Confirming Accuracy of the UIC Class II Area of Review

Using a Spatial Database to Increase Accuracy and Decrease Time in Reviewing Underground Injection Control (UIC) Applications for Class II Wells

Jessica Kane | December 14, 2017
University of Wisconsin - Madison

Introduction

The U.S. Environmental Protection Agency's Underground Injection Control (UIC) program ensures that certain criteria are met before a permit is issued to inject fluids underground for storage or disposal (U.S. Environmental Protection Agency, 2027). The UIC program is designed to protect underground sources of drinking water (USDW). Class II UIC wells are used to inject waste water from the oil and gas production process into the ground. As part of approving a permit application for Class II wells, regulators have to ensure that nearby wells would not act as conduits for waste water to move somewhere other than the intended injection zone. Operators are required to provide the following through the permit application process:

> *"Submit a topographic map, extending one mile beyond the property boundaries, showing the proposed injection well and the applicable ¼ mile radius area of review. The map should include all existing wells, that penetrate the injection zone, along with any surface bodies of water, springs, mines (surface and subsurface), quarries, and other pertinent surface features, including residences and roads, and faults, if known or suspected. Also submit a tabulation of information on these wells which will include the following: well name; surface location; date drilled; state permit #; operator name; total depth and operating status" (U.S. Environmental Protection Agency, 2017).*

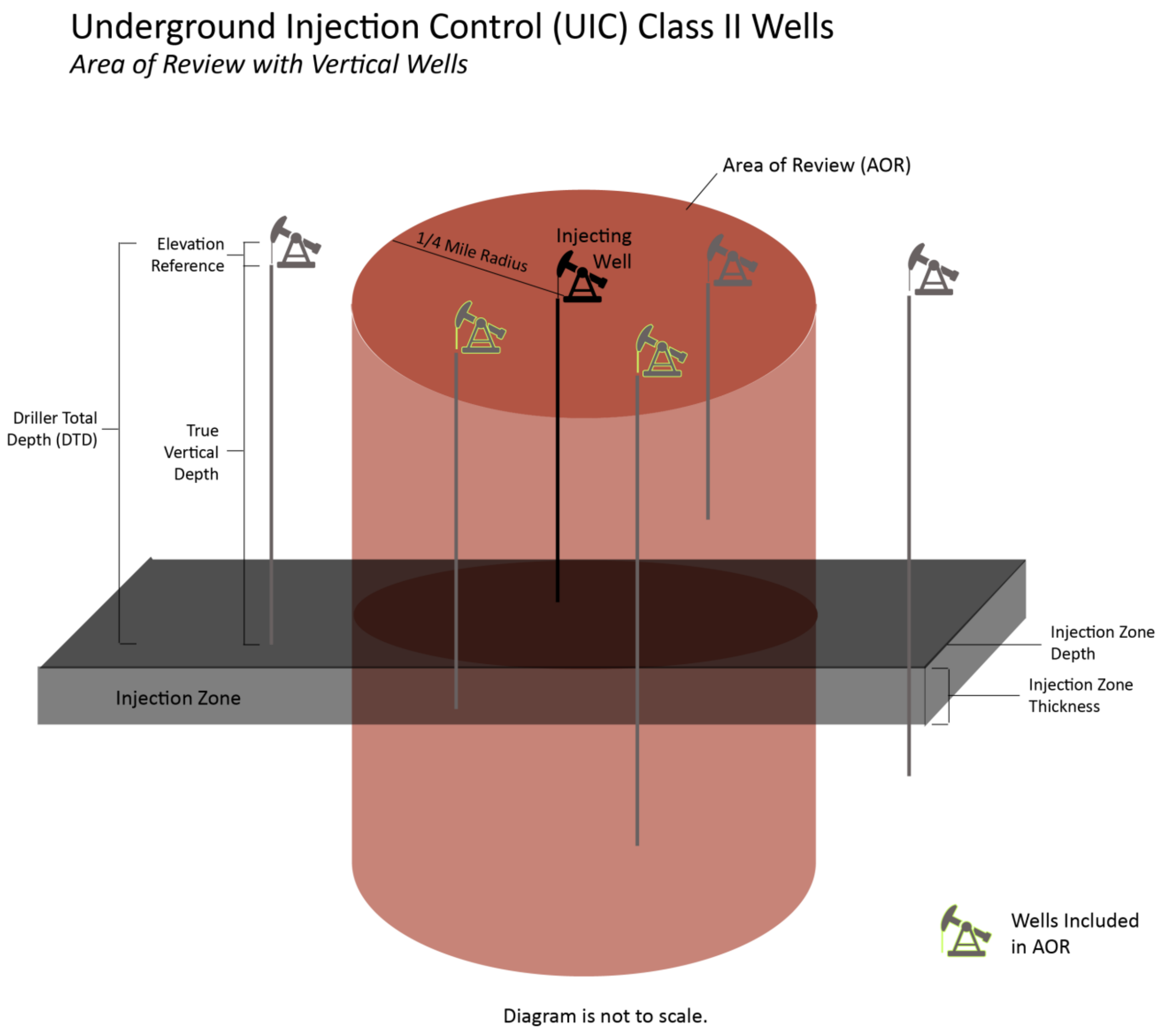


Figure 1 UIC Class II Area of Review

The EPA is primarily responsible for oversight of the UIC program; however, it has granted that responsibility to many states. In either case, the regulator reviewing the UIC permit application must confirm that the operator has submitted an accurate list of wells in the Area of Review (AoR) (Figure 1). This is important since this is the list that regulators and operators reference when ensuring that all wells in the AoR have the construction integrity necessary to withstand nearby injection activity. Identifying wells in the AoR is more complicated than just finding all wells within a ¼ mile of the proposed injection well since the depth of the wells must be taken into consideration as well.

A spatial database can be utilized to help ensure that agency reviews of UIC permit applications are thorough and decrease the chance of mistakes which might lead to groundwater contamination. It can also speed up the application review process so that operators do not have to wait as long to begin their injection projects.

This paper will cover the design and implementation of a spatial database with queries that allow the user to supply the location of the proposed injection well via latitude and longitude. The queries generate a list of the wells that penetrate the injection zone within a ¼ mile of that location, including the following information: well name, surface location, state permit number, operator name, total depth and operating status. The user can export this information to a .csv file or a shapefile for further analysis. The list and shapefile will help the regulatory agency answer the question: Did the operator submit a complete and accurate list of wells as required by the AoR process?

Methods

Dataset

The spatial database was designed to be used with data from the Michigan Department of Environmental Quality (MDEQ); however, with minor modifications it could be used with other states' data. A shapefile of all the oil and gas wells in the state of Michigan can be obtained through a secure FTP site that can be accessed through the MDEQ's GeoWebFace website (Michigan Department of Environmental Quality, 2017). A username and password must be requested through the MDEQ to access the FTP site.

Database Design

The spatial database is designed to hold the data from the MDEQ shapefile in one table while data generated through user input is held in other tables. The Conceptual Model (Figure 2) visualizes how the different entities are related to one another. Well and proposed injector entities are point data, as indicated by the black dots on those entities. An injection zone depth must be associated with a proposed injector as this number will be used in querying the database to see if a well should be included in the AoR. The well entities may have many attributes imported from the MDEQ shapefile, but the ones required by the UIC application process are shown in the Conceptual Model. In addition, the database will calculate the distance between a well and a proposed injector. AoR entities have a 1-to-1 relationship with proposed injectors and a many-to-many relationship with wells (i.e. One AoR can be related to multiple wells if multiple wells meet the AoR criteria. Also, one well can be related to multiple AoR's if it meets the AoR criteria for multiple nearby proposed injectors).

