designated as *area source modeler* has access to data entry and update screens for data tables related to area sources, but cannot enter data for point sources.

The EIMA menu for each user is constructed such that only the parts of the system for which he/she has access are shown. Other parts cannot be accessed through the EIMA screens. This is the first level of data security. Figure 2 shows the data maintenance screen for an *expert* user.

Figure 2. Data maintenance screen for user type *expert*.


Data Maintenance

Select Table to Modify From a Table Group

Table Groups	Table Description		
Scenario Data	Meteorological Scenarios	Add	Update
Area Source Data	Area Source Processes	Add	Update
On-Road Mobile Source Data	Reference Counties for On-Road Mobile Sources	Add	Update
Biogenic Source Data	BEIS3 Emission Factors	Add	Update
Point Source Data	Sites for Point Sources	Add	Update
Growth and Control Data	ALLOWABLE Control Packet	Add	Update
Ambient Data	Ambient Stations	Add	Update
AERMOD Data	AERMOD Site Data	Add	Update
Gridded Results	Gridded Results	Add	Update
Supporting Tables	Ambient Data Parameters	Add	Update
Master Tables	Types of Point Releases	Add	Update

Done

Data Security

Security must be set up very carefully. Each user must be allowed access to the parts of the AQMDSS that are appropriate to their job functions, but not to other parts. Three levels of security are provided with the AQMDSS, as follows.

- The operating system is responsible for validating that the user has access to the computer system. This is accomplished via user ID/password. When a systems administrator creates an account for a user, he/she also places the user into one or more groups and defines the parts of the system to which that user has access. The operating system then grants access to directories, files, and programs based on the permissions for that user ID.

- The EIMA is responsible for validating that the user has access to that application. This is accomplished via user ID and password for the underlying database of the AQMDSS. An expert user defines each user in the database and assigns a user type. The EIMA constructs the data maintenance menu for each user based on his assigned type.

- PostgreSQL is responsible for validating the user's access to the database and individual data tables. An expert user (i.e., a database administrator) creates a user account for the database and assigns the user to one or more database groups. PostgreSQL has a list of permissions which may be granted for each table in a database, and the database administrator defines which permissions are to be given to each user or group. PostgreSQL then allows or blocks the various types of access to its tables.

The person responsible for maintaining database security needs to grant permissions consistently for the EIMA and for PostgreSQL. For example, a new user may be assigned user type *area source modeler*, so the EIMA gives that user access to enter data into area source-related data tables. But, if that user is not also granted permission within PostgreSQL to all of those data tables, then PostgreSQL will block some or all of that user's activities with respect to some tables.

Data Entry and Validation

The data structure of an emission inventory is very complex. Many items are represented by numeric or alphanumeric codes (e.g., pollutant code, facility ID). Certain codes may appear to be numeric, but cannot be stored that way because some valid data values have a leading zero (e.g., state and county FIPS codes). Other fields must contain only certain valid values (e.g., less than or equal to seven days per week). These types of data validation are set up within the EIMA, so restricted data values are validated when they are entered.

For some data, the information's structure is too complex to exist in one data table; it is split into related tables. A good example of this is a point source. One facility is a point source, but it may have multiple processes, with multiple emission points, control equipment, and pollutants. The NIF v3.0 data format handles this situation by dividing the data elements for one point source into data records residing in multiple, related data tables using a relational design.

The AQMDSS database encompasses the NIF v3.0 relational design and implements it in PostgreSQL. Relationships between tables are explicitly defined in the data structure using primary keys, foreign keys, and unique keys. Returning to the example of a point source, the keys defined in its tables form parent/child relationships. The facility is the parent of processes (i.e., a process is a child of the facility) and must have a record in the appropriate data table before any processes can be entered for that facility.

The EIMA supports the NIF-3 structure. A user cannot directly enter data for a child table. Instead, the record in the parent table must first be created, completed, and saved; then data can be entered for a child table. Also, data in fields that identify a record in the database, such as a facility ID, cannot be changed after committing the record to the database. This feature prevents orphaned records (i.e., a child record whose link to its parent record is broken)

User Notes

The emission inventory can be expected to frequently change. For example, errors from the past may be found and corrected, control equipment for a source may be changed, or emission factors may be updated. The EIMA includes a feature that enables the user to document these changes when making them.

Each data entry screen includes a button for the user to enter a free-form note. When a value is changed, the user may want to record the old value, the reason for the change, and the source of the new value. By keying this information into a note that is stored in the database and attached to the specific data record, this type of information can be captured. The EIMA automatically records the ID of the user who created the note and the date and time of the entry, allowing a series of notes to be associated with any data record.

SET UP AT BMEPB

Set up of the AQMDSS and specifically the EIMA at Beijing Municipal Environmental Protection Bureau in Beijing was done by a team consisting of two RTI team members, the Beijing Technical Expert and numerous BMEPB employees. All together a group of roughly 10 people worked on the system at one point or another during the week long setup. Many of the people involved had an interest in a particular component of the system (e.g., GIS) and participated only in the part they would eventually use.

BMEPB had purchased 4 Pentium Dual-Processors 64 Bit Xenon rack-mount servers, 2.4 GHZ with HyperThreading. Each machine had 4 GB of RAM and a 200 GB SCSI hard drive. Extra drive space was available through a Samba mount, however it was not needed. Each machine was connected to a Gigabit switch. The Internet was not connected to the BMEPB computational machine.

Redhat AS 4 was installed by the Chinese expert upon arrival of the RTI team. All work on the machines was done using the Simple Chinese character set. X-Windows, PostgreSQL, and several Linux utilities were installed by the Redhat installation media and were not altered.

The climate controlled section of the lab held numerous servers including the ones used for this project. Some work was done standing at the console; however, the team usually sat at a console in the lab using VNC Viewer on a windows XP machine.

