AcuGIS Scout

Release 1.0.0

acugis

Jun 09, 2022

GETTING STARTED

CHAPTER

ONE

INTRO

1.1 Overview

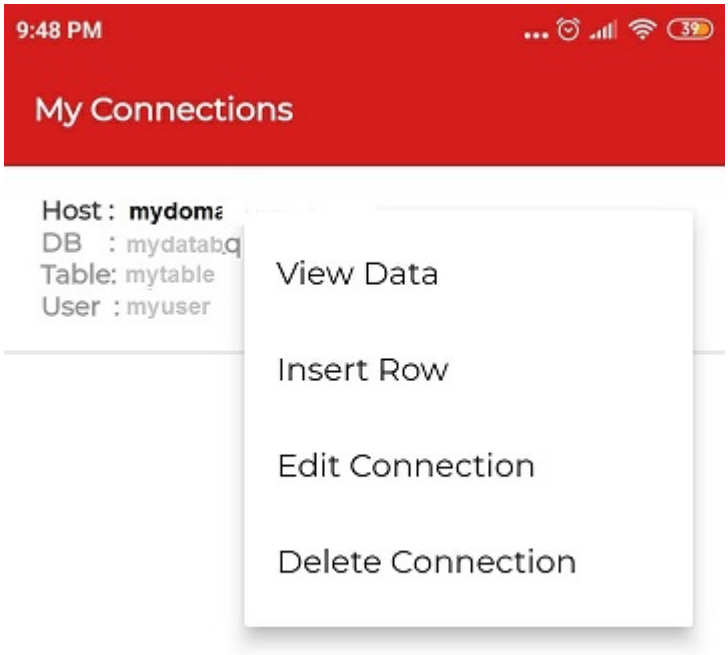
AcuGIS Scout is a simple Mobile Form Builder for PostgreSQL.

It creates a Mobile Form from a table you create in PostgreSQL.

In addition to rendering form fields, it also captures lat/lon and altitude.

Images can also be inserted as well as QR scanning.

Scout is designed for ease of use, allowing you deploy and update mobile forms quickly and easily.



1.2 Operating Systems

- Android
- iOS

1.3 Authors

- AcuGIS

CHAPTER

TWO

QUICK START

Below is the Quick Start

2.1 1. Create a table in PostgreSQL using below:

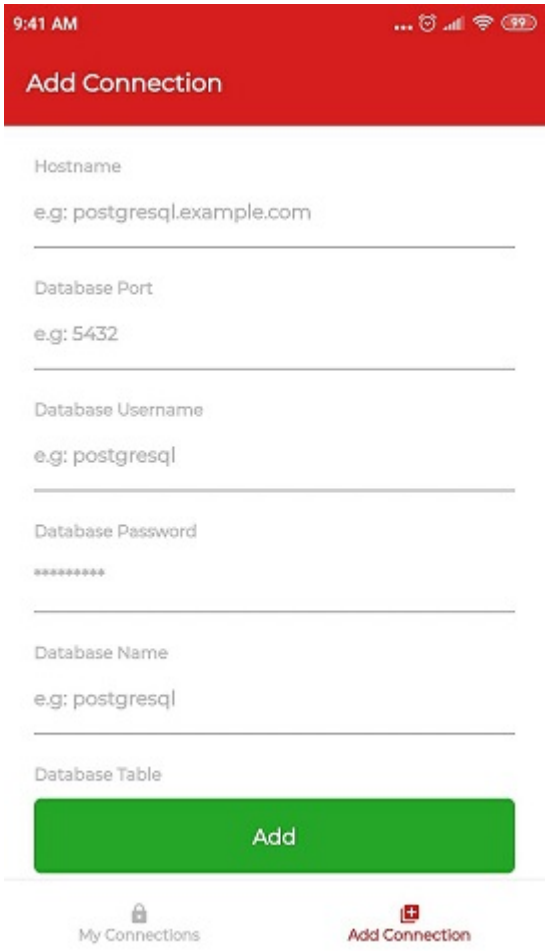
```
CREATE SEQUENCE public.my_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;

CREATE TABLE mytable (
id integer DEFAULT nextval('public.my_id_seq'::regclass) NOT NULL,
title character varying(200),
qr character varying(200),
location point,
latlngalt json,
"timestamp" timestamp without time zone
);
```

