

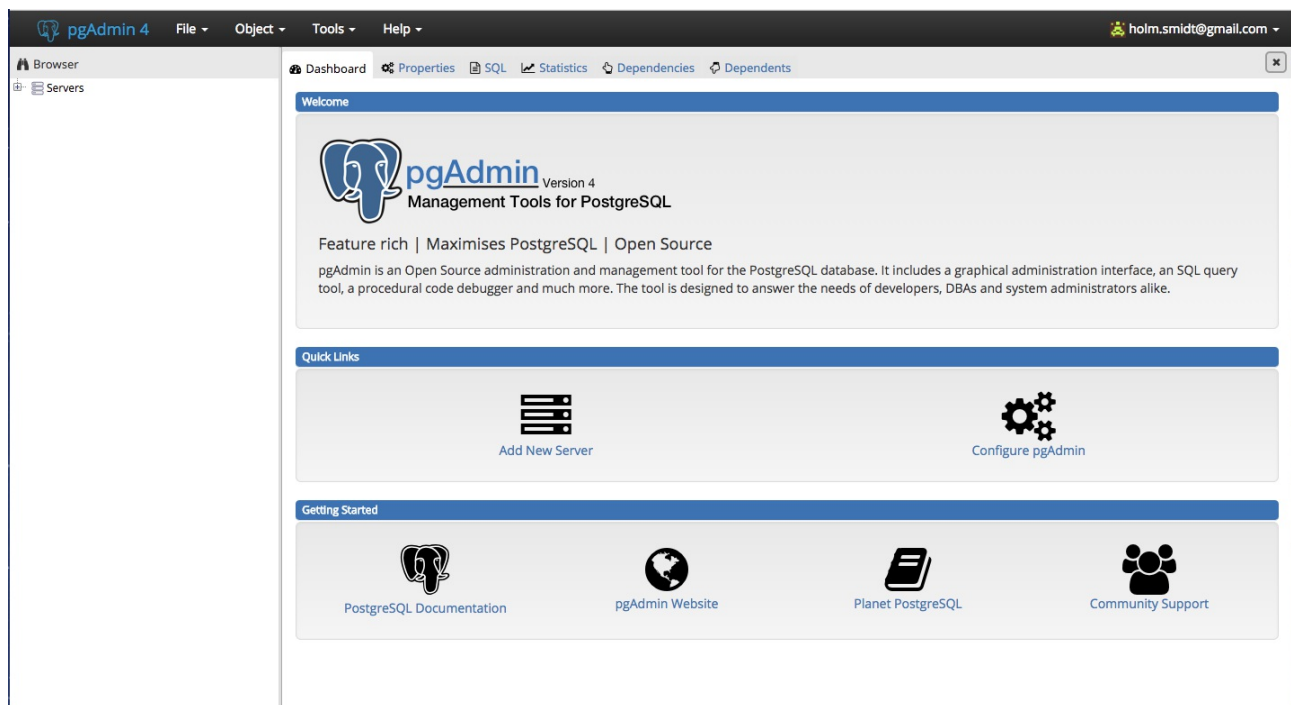
Introduction to Databases

Databases are structured collections of related data. Data are simply facts that can be recorded and accessed. **Data formats** include text, numbers, figures, graphics, images, audio/video recordings and more. In sensor applications, we typically want to store out data in a structured fashion so that we can easily access them. While there are different types of databases, the most fundamental type of database is a relational database. Relational databases are collections of related relations within which each relation has a unique name.

In the next few steps, you will learn how to connect to a database server using a database administration interface (pgadmin), create tables, insert data into tables, and query data from tables.

Connecting to a DB server

First you will need to log in to the adminstration application (pgadmin). Go to <http://deep.outtter.space:55082/browser/>, and login with your credentials (ask instructor for your credentials). You should see this screen.



The first thing to do now is to **Add New Server** using the link in the Quick Link section. The server

configurations are specific to each student.

- Under the **General** tab, you want to give the server a name, e.g. Daniel's DB, and maybe pick your preferred background and foreground colors.

Create - Server

General Connection SSL SSH Tunnel Advanced

Name ITM Copper

Server group Servers

Background [Orange square]

Foreground [Cross-hatch square]

Connect now? ☒

Comments

Either Host name, Address or Service must be specified.

Save Cancel Reset

- Under the **Connection** tab, you want to enter the provided IP for your DB as the Host name. This should be **deep.ottter.space**. The port and maintenance database are **55085** and **postgres** respectively. Lastly, enter your **username** (should be your first name all lower case) and **password** (your first name + 01). To make it easier on yourself, just check the **Save**

password? checkbox.