Java plays a crucial role in setting up the system. The client had wanted to use the SUN Hotspot java edition, but we found upon testing at the site that it was incompatible with MIMS. The reason for this was never determined. Instead, we used the SUN J2SE 64-bit version tested by RTI. Due to Linux's isolation of libraries, this required a simple change to the $JAVA_HOME environment variable. Also simplified was the transition from 32-bit to 64-bit. We switched seamlessly between 32- and 64-bit for development and testing using the SUN JVM.

NIF3 Data Format

Examination of the NIF3 database format implemented by RTI shows that most tables have a dependency on one or more tables that must be filled before population can occur. These dependencies are enforced by the database structure and cannot be ignored when importing data.

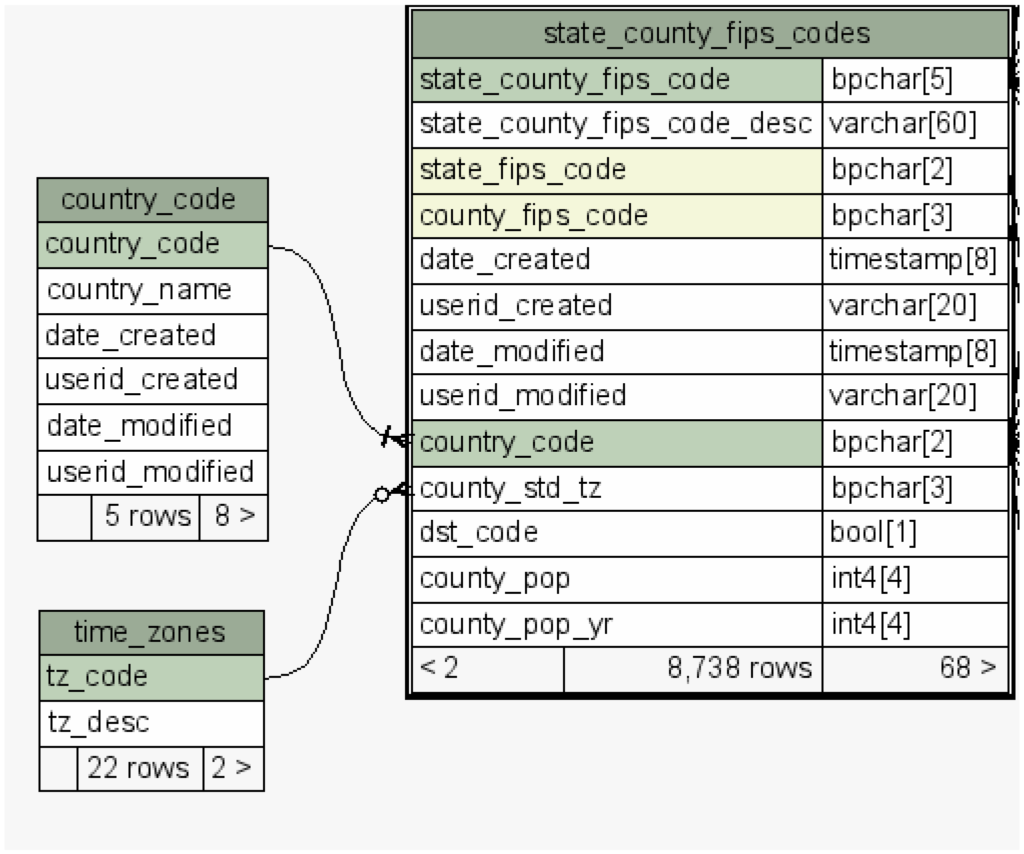
Populating Data Tables for Point Sources

For example if a user had some point source emission data for SMOKE that they wanted to enter and later extract for the PTINV file, they would need to fill a series of tables that would ultimately uniquely identify each data point. Along the way they would identify such information as the location in space, the facility id, the source, and ultimately the pollutant.

The following pages go through an example of starting at the highest level table (state_county_fips_codes) filling dependencies before filling dependent tables. Several figures are included to illustrate the layout of the database as well as usage of the EIMA. All diagrams for the database are created by the SCHEMASpy documentation system (http://schemaspy.sourceforge.net/).

At the top level are the country_code, time_zones, and state_county_fips_code tables. Figure 3 shows the relationship between the top level tables.

Figure 3. The state_county_fips_codes table is the first step in identifying point source information.

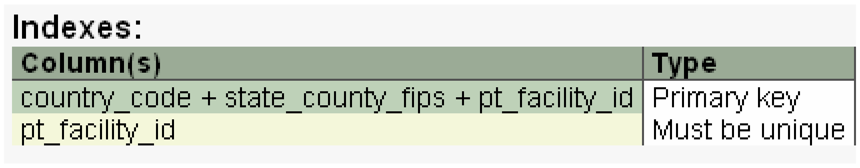


When used together, these tables identify the area in which the facility is located. While this works wonderfully in the United States the RTI team found that it could be adapted to other organizational structures. For instance the Chinese use province/state organizations. Working in concert with BMEPB we mapped the provinces to the state FIPS and district to the county FIPS. While slightly artificial it proved to be a good compromise for importing data.

Included in most tables are 4 fields: date_created, userid_created, date_modified, and userid_modified. These fields are used by the EIMA in conjunction with security to help identify who and when changes are made.

After filling the location tables, a series of facilities must be entered into the point_source_site table. Looking at the keys for the point_source_site table we see that it references state_county_fips. Therefore any data previously entered for the state_county_fips table will satisfy relevant dependencies for the point_source_site table, as illustrated in Figure 4.

Figure 4. Keys for the point_source_site table.