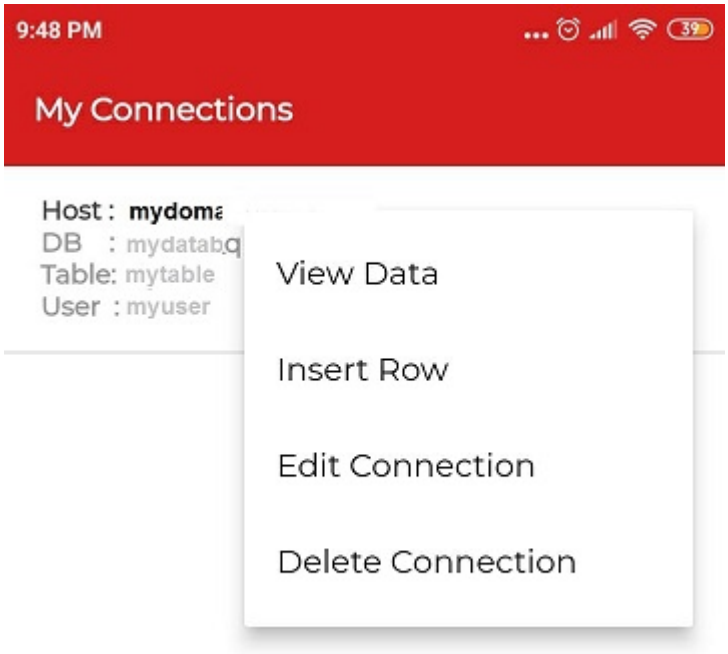
2.2 2. Install Scout on your iOS or Android Device

App can be downloaded via Google Playstore or Apple App Store

2.3 3. Add your PostgreSQL connection details.



2.4 4. Start adding data



Note: Be sure to read the rest of the Docs for important information about best practice, security, and other features.

CHAPTER

THREE

CREATE TABLE

The minimum requirements are four columns ‘id’, ‘location’, ‘latlngalt’, and ‘timestamp’

Create the table in PostgreSQL:

```
CREATE TABLE mytable (
id integer DEFAULT nextval('public.my_id_seq'::regclass) NOT NULL,
location point,
latlngalt json,
"timestamp" timestamp without time zone
);
```

AcuGIS Scout is programmed to ignore Primary auto-incremented key when creating forms.

So create an appropriate sequence

```
CREATE SEQUENCE public.my_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

In the next section, we’ll look at optional field data types.

3.1 Mandatory Columns

3.1.1 id

Primary key with autoincrement. Does not appear on rendered form

3.1.2 location

Stores location as lat/lon point

3.1.3 latlngalt

Stores altitude, along with lat/lon as json

3.1.4 timestamp

Stores date and time

CHAPTER

FOUR

IMAGES

Image columns must be of the Data Type bytea

The column name can be anything you wish. Below, we are using 'myimage'

You can add as many image columns as wish as well

Create the table in PostgreSQL with image field:

```
CREATE TABLE mytable (
id integer DEFAULT nextval('public.my_id_seq'::regclass) NOT NULL,
myimage bytea,
location point,
latlngalt json,
"timestamp" timestamp without time zone
);
```

Images are stored in the column as bytea binary files.

CHAPTER

FIVE

QR SCANNER

All varchar columns will have the option to use QR Scan

The column name can be anything you wish. Below, we are using ‘myqr’

You can add as many varchar columns as wish as well

Below, we add Column ‘myqr’ to serve as our QR scan field:

```
CREATE TABLE mytable (
id integer DEFAULT nextval('public.my_id_seq'::regclass) NOT NULL,
myqr character varying(200),
location point,
latlngalt json,
"timestamp" timestamp without time zone
);
```

QR values are stored as text in the database

CHAPTER

SIX

ADDING COLUMNS

You can add additional columns to your table at any time.

The new column(s) will be added to the form automatically.

CHAPTER

SEVEN

SECURITY

Create your table in a dedicated schema and grant access to a minimally privlaged user.

Below, we'll create a new user, 'minuser' with only the required permissions for the table 'mytable'.

```
CREATE ROLE minuser with PASSWORD 'password';
```

Grant Connect on the database

```
GRANT CONNECT ON DATABASE mydb TO minuser;
```

Grant SELECT, INSERT, and UPDATE on table 'mytable'

```
GRANT SELECT, INSERT, UPDATE ON mytable TO minuser;
```

CHAPTER

EIGHT

CREATE CONNECTION

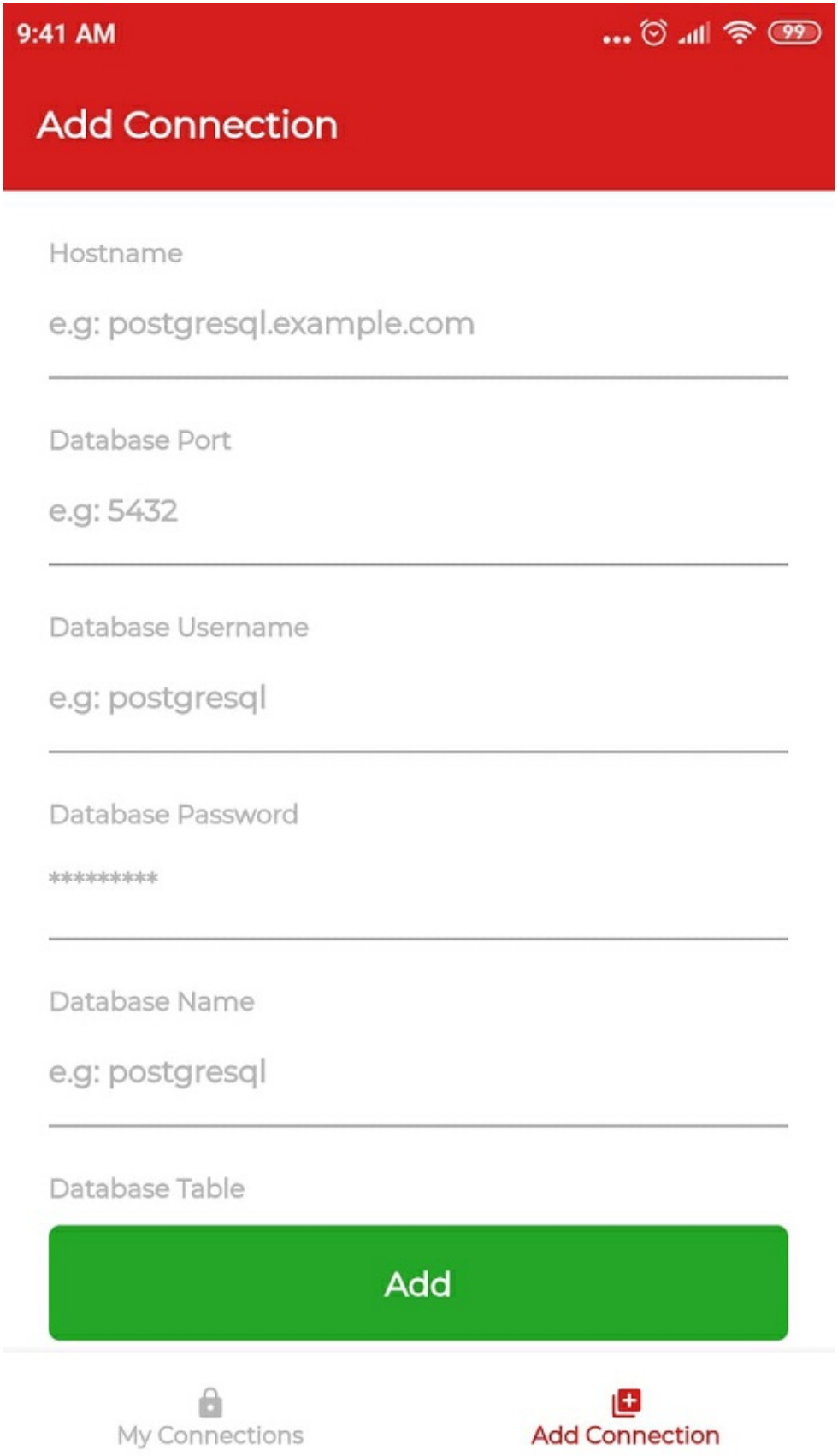
8.1 Create New Connection

New Connections (Datasources), can be added via the home page.

To create a new Connection (Datasource), click ‘Add Connection’ at the bottom of the screen as show below

8.2 Enter Connection Info

Enter the required information as show below


9:41 AM
99
Add Connection
Hostname
e.g: postgresql.example.com
Database Port
e.g: 5432
Database Username
e.g: postgresql
Database Password
****
Database Name
e.g: postgresql
Database Table
Add
My Connections
Add Connection

Important: The table location defaults to the Public schema. If you are using another schema, enter the schema name and table name in the database table field, seperated by a dot. For example, for Schema 'myschema' and Table 'mytable', enter as 'myschema.mytable' For Public schema, you do not need to specifiy the schema.