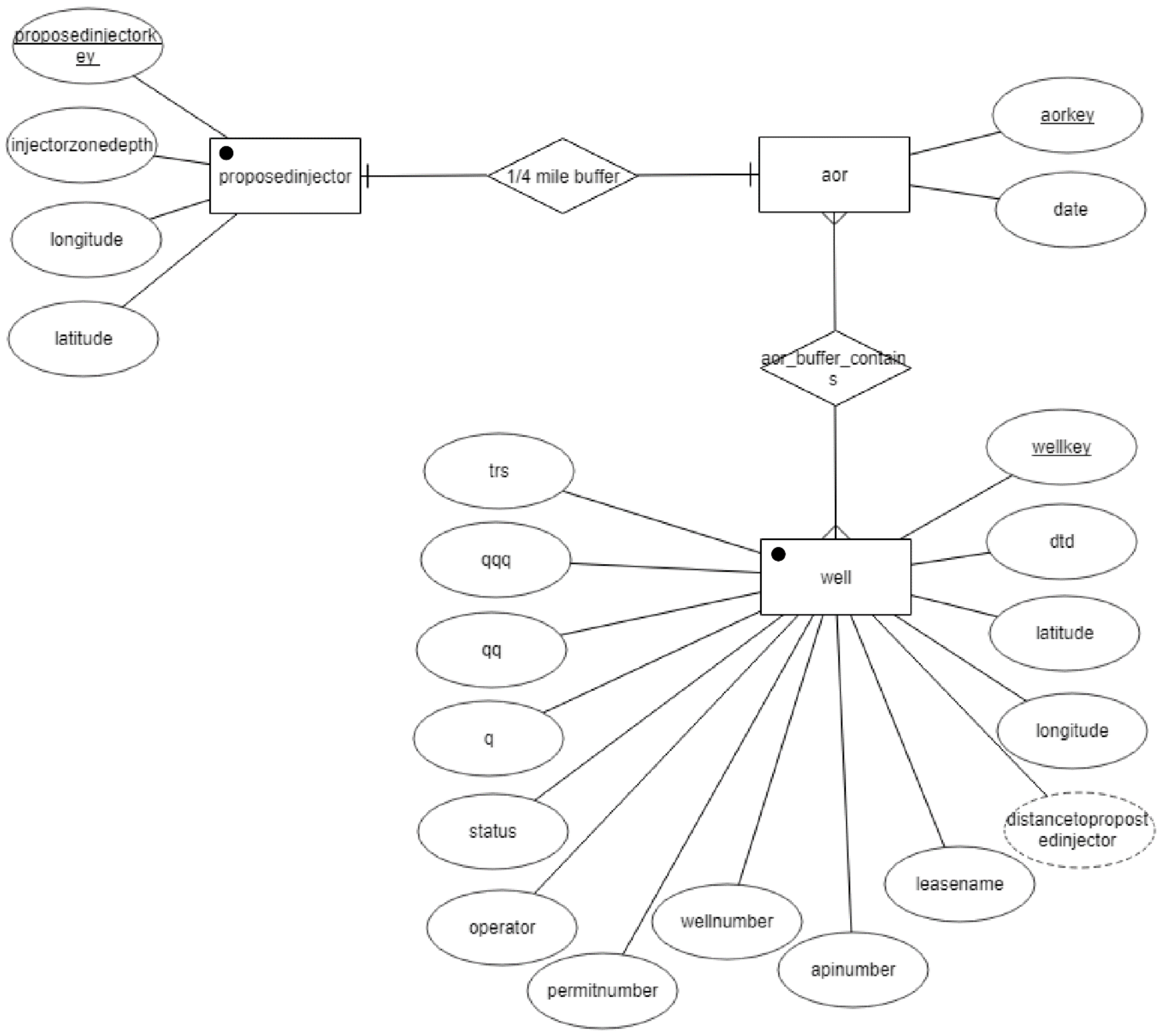


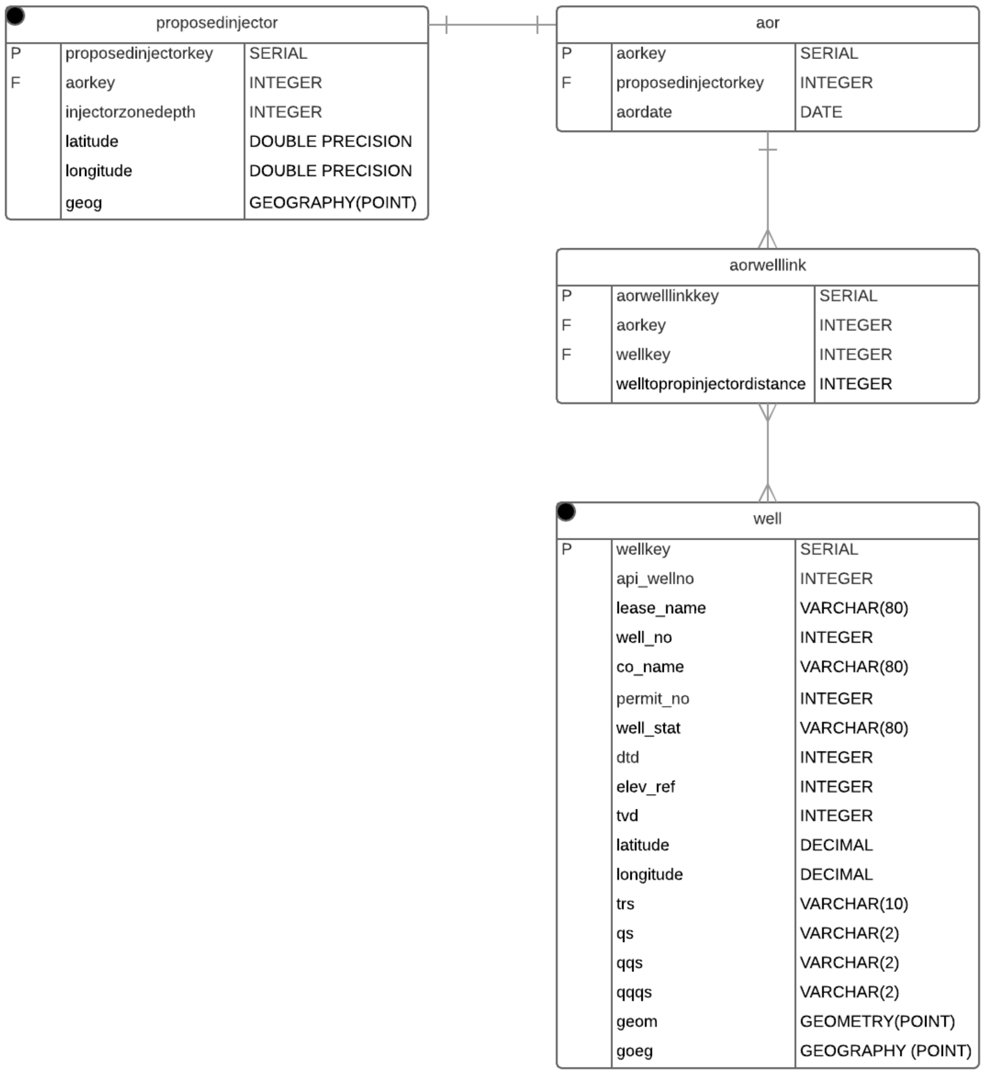
Figure 2 Conceptual Model

The Logical Model (Figure 3) takes the entities and relationships from the Conceptual Model and breaks them down into the tables that will be implemented in the database. A unique primary key is generated by the database for each record (automatically incrementing by one). Foreign keys are used to relate records from one table to records in another table. These are the primary key for the record stored in another table as part of a record in that table.