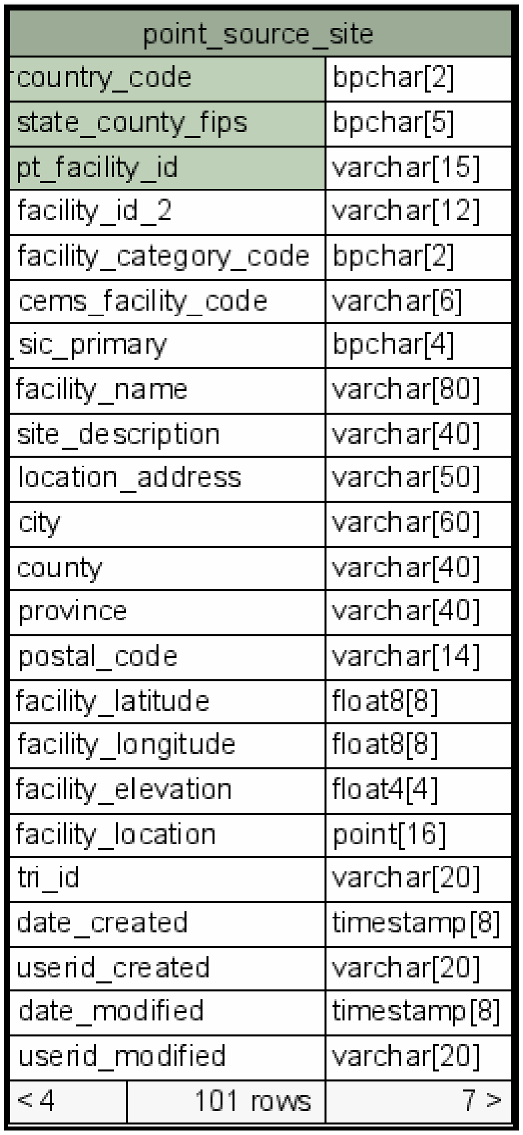


Indexes:

Column(s)	Type
country_code + state_county_fips + pt_facility_id	Primary key
pt_facility_id	Must be unique

Though not strictly required, each table has several pieces of supporting information to make the information in the database more useful. These fields often link into other tables to help narrow the scope of what can be entered into the database. For example, the facility_category_code links to the facility_category table. Thus any entries into that field must reside in the facility_category table. Additional fields are shown in Figure 5.

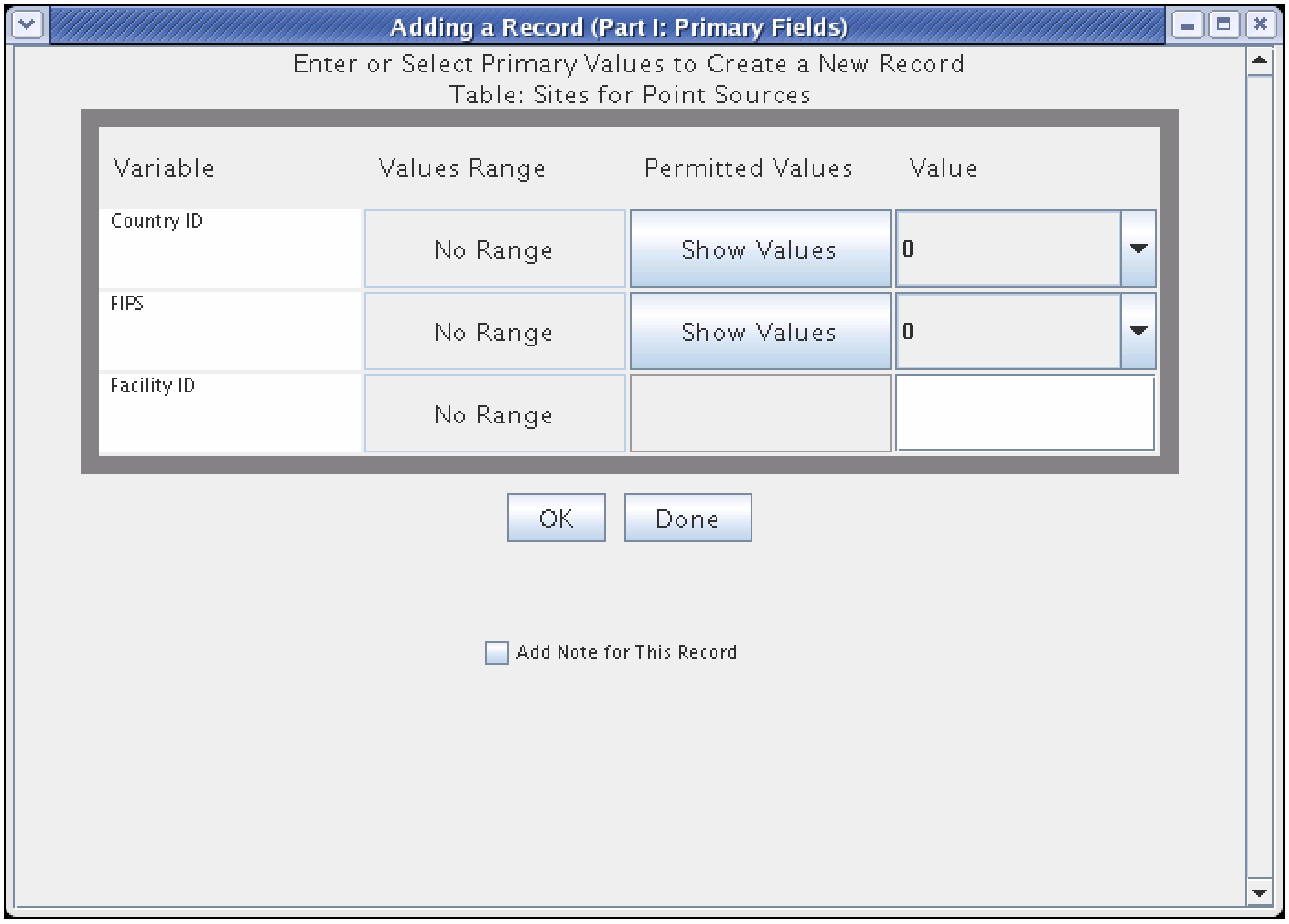
Figure 5. Supporting information in the point_source_site table.