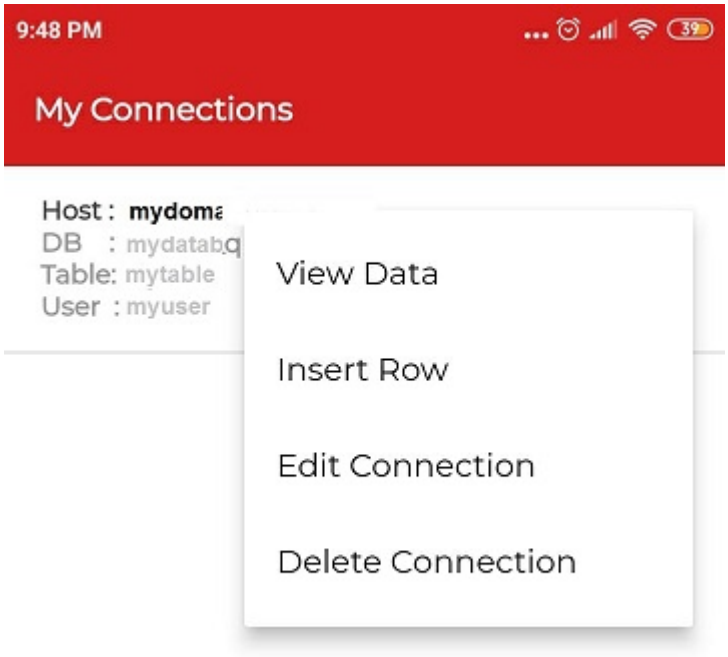
Once all of the fields are popluated, click the "Add" button.

CHAPTER

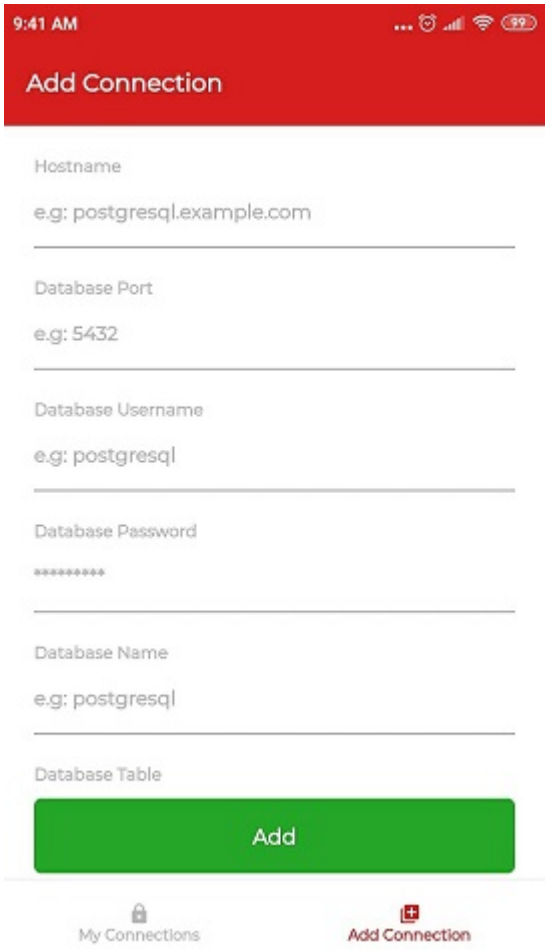
NINE

INSERT ROW

To insert a new row, click the 3 dots for the drop down menu and select “Insert New Row” as shown below



Your form should now open as shown below



Once all of the fields are popluated, click the “Insert” button.

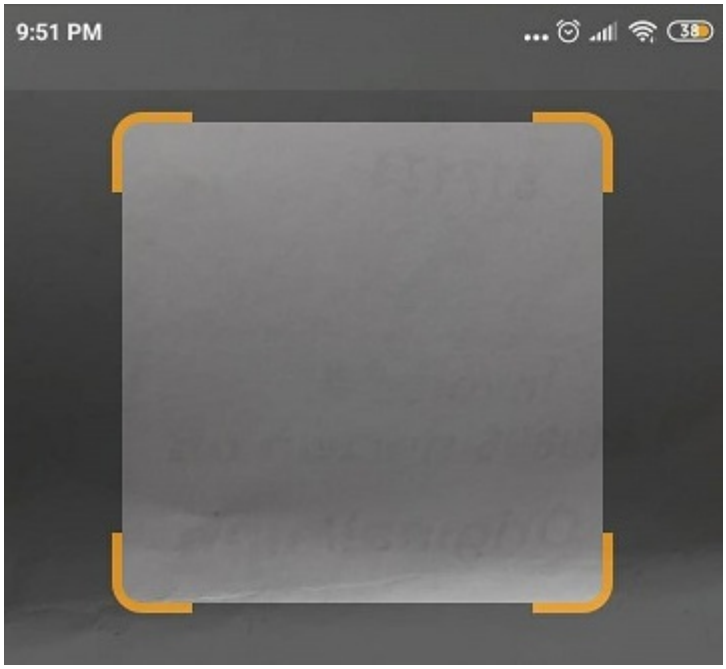
9.1 Feild Types

Scout produces 2 field types:

1. Text and QR

All text fields (charactervarying) include option to scan QR code. At the right side of the text field, the image below appears:

Clicking the icon will open the QR Scanner function as shown below

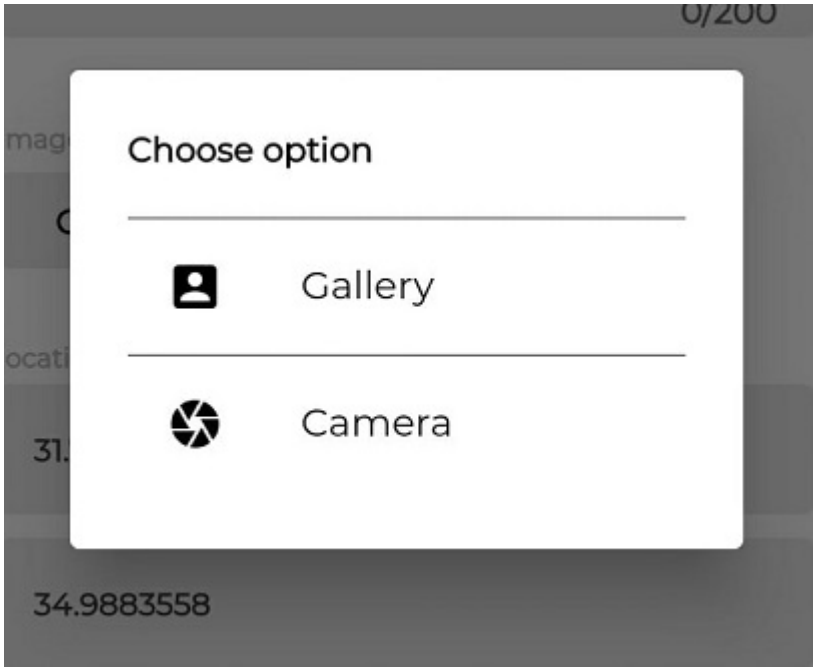


Simply place the QR code within the box and the code will be scanned.

2. image (bytea)