The **aorwelllink** table links the **aor** and **well** tables so that each record in the **aor** table is a unique AoR and each record in the **well** table is a unique well. The distance between a well included in the AoR and the proposed injector is stored in the **aorwelllink** table. This is because each **aor** record is only related to one

proposedinjector record, but a **well** record can be related to multiple **proposedinjector** records, so it wouldn't make sense to store this in the **well** table.

When the MDEQ shapefile is imported into the database, a column with the *geometry* is automatically created. In order for the system to be able to calculate distances in the chosen spatial reference (WGS 84), the points must have a *geography*. So the **well** table has both a *geom* and a *geog* column while the points generated by user input of latitude and longitude (stored in the **proposedinjector** table) just have a *geog*.



	proposedinjector	
P	proposedinjectorkey	SERIAL
F	aorkey	INTEGER
	injectorzonedepth	INTEGER
	latitude	DOUBLE PRECISION
	longitude	DOUBLE PRECISION
	geog	GEOGRAPHY(POINT)

	aor	
P	aorkey	SERIAL
F	proposedinjectorkey	INTEGER
	aordate	DATE

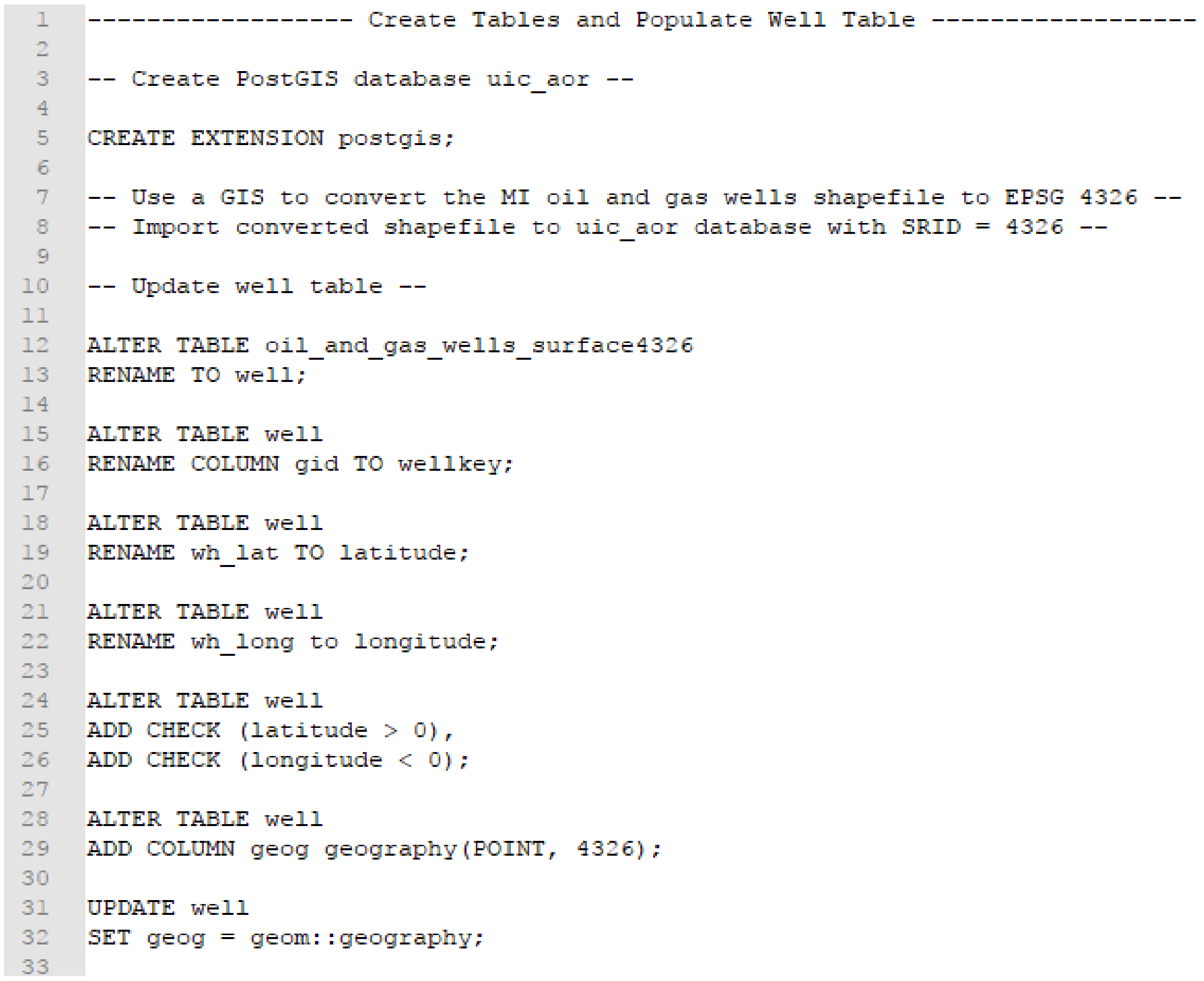
	aorwelllink	
P	aorwelllinkkey	SERIAL
F	aorkey	INTEGER
F	wellkey	INTEGER
	welltopropinjectordistance	INTEGER

	well	
P	wellkey	SERIAL
	api_wellno	INTEGER
	lease_name	VARCHAR(80)
	well_no	INTEGER
	co_name	VARCHAR(80)
	permit_no	INTEGER
	well_stat	VARCHAR(80)
	dtd	INTEGER
	elev_ref	INTEGER
	tvd	INTEGER
	latitude	DECIMAL
	longitude	DECIMAL
	trs	VARCHAR(10)
	qs	VARCHAR(2)
	qqs	VARCHAR(2)
	qqqs	VARCHAR(2)
	geom	GEOMETRY(POINT)
	goeg	GEOGRAPHY (POINT)