point_source_site	
country_code	bpchar[2]
state_county_fips	bpchar[5]
pt_facility_id	varchar[15]
facility_id_2	varchar[12]
facility_category_code	bpchar[2]
cems_facility_code	varchar[6]
sic_primary	bpchar[4]
facility_name	varchar[80]
site_description	varchar[40]
location_address	varchar[50]
city	varchar[60]
county	varchar[40]
province	varchar[40]
postal_code	varchar[14]
facility_latitude	float8[8]
facility_longitude	float8[8]
facility_elevation	float4[4]
facility_location	point[16]
tri_id	varchar[20]
date_created	timestamp[8]
userid_created	varchar[20]
date_modified	timestamp[8]
userid_modified	varchar[20]
< 4 101 rows	7 >

The EIMA requires that the user enter all required information before moving on to secondary fields. Looking at point_source_site table specifically we see in Figure 6 that the three required fields of country_id, fips, and facility_id are input first. Both country_id and fips are drop-down boxes so that the user can use only previously entered values (from the appropriate table).

Figure 6. Inputting required information into the EIMA.

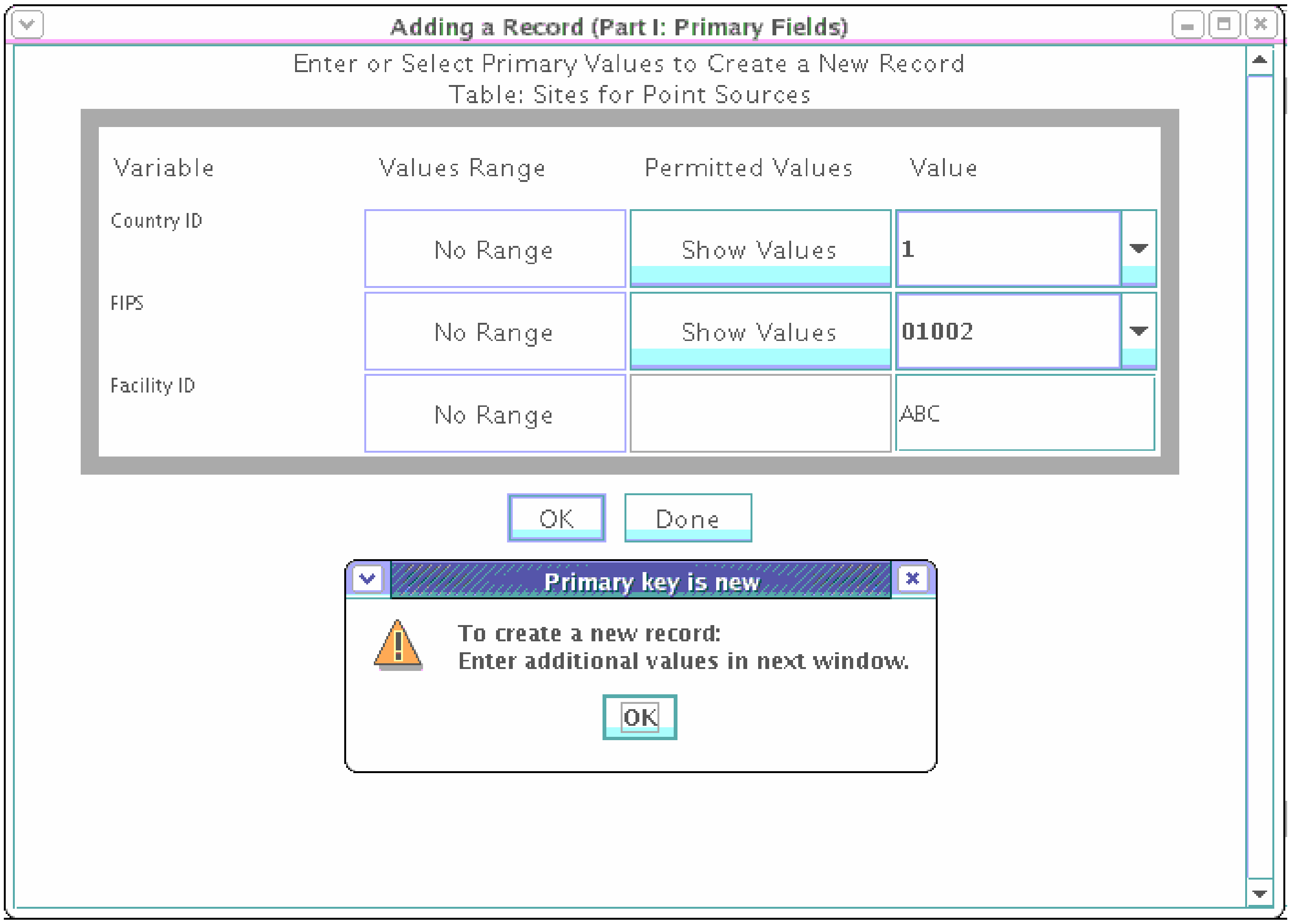


Enter or Select Primary Values to Create a New Record
Table: Sites for Point Sources

Variable	Values Range	Permitted Values	Value
Country ID	No Range	Show Values	0
FIPS	No Range	Show Values	0
Facility ID	No Range		

When the user indicates completion, then the database checks each constraint to be sure it is a valid selection. If no errors are found, then the value is written to the database and the user is notified of a successful write as seen in Figure 7.