For image fields, clicking the green “Image” button will open the camera function.

From here you can take a photo or select an existing photo from your gallery, as shown below

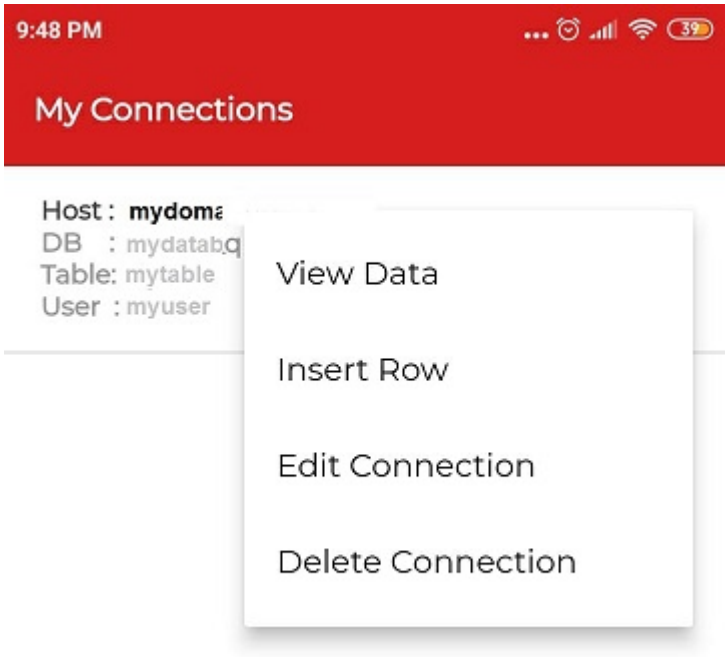


CHAPTER

TEN

EDIT CONNECTION

To edit a data source, click the 3 dots for the drop down menu and select “Edit Connection” as shown below



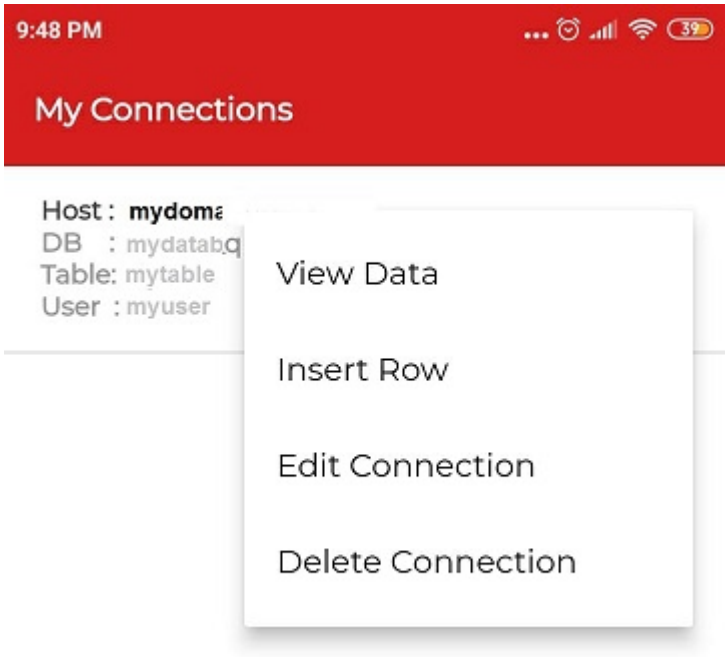
Make your changes and click “Save”

CHAPTER

ELEVEN

VIEW DATA

To view data for a given Connection, click the 3 dots for the drop down menu and select “View Data” as shown below



Make your changes and click “Save”

CHAPTER

TWELVE

JSON SQL EXAMPLES

While lat/lon is stored in the location column as a Point, altitude is stored in a json array (along with lat/lon) in the latlngalt column.

A standard select query such as 'select latlngalt from mytable', latwould produce as below:

```
"{""latitude"":""41.7408504"",""longitude"":""87.9883978"",""altitude"":""327.
↪3999938964844""}"
```

To query altitude value alone, use the json operators >> for text:

```
select latlngalt->>'altitude' from mytable;
```

Which produces the values for altitude as text.

To select the values as json, use the 'short' arrow

```
select latlngalt->>'altitude' from mytable;
```

CHAPTER

THIRTEEN

IMAGES

Images can be queried directly from the database, of course, for rendering into your applications.

If you wish to pull images directly from the database to the file system, you can use similar to below

```
psql -d mydb -Aqt -c "SELECT encode(myimagecol, 'base64') FROM mytable where id=88" |␣
↪base64 -d > mypicture.png
```

CHAPTER

FOURTEEN

CONVERT TO GEOMETRY

lat/lon is stored in the location column as a Point.

This is done to make the app useful to PostgreSQL users who do not have (or want) PostGIS enabled.

A select of the location column will produce an output like below:

```
(28.657320220545436,-81.2845245094787)
(31.7408504,34.9883978)
```

To convert the location column to Geometry type Point, you can use below

```
select ST_SetSRID((location)::GEOMETRY(POINT), 4326) from mytable;
```

This will provide an output similar to below:

```
0101000020E6100000FE3D522346A83C40C6C749A6355254C0
0101000020E61000000803A2F5FA8BD3F401C38B1D1837E4140
```

To convert to this output WKT, we can use st_asewkt as below:

```
SELECT st_asewkt(ST_SetSRID((location)::GEOMETRY(POINT), 4326)) from mytable;
```

This will provide the output in WKT format as below:

```
SRID=4326;POINT(28.657320220545436 -81.2845245094787)
SRID=4326;POINT(31.7408504 34.9883978)
```

If you wish to automate this, you can create a trigger like below to update a geom column within the table or in another table

```
CREATE OR REPLACE FUNCTION togeom() RETURNS trigger
   LANGUAGE plpgsql AS
$$BEGIN
   NEW.geom = (select ST_SetSRID((new.location)::GEOMETRY(POINT), 4326) from mytable␣
↪LIMIT 1);
  RETURN NEW;
END;$$;

CREATE OR REPLACE TRIGGER update_geom
   AFTER INSERT ON mytable FOR EACH ROW
  EXECUTE PROCEDURE togeom();
```

CHAPTER

FIFTEEN

COMMERCIAL SUPPORT

Commercial Support with enhanced features and support is available via AcuGIS

CHAPTER

SIXTEEN

KNOWN ISSUES IN CURRENT RELEASE

1.1.