Figure 3 Logical Model

Database implementation

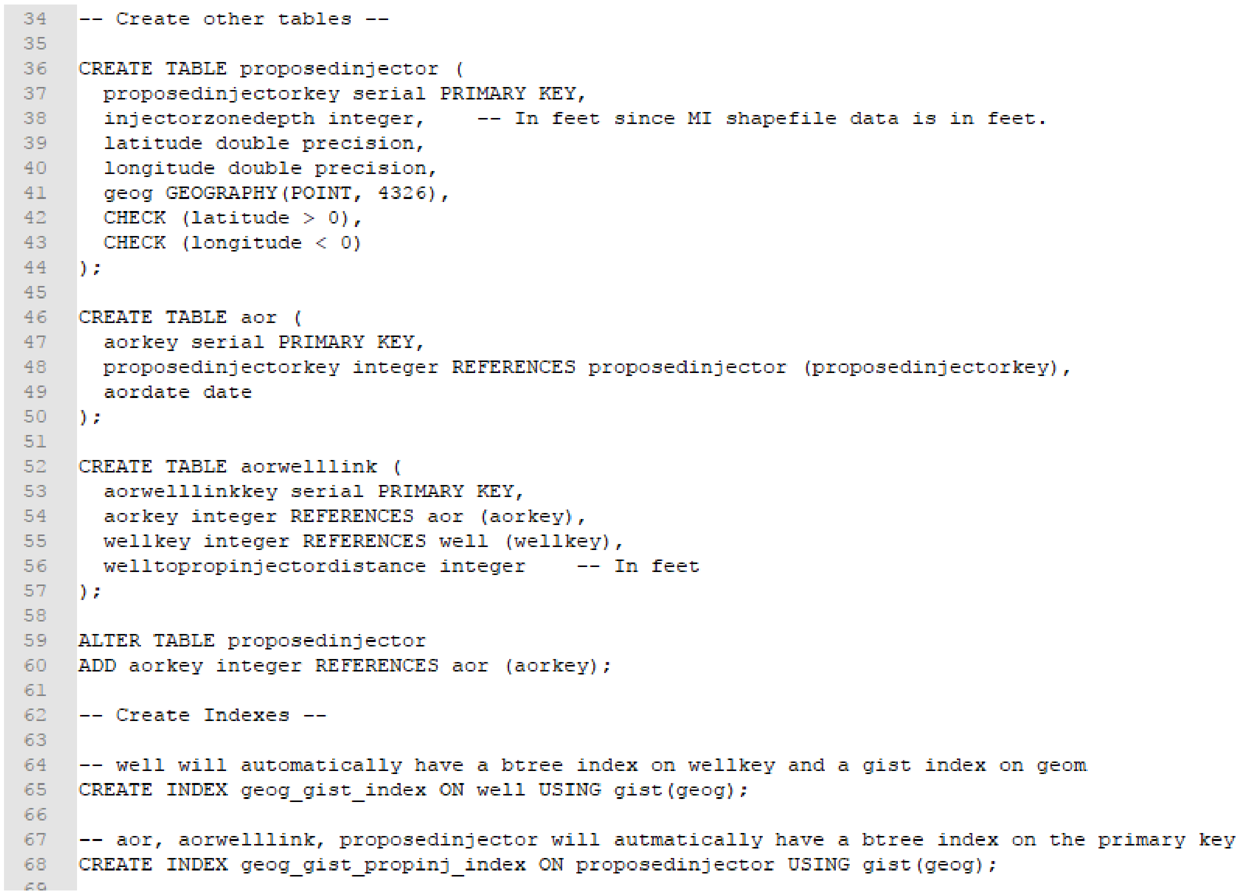
The database is implemented in PostgreSQL, utilizing the PostGIS spatial extension of that database. SQL is used to create the database, import the MDEQ shapefile, rename the table and some of the columns including the id column to *wellkey* as the primary key (Appendix A). Constraints are added to the *latitude* and *longitude* columns (Figure 4).



```
------------------ Create Tables and Populate Well Table ------------------

-- Create PostGIS database uic_aor --

CREATE EXTENSION postgis;

-- Use a GIS to convert the MI oil and gas wells shapefile to EPSG 4326 --
-- Import converted shapefile to uic_aor database with SRID = 4326 --

-- Update well table --

ALTER TABLE oil_and_gas_wells_surface4326
RENAME TO well;

ALTER TABLE well
RENAME COLUMN gid TO wellkey;

ALTER TABLE well
RENAME wh_lat TO latitude;

ALTER TABLE well
RENAME wh_long to longitude;

ALTER TABLE well
ADD CHECK (latitude > 0),
ADD CHECK (longitude < 0);

ALTER TABLE well
ADD COLUMN geog geography(POINT, 4326);

UPDATE well
SET geog = geom::geography;

```

Figure 4 Creating Database and well **Table (Appendix A)**

The other tables are created with their columns, including primary keys, foreign keys, and constraints on *latitude* and *longitude*. PostgreSQL automatically creates indexes on the primary key columns as well as the *geom* column in the **well** table. Since the database will be using the *geog* columns in the **well** and **proposedinjector** tables, the SQL script creates a generalized search tree (GiST) index on those two columns (Figure 5).

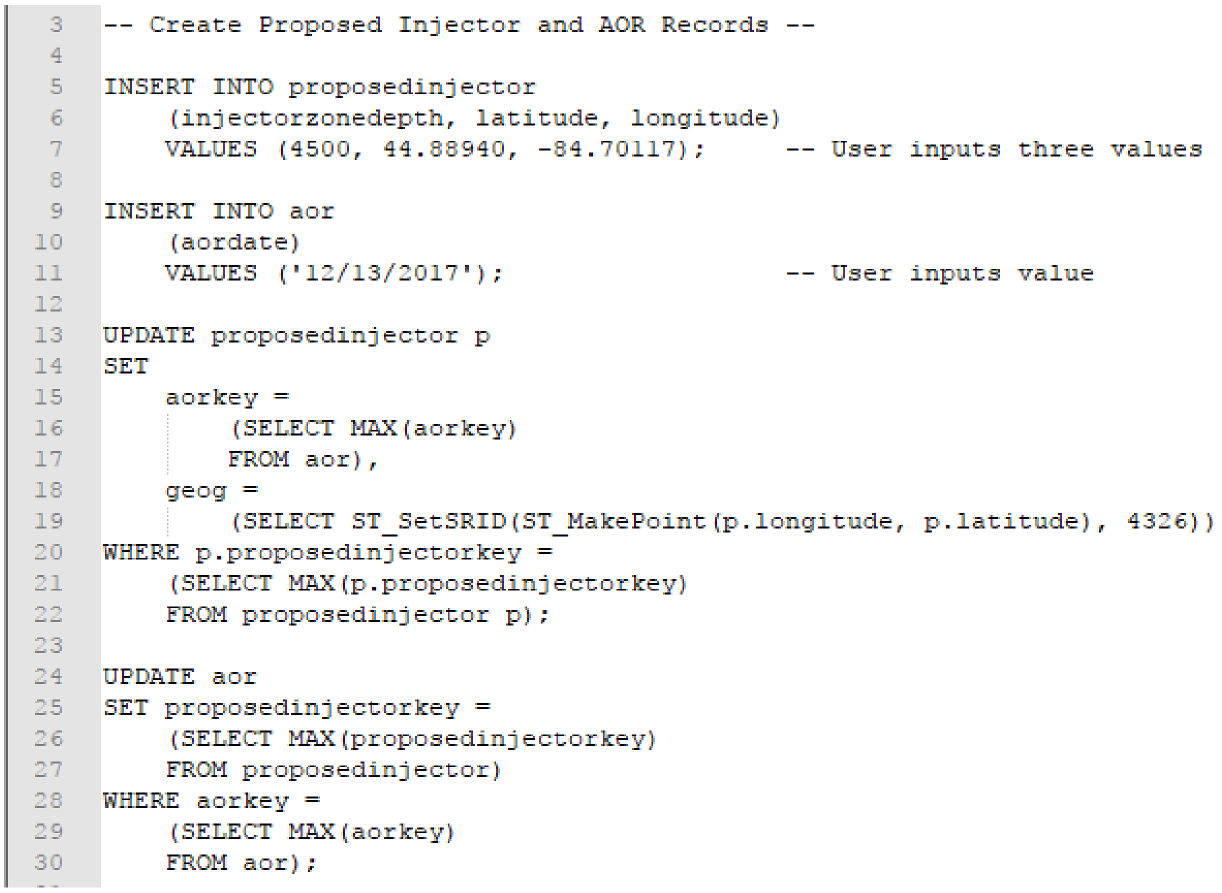


```sql
-- Create other tables --

CREATE TABLE proposedinjector (
  proposedinjectorkey serial PRIMARY KEY,
  injectorzonedepth integer,    -- In feet since MI shapefile data is in feet.
  latitude double precision,
  longitude double precision,
  geog GEOGRAPHY(POINT, 4326),
  CHECK (latitude > 0),
  CHECK (longitude < 0)
);

CREATE TABLE aor (
  aorkey serial PRIMARY KEY,
  proposedinjectorkey integer REFERENCES proposedinjector (proposedinjectorkey),
  aordate date
);

CREATE TABLE aorwelllink (
  aorwelllinkkey serial PRIMARY KEY,
  aorkey integer REFERENCES aor (aorkey),
  wellkey integer REFERENCES well (wellkey),
  welltopropinjectordistance integer    -- In feet
);

ALTER TABLE proposedinjector
ADD aorkey integer REFERENCES aor (aorkey);

-- Create Indexes --

-- well will automatically have a btree index on wellkey and a gist index on geom
CREATE INDEX geog_gist_index ON well USING gist(geog);

-- aor, aorwelllink, proposedinjector will autmatically have a btree index on the primary key
CREATE INDEX geog_gist_propinj_index ON proposedinjector USING gist(geog);
```