Figure 7. When valid data are entered, the data are written to the database.



Adding a Record (Part I: Primary Fields)

Enter or Select Primary Values to Create a New Record
Table: Sites for Point Sources

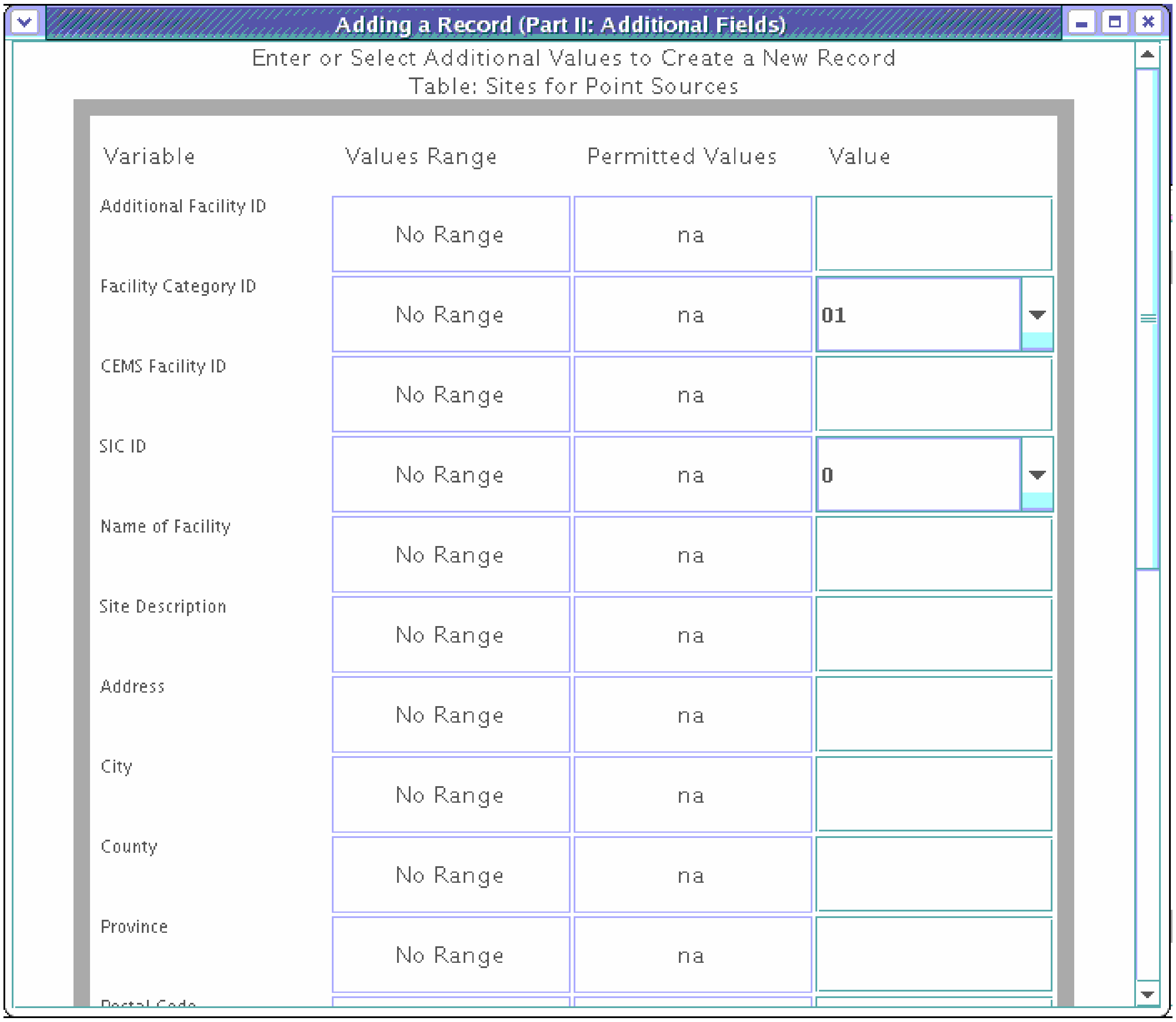
Variable	Values Range	Permitted Values	Value
Country ID	No Range	Show Values	1
FIPS	No Range	Show Values	01002
Facility ID	No Range		ABC

Primary key is new

To create a new record:
Enter additional values in next window.

A final window comes up with the rest of the fields in the table as seen in Figure 8. The Value Range and Permitted Values fields show any constraints that the database may impose on entered values. Value ranges might be something like 0-100 while Permitted Values will be types of data such as integers or characters.

Figure 8. More data to be entered for point_source_site table.



Adding a Record (Part II: Additional Fields)

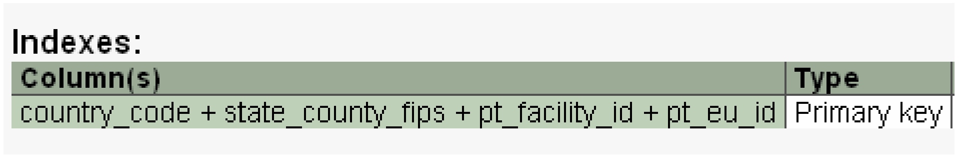
Enter or Select Additional Values to Create a New Record
Table: Sites for Point Sources

Variable	Values Range	Permitted Values	Value
Additional Facility ID	No Range	na	
Facility Category ID	No Range	na	01
CEMS Facility ID	No Range	na	
SIC ID	No Range	na	0
Name of Facility	No Range	na	
Site Description	No Range	na	
Address	No Range	na	
City	No Range	na	
County	No Range	na	
Province	No Range	na	

Notice that field for facility_category_ID presents a drop down list that links to facility_category table. This ensures that only values in that table can be selected, forcing consistency throughout all records for this table.