The screenshot shows a 'Create - Server' dialog box with a blue header bar containing a menu icon and a close button. Below the header is a tabbed interface with five tabs: 'General', 'Connection' (selected), 'SSL', 'SSH Tunnel', and 'Advanced'. The 'Connection' tab contains several form fields: 'Host name/address' with the value '172.17.0.8', 'Port' with '5432', 'Maintenance database' with 'postgres', 'Username' with 'holm', 'Password' with masked characters '.....', 'Save password?' with a checked checkbox, 'Role' (empty), and 'Service' (empty). At the bottom of the dialog are three buttons: 'Save' (blue), 'Cancel' (red), and 'Reset' (orange). On the left side of the bottom bar are two small buttons with an 'i' and a '?' icon.

Field	Value
Host name/address	172.17.0.8
Port	5432
Maintenance database	postgres
Username	holm
Password
Save password?	<input checked="" type="checkbox"/>
Role	
Service	

- Click save and hopefully it will connect you to your DB.
- To debug, or if this fails at any point, make sure that the IP, username, and password are entered correctly. If it still doesn't work, contact the instructor as the IP may have changed due to a container restart.

Creating a table

We use Structured Query Language (SQL) to interface with database systems. First, you need to access the query tool (see below):

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane showing a tree of database objects: Servers (5), Databases (3) including 'asb', 'holm', and 'postgres', Login/Group Roles, Tablespace, and several ITM databases (Carbon, Helium, Neon, Oxygen). The main pane displays the 'Query Tool' menu, which includes options like 'Reload Configuration', 'Pause Replay of WAL', 'Resume Replay of WAL', 'Add Named Restore Point...', 'Import/Export...', 'Maintenance...', 'Backup...', 'Backup Globals...', 'Backup Server...', 'Restore...', and 'Grant Wizard...'. Below the menu, there are four performance graphs: 'Transactions per second' (showing Transactions, Commits, and Rollbacks), 'Tuples out' (showing Fetched and Returned), 'Block I/O' (showing Reads and Hits), and 'Database activity' (showing Sessions, Locks, and Prepared Transactions). The 'Database activity' section includes a table with columns: PID, User, Application, Client, Backend start, State, Wait Event, and Blocking PIDs. The table lists three sessions: PID 272 (User: holm, Application: psql), PID 4470 (User: holm, Application: pgAdmin 4 - DB:asb), and PID 4476 (User: holm, Application: pgAdmin 4 - DB:asb).

	PID	User	Application	Client	Backend start	State	Wait Event	Blocking PIDs
✖	272	holm	psql		2018-07-17 03:27:22 UTC	idle	Client: ClientRead	
✖	4470	holm	pgAdmin 4 - DB:asb	172.17.0.3	2018-07-18 02:47:23 UTC	idle	Client: ClientRead	
✖	4476	holm	pgAdmin 4 - DB:asb	76.173.115.190	2018-07-18 02:49:07 UTC	idle	Client: ClientRead	

Using the query tool (see below), the following code block explains how to create tables:

pgAdmin 4 File Object Tools Help hsmidt@hawaii.edu

Browser

- Extensions
- Foreign Data Wrappers
- Languages
- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Sequences
 - Tables (5)
 - companies
 - company_locations
 - contains
 - funds
 - securities
 - Trigger Functions
 - Types
 - Views
- holm
- postgres
- Login/Group Roles
- Tablespaces
- ITM Carbon
- ITM Helium
- ITM Neon
- ITM Oxygen

Dashboard Properties SQL Statistics Dependencies Dependents Query - asb on holm@ITM 354 DB

asb on holm@ITM 354 DB

```
1 select * from securities;
```

Data Output Explain Messages Notifications Query History

	id	name	type
	integer	character varying (255)	character varying (255)
1	1	security A	typeA

```
CREATE TABLE tablename (
    Att1    DataType ,
    Att2    DataType ,
    ...
);
```

For example:

```
CREATE TABLE vendor (
    vendorid CHAR(2) NOT NULL,
    vendorname VARCHAR(25) NOT NULL,
    PRIMARY KEY (vendorid));
```

Maybe a more relevant example would be

```
create table measurements (
```

```
id int generated by default as identity,  
ts timestamp,  
temperature float,  
humidity float,  
heatindex float,  
primary key (id)  
)
```

The primary key must be unique for every entry, so it's good to generate them by default as an increasing integer.

Inserting Data

Data inserts are very important, since that's how you can insert your sensor data into your database.

You can run this example in your query tool to test it out:

```
insert into measurements (ts, temperature, humidity, heatindex) values (now(), 76.2, 80)
```

Selecting Data

Selecting or reading data from the database is the most commonly used command, the following gives an example on how to select all data from a table:

```
select * from measurements;
```

You can also specify which columns and in which order you want to select them:

```
select ts, temperature, humidity from measurements;
```

Note that here, I did not want the `id` and the `heatindex`.

Congratulations, you set up your first database. Now we'll see how you can insert data from node-red.

Node-Red