Figure 5 Creating Remaining Tables and Indexes (Appendix A)

Database manipulations

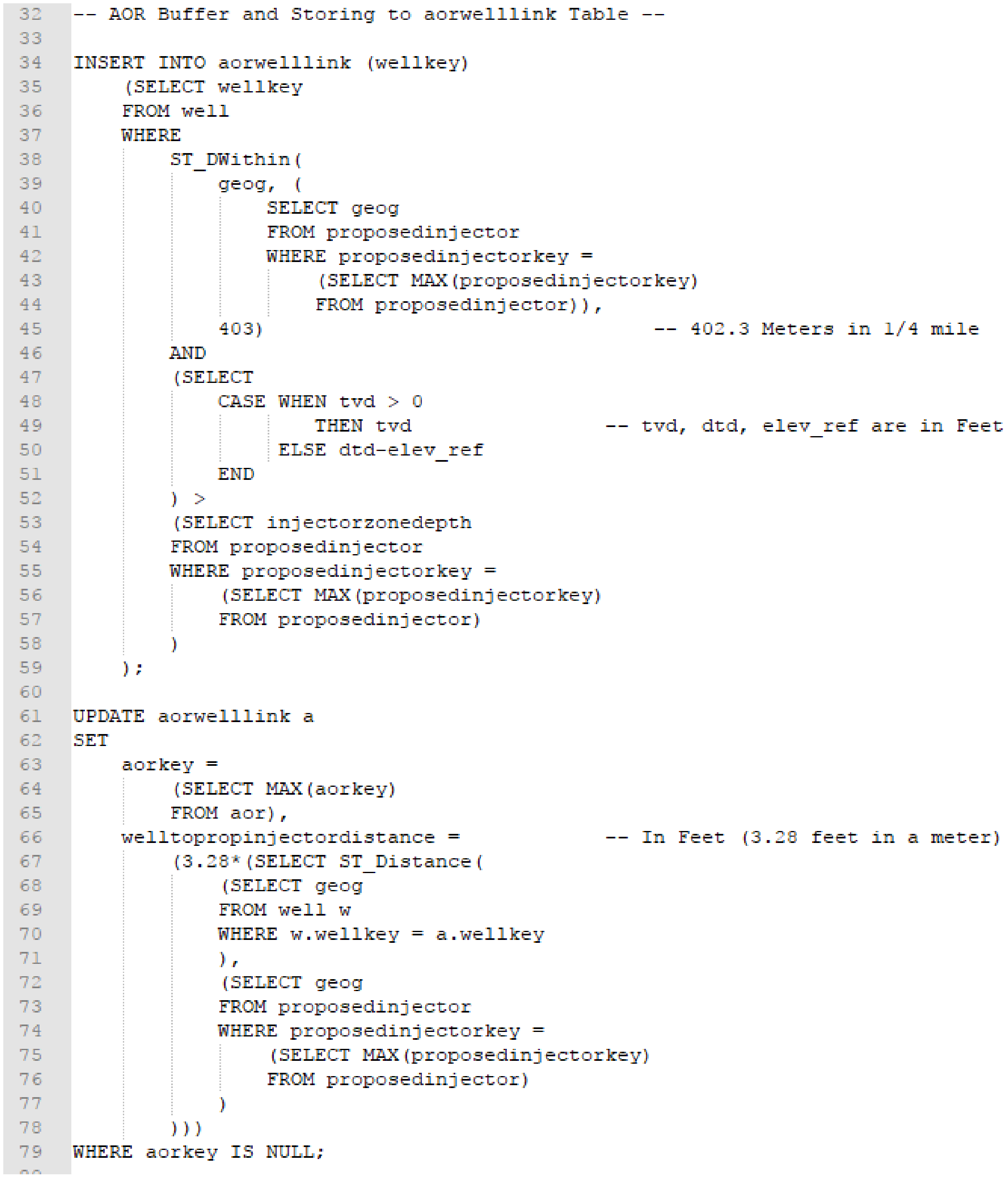
At this point, the database is set up and is ready for the user to input the necessary values from the UIC permit application to return the AoR wells. The SQL query (Appendix B) sets the necessary foreign keys to create the relationships between records in different tables. It also populates the *geog* column in the recently created record in the **proposedinjector** table based on the user-provided latitude and longitude (Figure 6).



```sql
-- Create Proposed Injector and AOR Records --

INSERT INTO proposedinjector
    (injectorzonedepth, latitude, longitude)
    VALUES (4500, 44.88940, -84.70117);      -- User inputs three values

INSERT INTO aor
    (aordate)
    VALUES ('12/13/2017');                   -- User inputs value

UPDATE proposedinjector p
SET
    aorkey =
        (SELECT MAX(aorkey)
        FROM aor),
    geog =
        (SELECT ST_SetSRID(ST_MakePoint(p.longitude, p.latitude), 4326))
WHERE p.proposedinjectorkey =
    (SELECT MAX(p.proposedinjectorkey)
    FROM proposedinjector p);

UPDATE aor
SET proposedinjectorkey =
    (SELECT MAX(proposedinjectorkey)
    FROM proposedinjector)
WHERE aorkey =
    (SELECT MAX(aorkey)
    FROM aor);
```

Figure 6 Creating Proposed Injector and AoR Records Based on User Input (Appendix B)