After entering data for a facility in the point_source_site table, each emission unit may be added to the point_source_er table. As shown in Figure 9, the values for the country_code, state_county_fips, and pt_facility_id are carried along to the primary key of this table. The value of pt_eu_id is added to the primary key, allowing multiple emission units to be entered for each facility.

Figure 9. Keys for point_source_er.

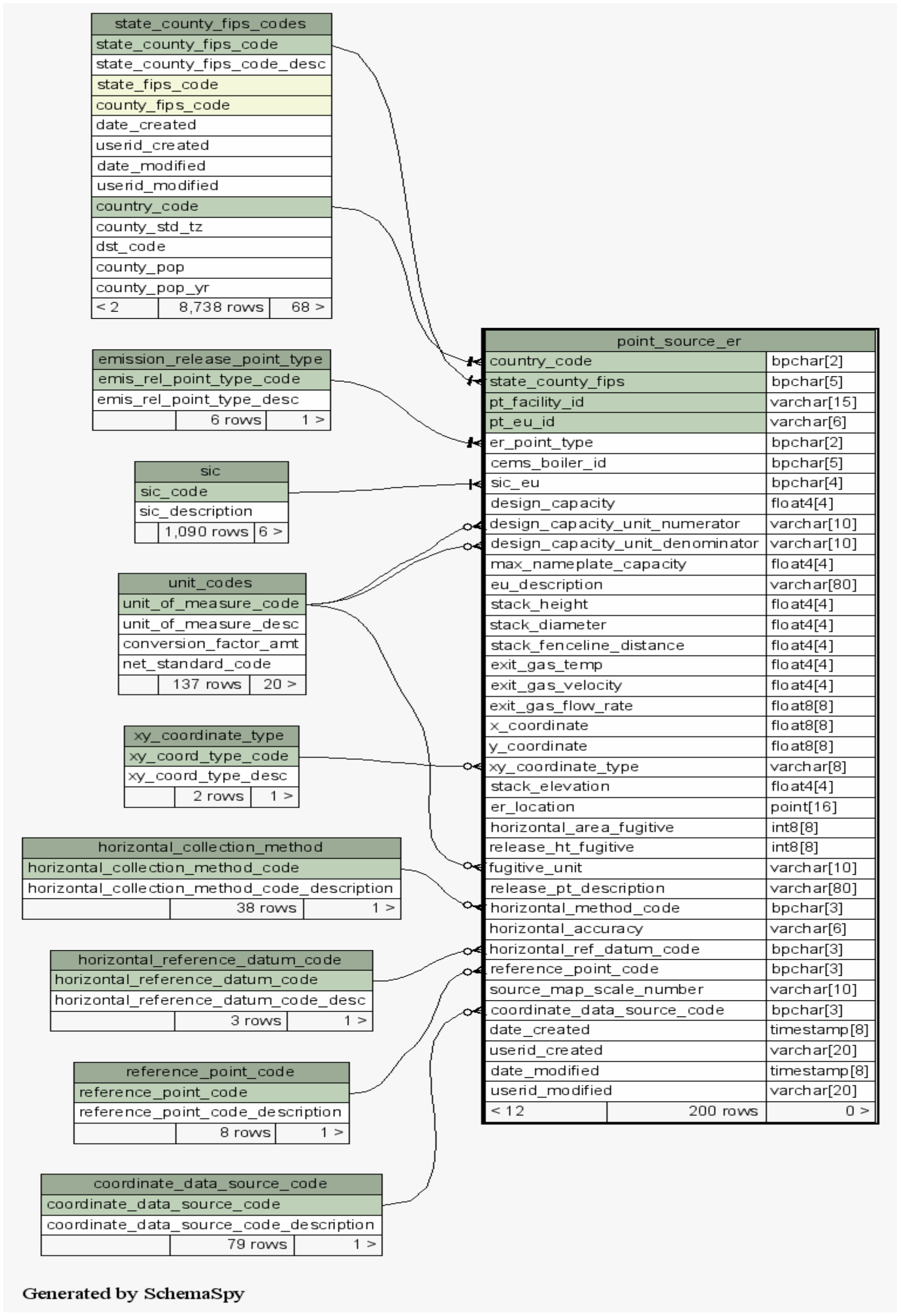


Indexes:

Column(s)	Type
country_code + state_county_fips + pt_facility_id + pt_eu_id	Primary key