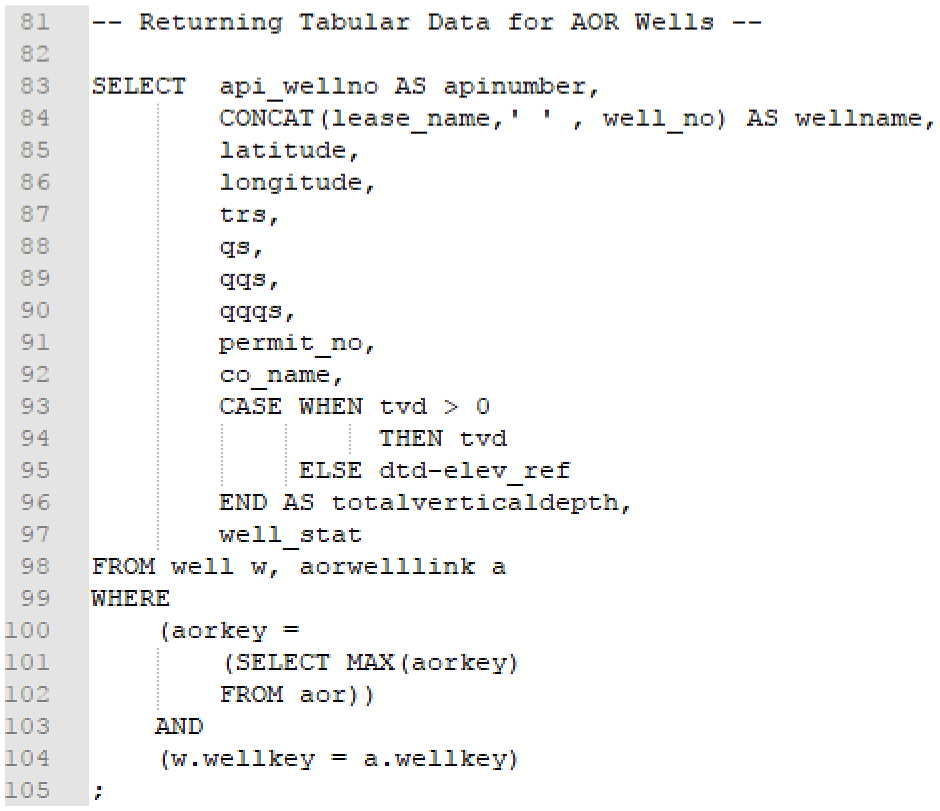
Through a spatial query, the SQL (Appendix B) then finds the wells that are within a ¼ mile of the proposed injection well and penetrate the injection zone. In the MDEQ data, *tvd* (total vertical depth) is the depth of the well. This is the value needed to determine if the well penetrates the injection zone; however, in many cases this column is not populated. Instead *dtd* (driller total depth) and *elev_ref* (elevation reference) are populated. Driller total depth is the measurement from the bottom of the hole to the top of a specified elevation reference, sometimes the well derrick floor. In this case, the total vertical depth can be calculated by taking the driller total depth minus the elevation reference number (Figure 1). When the total vertical depth is available, the SQL uses this value. Otherwise, it calculates this from the driller total depth and elevation reference numbers. Once these calculations are done, the resulting wells' primary keys are stored in **aorwelllink** as well as the AoR's primary key. The SQL also calculates the distance from the well to the proposed injector and stores the distance in feet (Figure 7).



```
-- AOR Buffer and Storing to aorwelllink Table --

INSERT INTO aorwelllink (wellkey)
    (SELECT wellkey
    FROM well
    WHERE
        ST_DWithin(
            geog, (
                SELECT geog
                FROM proposedinjector
                WHERE proposedinjectorkey =
                    (SELECT MAX(proposedinjectorkey)
                    FROM proposedinjector)),
            403)                                -- 402.3 Meters in 1/4 mile
        AND
        (SELECT
            CASE WHEN tvd > 0
                    THEN tvd                -- tvd, dtd, elev_ref are in Feet
                 ELSE dtd-elev_ref
            END
        ) >
        (SELECT injectorzonedepth
        FROM proposedinjector
        WHERE proposedinjectorkey =
            (SELECT MAX(proposedinjectorkey)
            FROM proposedinjector)
        )
    );

UPDATE aorwelllink a
SET
    aorkey =
        (SELECT MAX(aorkey)
        FROM aor),
    welltopropinjectordistance =            -- In Feet (3.28 feet in a meter)
        (3.28*(SELECT ST_Distance(
            (SELECT geog
            FROM well w
            WHERE w.wellkey = a.wellkey
            ),
            (SELECT geog
            FROM proposedinjector
            WHERE proposedinjectorkey =
                (SELECT MAX(proposedinjectorkey)
                FROM proposedinjector)
            )
        )))
WHERE aorkey IS NULL;
```

Figure 7 Selecting AoR Wells and Storing Data to aorwelllink **Table (Appendix B)**

After running the SQL in Figures 6 and 7, all of the data has been generated and stored from the four values that the user specified. The remainder of the SQL in Appendix B is a query that returns the AoR wells with the data required by the UIC permit application (Figure 8).



```sql
-- Returning Tabular Data for AOR Wells --

SELECT  api_wellno AS apinumber,
        CONCAT(lease_name,' ' , well_no) AS wellname,
        latitude,
        longitude,
        trs,
        qs,
        qqs,
        qqqs,
        permit_no,
        co_name,
        CASE WHEN tvd > 0
                  THEN tvd
             ELSE dtd-elev_ref
        END AS totalverticaldepth,
        well_stat
FROM well w, aorwelllink a
WHERE
    (aorkey =
        (SELECT MAX(aorkey)
        FROM aor))
    AND
    (w.wellkey = a.wellkey)
;
```

Figure 8 Returning Data for AoR Wells (Appendix B)

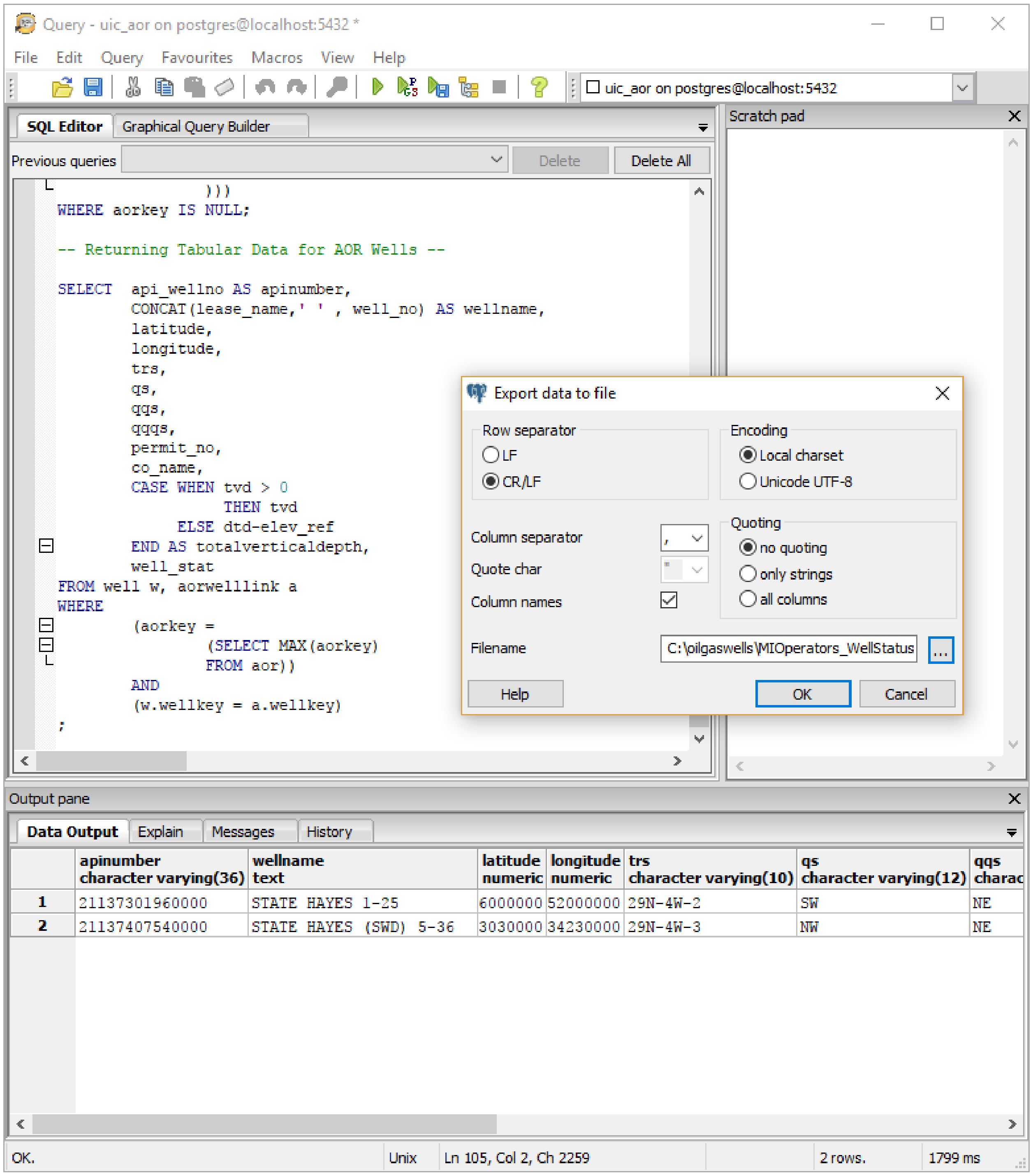
Now all that remains is exporting the AoR wells in a format that the user can use, either as a .csv file or a shapefile.