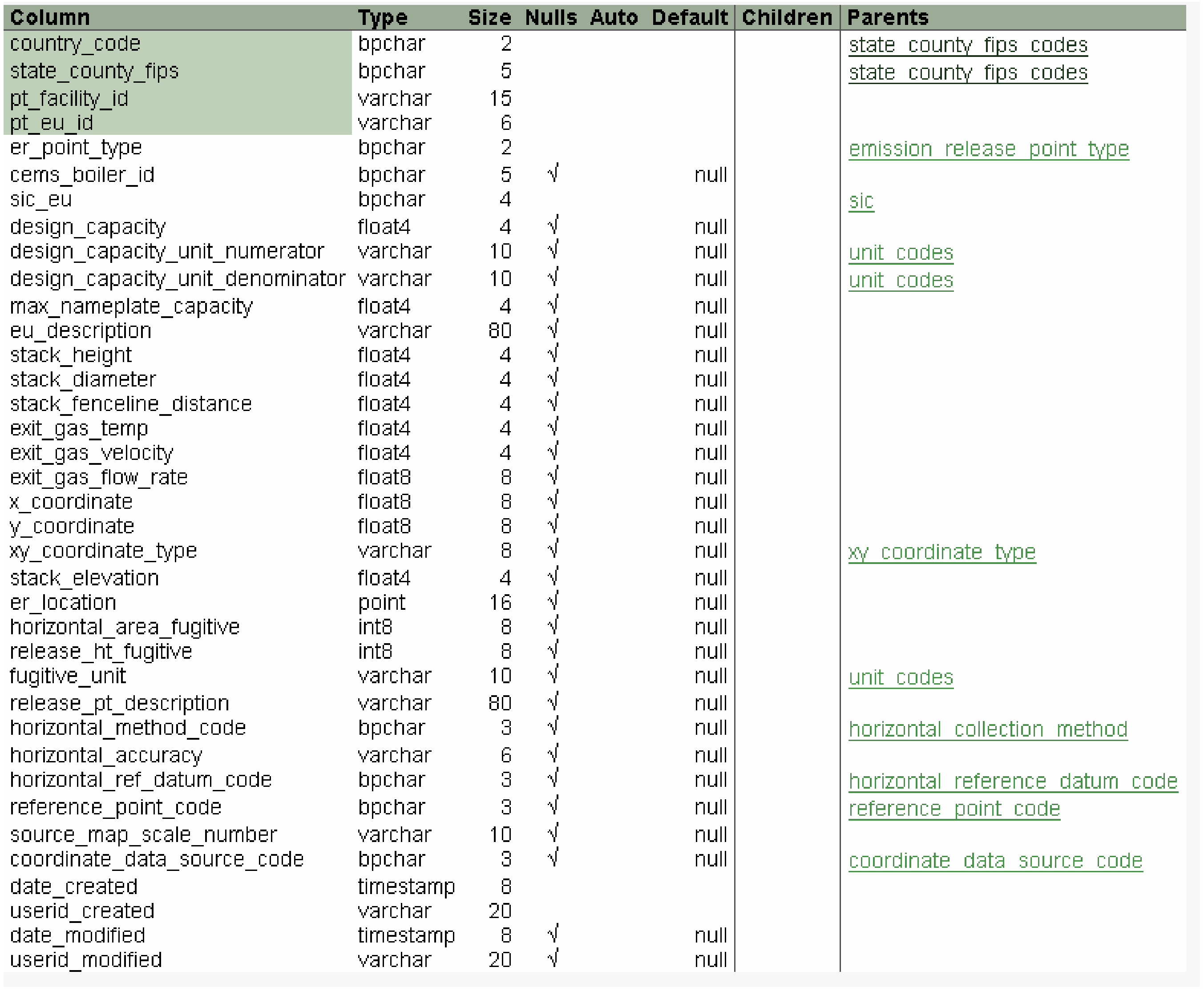
Looking at the table for point_source_er in Figure 10 we see that quite a few non-key dependencies have been added. Most tables in the database ultimately have many dependencies. Understanding these is the key to successfully entering data.

Figure 10. More linkages are added as needed to help fill tables.


state_county_fips_codes
state_county_fips_code
state_county_fips_code_desc
state_fips_code
county_fips_code
date_created
userid_created
date_modified
userid_modified
country_code
county_std_tz
dst_code
county_pop
county_pop_yr
< 2
8,738 rows
68 >
emission_release_point_type
emis_rel_point_type_code
emis_rel_point_type_desc
6 rows
1 >
sic
sic_code
sic_description
1,090 rows
6 >
unit_codes
unit_of_measure_code
unit_of_measure_desc
conversion_factor_amt
net_standard_code
137 rows
20 >
xy_coordinate_type
xy_coord_type_code
xy_coord_type_desc
2 rows
1 >
horizontal_collection_method
horizontal_collection_method_code
horizontal_collection_method_code_description
38 rows
1 >
horizontal_reference_datum_code
horizontal_reference_datum_code
horizontal_reference_datum_code_desc
3 rows
1 >
reference_point_code
reference_point_code
reference_point_code_description
8 rows
1 >
coordinate_data_source_code
coordinate_data_source_code
coordinate_data_source_code_description
79 rows
1 >
point_source_er
country_code bpchar[2]
state_county_fips bpchar[5]
pt_facility_id varchar[15]
pt_eu_id varchar[6]
er_point_type bpchar[2]
cems_boiler_id bpchar[5]
sic_eu bpchar[4]
design_capacity float4[4]
design_capacity_unit_numerator varchar[10]
design_capacity_unit_denominator varchar[10]
max_nameplate_capacity float4[4]
eu_description varchar[80]
stack_height float4[4]
stack_diameter float4[4]
stack_fenceline_distance float4[4]
exit_gas_temp float4[4]
exit_gas_velocity float4[4]
exit_gas_flow_rate float8[8]
x_coordinate float8[8]
y_coordinate float8[8]
xy_coordinate_type varchar[8]
stack_elevation float4[4]
er_location point[16]
horizontal_area_fugitive int8[8]
release_ht_fugitive int8[8]
fugitive_unit varchar[10]
release_pt_description varchar[80]
horizontal_method_code bpchar[3]
horizontal_accuracy varchar[6]
horizontal_ref_datum_code bpchar[3]
reference_point_code bpchar[3]
source_map_scale_number varchar[10]
coordinate_data_source_code bpchar[3]
date_created timestamp[8]
userid_created varchar[20]
date_modified timestamp[8]
userid_modified varchar[20]
< 12
200 rows
0 >
Generated by SchemaSpy

However, a closer look at the table definition in Figure 11 shows that the only fields that cannot be null, in addition to those comprising the primary key, are sic_eu, user_id_created, and date_created. This means that we will have to go back and fill only the SIC table with each sic_eu we intend to use.