Results

Exporting Data

As an example, let's assume that the user has received a UIC permit application for an injection well with the coordinates (44.87398, -84.74016) and an injection zone depth of 1,100 feet. After running the SQL in Appendix B, two wells are returned in the AoR. The user then has two options for exporting the data for further analysis.

The Graphical User Interface (GUI) for PostgreSQL called pgAdmin can be used to export the data to a .csv file (Figure 9) and compared to the list of wells that the operator submitted with the UIC permit application.


Query - uic_aor on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
uic_aor on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History

```
                    )))
WHERE aorkey IS NULL;

-- Returning Tabular Data for AOR Wells --

SELECT  api_wellno AS apinumber,
        CONCAT(lease_name,' ' , well_no) AS wellname,
        latitude,
        longitude,
        trs,
        qs,
        qqs,
        qqqs,
        permit_no,
        co_name,
        CASE WHEN tvd > 0
                  THEN tvd
             ELSE dtd-elev_ref
        END AS totalverticaldepth,
        well_stat
FROM well w, aorwelllink a
WHERE
        (aorkey =
                (SELECT MAX(aorkey)
                 FROM aor))
        AND
        (w.wellkey = a.wellkey)
;
```

Export data to file

Row separator: ( ) LF [✓ CR/LF]

Encoding: [✓ Local charset] ( ) Unicode UTF-8

Column separator: ,

Quote char: "

Column names: ✓

Quoting: [✓ no quoting] ( ) only strings ( ) all columns

Filename: C:\oilgaswells\MIOperators_WellStatus

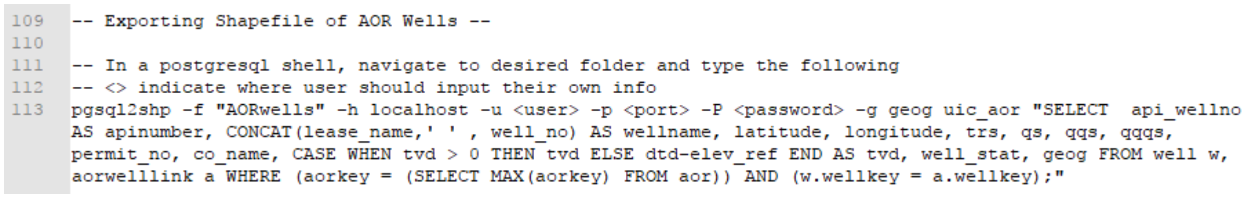
Help | OK | Cancel

	apinumber character varying(36)	wellname text	latitude numeric	longitude numeric	trs character varying(10)	qs character varying(12)	qqs charac
1	21137301960000	STATE HAYES 1-25	6000000	52000000	29N-4W-2	SW	NE
2	21137407540000	STATE HAYES (SWD) 5-36	3030000	34230000	29N-4W-3	NW	NE

OK. Unix Ln 105, Col 2, Ch 2259 2 rows. 1799 ms

Figure 9 Exporting the AoR Wells to a .csv File

The data can also be exported to a shapefile, then viewed in a GIS. This is done using the pgsql2shp utility in a PostgreSQL shell (Figure 10).



```
-- Exporting Shapefile of AOR Wells --

-- In a postgresql shell, navigate to desired folder and type the following
-- <> indicate where user should input their own info
pgsql2shp -f "AORwells" -h localhost -u <user> -p <port> -P <password> -g geog uic_aor "SELECT  api_wellno
AS apinumber, CONCAT(lease_name,' ' , well_no) AS wellname, latitude, longitude, trs, qs, qqs, qqqs,
permit_no, co_name, CASE WHEN tvd > 0 THEN tvd ELSE dtd-elev_ref END AS tvd, well_stat, geog FROM well w,
aorwelllink a WHERE (aorkey = (SELECT MAX(aorkey) FROM aor)) AND (w.wellkey = a.wellkey);"
```

Figure 10 Exporting a Shapefile of AoR Wells (See Appendix B)

In the example, four wells are within a ¼ mile of the proposed injection well (in yellow in Figure 11). The generated shapefile only includes the two wells within that ¼ mile whose total vertical depth is more than 1,100 feet (in red).

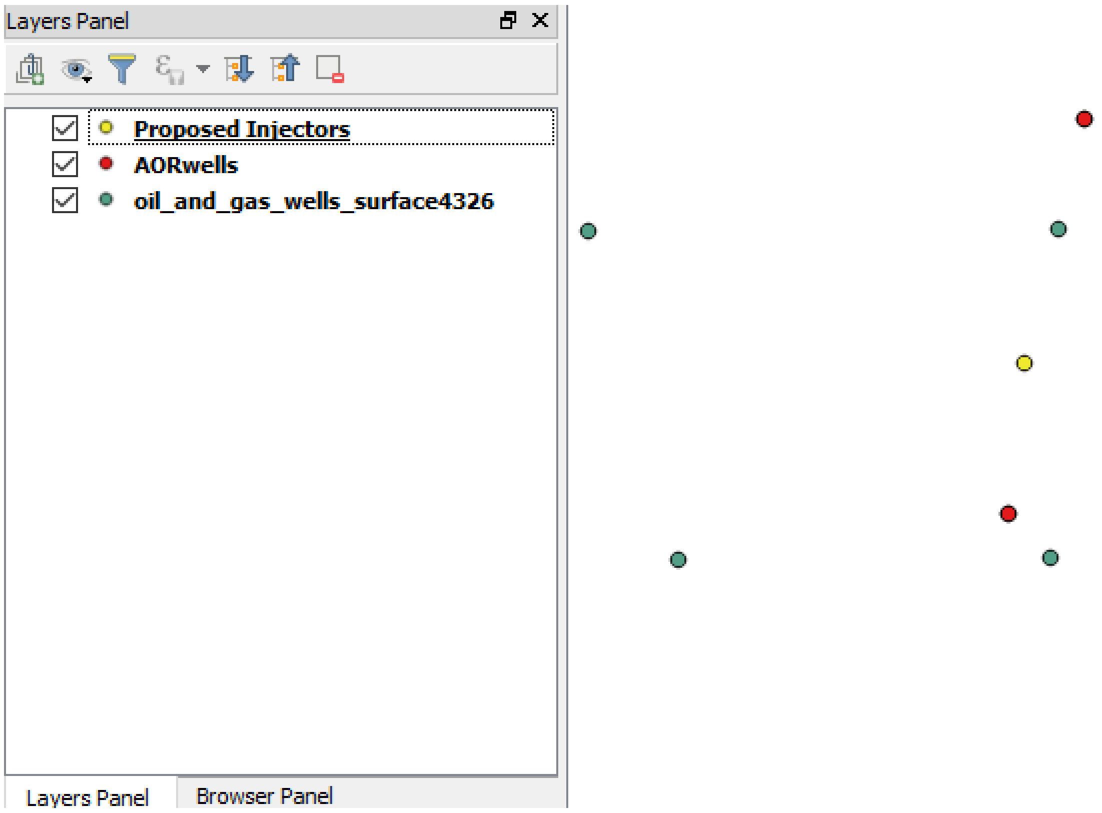


Figure 11 Displaying Generated Shapefile in QGIS

The user can then bring in other data layers such as satellite imagery and bodies of water to examine other features in the Area of Review.