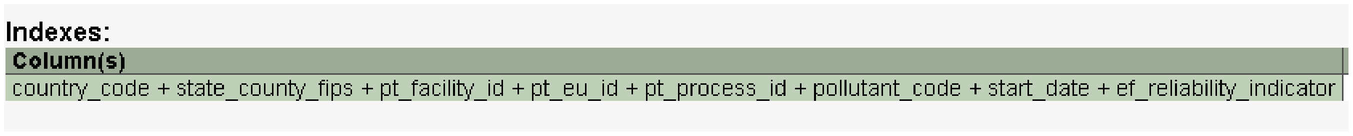
Figure 11. Table definition for point_source_er.



Column	Type	Size	Nulls	Auto	Default	Children	Parents
country_code	bpchar	2					state_county_fips_codes
state_county_fips	bpchar	5					state_county_fips_codes
pt_facility_id	varchar	15					
pt_eu_id	varchar	6					
er_point_type	bpchar	2					emission_release_point_type
cems_boiler_id	bpchar	5	√		null		
sic_eu	bpchar	4					sic
design_capacity	float4	4	√		null		
design_capacity_unit_numerator	varchar	10	√		null		unit_codes
design_capacity_unit_denominator	varchar	10	√		null		unit_codes
max_nameplate_capacity	float4	4	√		null		
eu_description	varchar	80	√		null		
stack_height	float4	4	√		null		
stack_diameter	float4	4	√		null		
stack_fenceline_distance	float4	4	√		null		
exit_gas_temp	float4	4	√		null		
exit_gas_velocity	float4	4	√		null		
exit_gas_flow_rate	float8	8	√		null		
x_coordinate	float8	8	√		null		
y_coordinate	float8	8	√		null		
xy_coordinate_type	varchar	8	√		null		xy_coordinate_type
stack_elevation	float4	4	√		null		
er_location	point	16	√		null		
horizontal_area_fugitive	int8	8	√		null		
release_ht_fugitive	int8	8	√		null		
fugitive_unit	varchar	10	√		null		unit_codes
release_pt_description	varchar	80	√		null		
horizontal_method_code	bpchar	3	√		null		horizontal_collection_method
horizontal_accuracy	varchar	6	√		null		
horizontal_ref_datum_code	bpchar	3	√		null		horizontal_reference_datum_code
reference_point_code	bpchar	3	√		null		reference_point_code
source_map_scale_number	varchar	10	√		null		
coordinate_data_source_code	bpchar	3	√		null		coordinate_data_source_code
date_created	timestamp	8					
userid_created	varchar	20					
date_modified	timestamp	8	√		null		
userid_modified	varchar	20	√		null		

The last table for examination and population is point_source_em. This table holds all the details on the actual emissions for the point source. Looking at this primary key we see that it has been built upon the previous tables by adding a few more fields to handle uniqueness criteria (Figure 12).

Figure 12. Keys for point_source_em.



Indexes:

Column(s)
country_code + state_county_fips + pt_facility_id + pt_eu_id + pt_process_id + pollutant_code + start_date + ef_reliability_indicator

Looking at the table definition we see that start_date is a table key much like pt_process_id that is only contained in point_source_em. The other two are in separate tables that need to be populated.

By following the linkages of tables from top to bottom, a complete data set for the SMOKE PTINV file can be entered. However, performing all that data entry by hand is very cumbersome if using the EIMA. The solution is to make a Comma Separated Value (CSV) file for each table and use a SQL utility to load the data into the database.

Bulk Data Loads

Bulk entry into the database can be done by using the SQL COPY command from the command line. To use this command the data must be in comma-delimited format with the column name at the top of each column. Through RTI's experience with importing data and from working with the client, we discovered that Excel would load, change, and export CSV files with the least effort. However, Excel occasionally leaves an odd character at the beginning of some strings. Therefore, the contents of each data file should be reviewed in a text editor (i.e., Notepad) before starting the import.

The database warns of any constraints not met while using the COPY command so understanding the requirements of each table before import is crucial. Many attempts may have to be made to import a single file as dependencies are checked and rechecked by the database's constraints.

For more information on the COPY command see the PostgreSQL user's guide here: http://www.postgresql.org/docs/techdocs.15. Syntax of COPY can be found here: http://www.postgresql.org/docs/8.0/interactive/sql-copy.html

CONCLUSIONS

Version 1 of the Emissions Inventory Maintenance Application (EIMA) has been successfully installed on the computers at BMEPB, where its user interface is Simplified Chinese language. It is also available in English, and can be translated to other languages without any reprogramming. The EIMA is licensed under the GNU Public License.

The data extraction utilities in the AQMDSS extract the emissions inventory data from the underlying database and create the input data files for SMOKE. Because all codes and related data are verified by EIMA during data entry, extremely few data problems can be expected when ingesting the data into SMOKE.

Finally, by using PostgreSQL as the database management system, the underlying emissions inventory database can be easily accessed by other programmers. Application programming interfaces are available for Java, C, C++, Perl, Python, and other languages. Also, the database can be directly accessed through SQL queries for ad hoc reporting. It also supports stored procedures, allowing additional, efficient data processing routines to be added to the AQMDSS in the future.

REFERENCES

[1] Brandmeyer, J.E., E. Solano, R.A. Zerbonia, G. Gao, X. Li, T. Chen, and A. Shi, 2005, The Emissions Inventory Database Application Component of the Air Quality Management Decision Support System for Beijing, 14th Annual Emission Inventory Conference, April 12–14, 2005, Las Vegas, NV

[2] RTI International, "Integration Report for the Air Quality Management Decision Support System," Prepared for PA Consulting Group by RTI International, Research Triangle Park, NC 2006.

[3] RTI International, "System Analysis and Design Report for the Air Quality Management Decision Support System – Revised Version," Prepared for PA Consulting Group by RTI International, Research Triangle Park, NC 2004.