Time Savings

Without the use of a spatial database, the user would have to double-check the operator's AoR list manually. This would involve finding the proposed injector coordinates in a GIS, measuring a ¼ mile and examining each well in that radius for its total vertical depth. Sometimes the user would need to calculate the total vertical depth before comparing it to the injection zone depth. This tedious work could take a long time and has many opportunities for error.

With a spatial database already implemented, the user can complete the same task with the resulting .csv and shapefile in less than a minute.

Discussion

This paper has demonstrated how to implement a simple spatial database that takes a few user inputs and results in a list of AoR wells, both in .csv and shapefile format. This time-saving tool has the following capabilities:

- Stores all oil & gas well locations and attributes for a particular state (including latitude and longitude)
- Finds all the wells within a 1/4 mile of a location
- Accepts the value for the top of the injection zone (user inputted value) and compares that to the total vertical depth of the wells in the buffer. Then returns just those that are deeper than the top of the injection zone.
- Returns required attributes of those wells, allowing for export to a .csv file
- Creates a shapefile with the final well list (including non-spatial attributes) for further analysis in a GIS

While this paper has demonstrated the benefits of utilizing a spatial database to assist regulators in reviewing UIC permit applications, the current implementation is a proof of concept only. There are a number of areas where further analysis and development would add value.

User Interface

With detailed step-by-step instructions, non-technical users would be able to use this tool as is; however, for widespread adoption, a user interface should be developed to facilitate ease of use.

Expanded Data Model

Expanding the data model to include additional tables such as an **operator** table as well as additional data in the **aor** table would allow users to return at a later data and more easily access previous AoR's.

Directional and Horizontal Wells

Advances in technology have made drilling of directional and horizontal wells (i.e. non-vertical wells) possible. The current database implementation presented in this paper would only guarantee capture of vertical wells in the Area of Review. Horizontal and directional wells that should be included in the AoR might be excluded if their surface location was not within ¼ mile of the proposed injection well (Figure 12). As long as states are collecting directional data, the database could be expanded to store this information and the spatial queries adjusted to ensure that horizontal and directional wells are not missed.

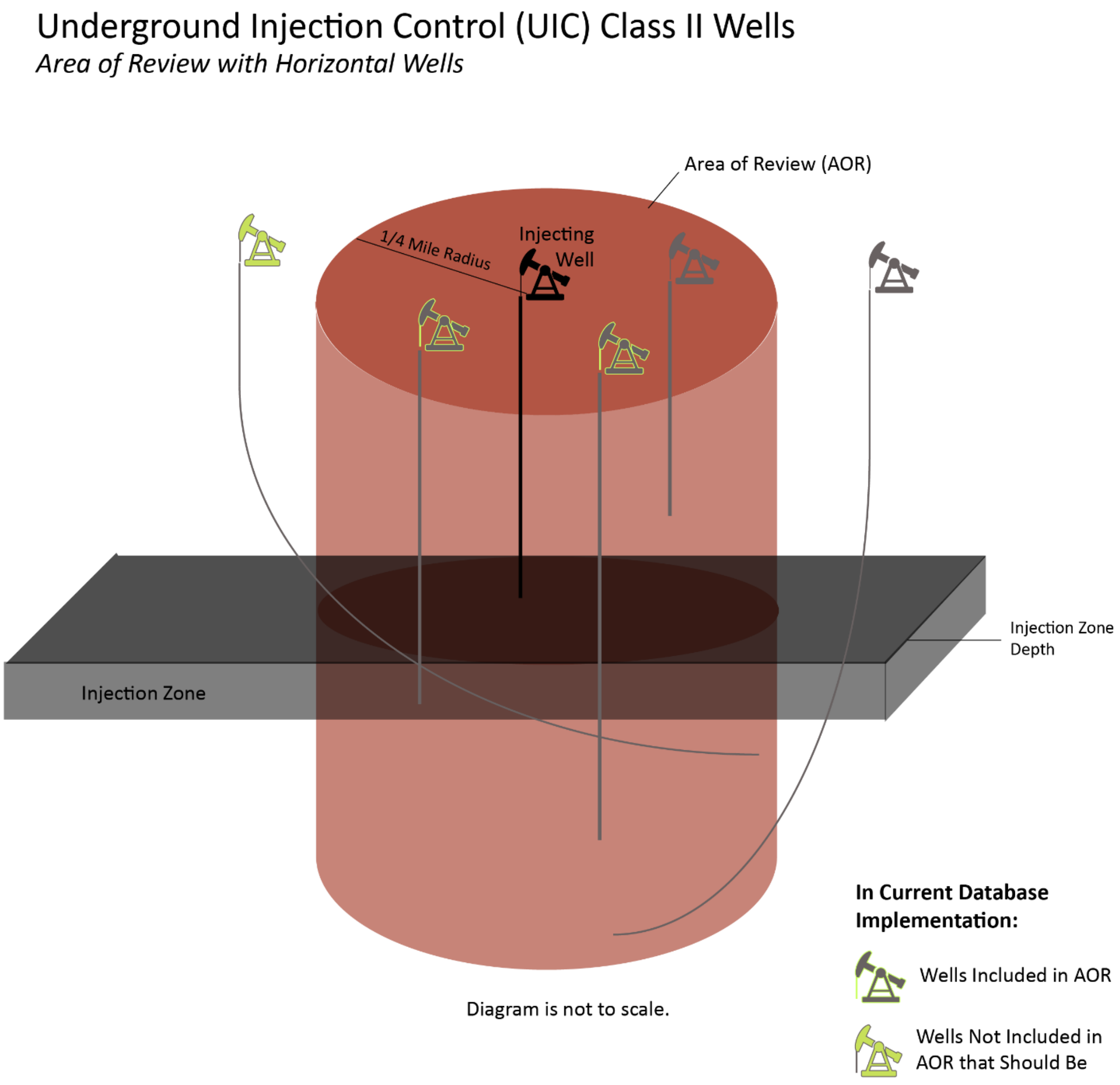

Underground Injection Control (UIC) Class II Wells
Area of Review with Horizontal Wells
Area of Review (AOR)
1/4 Mile Radius
Injecting Well
Injection Zone Depth
Injection Zone
Diagram is not to scale.
In Current Database Implementation:
Wells Included in AOR
Wells Not Included in AOR that Should Be

Figure 12 UIC Class II Area of Review with Horizontal Wells

Equation vs. ¼ Mile

Sometimes an operator is permitted to use an equation instead of a ¼ mile radius to determine the wells in the AoR (U.S. Environmental Protection Agency, 2017). Further analysis of such equations could be conducted to determine how this tool might be adapted to be used in those situations as well.

Acknowledgements

This paper was produced as part of the Masters of Science in Cartography & Geographic Information Systems – GIS Development at the University of Wisconsin – Madison during the fall of 2017.

The author of this paper held a Career Development Grant from AAUW.

References

Michigan Department of Environmental Quality. (2017, September 24). *GeoWebFace*. Retrieved from Department of Environmental Quality: http://www.deq.state.mi.us/GeoWebFace/

U.S. Environmental Protection Agency. (2017, November 6). *EPA Region 5 Example Underground Injection Control Permit Application Documents.* Retrieved from United States Environmental Protection Agency: https://www.epa.gov/uic/epa-region-5-example-underground-injection-control-permit-application-documents

U.S. Environmental Protection Agency. (2027, December 12). *Protecting Underground Sources of Drinking Water from Underground Injection (UIC)*. Retrieved from United States Environmental Protection Agency: https://www.epa.gov/uic

Appendices

Appendix A: SQL_CreateTables.txt

Appendix B: SQL_InputProposedInjector.txt