PostgreSQL Quick Start

• Prerequisites — for this and pretty much any open source C package:
  ◇ A working C compiler (gcc works quite well, and is probably what you already have anyway)
  ◇ Any archival program that can open .tar.gz or .tar.bz files (tar or gnutar are just fine — and you probably already have that, too)

• Source available from: http://www.postgresql.org
• Unroll the tarball/archive
• Key document to read: INSTALL

<table>
<thead>
<tr>
<th>Action</th>
<th>Alternative(s)</th>
<th>What It Does</th>
</tr>
</thead>
<tbody>
<tr>
<td>./configure</td>
<td>Add arguments to ./configure</td>
<td>Scans your system and figures out what it can/can’t do</td>
</tr>
<tr>
<td>gmake</td>
<td>make (a.k.a. GNU make)</td>
<td>Builds PostgreSQL</td>
</tr>
<tr>
<td>su</td>
<td>sudo command</td>
<td>The next four actions must be performed as the superuser</td>
</tr>
<tr>
<td>gmake install</td>
<td>make install</td>
<td>Installs PostgreSQL into its final destination (pg_dir in these notes)</td>
</tr>
<tr>
<td>adduser postgres</td>
<td>GUI</td>
<td>Creates the PostgreSQL user</td>
</tr>
<tr>
<td>mkdir db_dir</td>
<td>If doing this where the PostgreSQL user can already create directories, then just mkdir as that user</td>
<td>Creates the database directory</td>
</tr>
<tr>
<td>chown postgres db_dir</td>
<td></td>
<td>Makes sure that the PostgreSQL user owns that directory</td>
</tr>
<tr>
<td>su – postgres</td>
<td>sudo –u postgres command</td>
<td>The next two actions must be performed as the PostgreSQL user (the last two don’t have to be, with PostgreSQL users properly set up)</td>
</tr>
<tr>
<td>pg_dir/bin/initdb –D db_dir</td>
<td></td>
<td>Creates the physical PostgreSQL database</td>
</tr>
<tr>
<td>pg_dir/bin/postmaster –D db_dir &gt;logfile &gt;2&gt;&amp;1 &amp;</td>
<td>Use the pg_ctl script</td>
<td>Starts the PostgreSQL database server</td>
</tr>
<tr>
<td>pg_dir/bin/createdb db_name</td>
<td></td>
<td>Creates a “logical” database</td>
</tr>
<tr>
<td>pg_dir/bin/psql db_name</td>
<td></td>
<td>Connects to the database</td>
</tr>
</tbody>
</table>
Variations on the Theme

- You can customize a lot of things at build time — read the rest of INSTALL (for example, you can enable SSL by adding `--with-openssl` to your `.configure` invocation; you can enable Rendezvous/Bonjour [great for Mac OS X] by adding `--with-rendezvous`)
- You can create database-level users (that is, users for the database, not for logging in to your overall system) — use the `createuser` command
- Other commands such as `createdb`, `psql`, etc., can be executed under the guise of this database-level user

psql is Your Friend

- Get to know it and love it — until you start coding external applications, and even while you’re doing that, you’ll spend a lot of time here
- GUI layers exist, if you really hate the command line
- General guidelines:
  ◆ If you type something in directly, `psql` assumes it is SQL — end these with a semi-colon (`;`)
  ◆ Backslash (`\`) is the `psql` escape key — it invokes `psql`'s own command set
  ◆ Start with `\?` for general `psql` help and `\h` for SQL help
Really Really Basic SQL

• To create a (really really basic) table:
  
  create table table_name ( column_list );

• The column list is a comma-separated list of names and data types (e.g. int, varchar, float…look ‘em up)

• For example:
  
  create table product (name varchar, description varchar, price int, weight float);

• Type “\h create table” in psql for more details

• You can use “\d table_name” to inspect your creation

• Add data to your table this way:

  insert into table [ ( columns ) ] values ( expressions );

• columns and expressions are comma-separated lists of column names and expressions (strings, numbers, etc.), respectively — they are supposed to correspond

• columns is optional, but recommended — otherwise, SQL sticks your expressions in the order that the columns are stored by the database

• Thus:

  insert into product (name, description, price, weight) values ('Widget', 'A generic widget', 5000, 4.8);

• Use single quotes for strings
Look Who’s Querying

• Let’s say you’ve invoked insert a few million times…
you probably want to look at your data now — here’s
a very distilled form:

\[
\text{select [ distinct ] ( * | expression\_list ) from source}
\text{ where condition;}
\]

• \textit{expression\_list} is a comma-separated list of fields,
among other things

• \textit{source} is typically a table, for this first simple cut

• \textit{condition} is a boolean SQL expression

• As the main way for retrieving data from the database,
the full-blown select statement is way more involved
than this; try “\h select” in \texttt{psql} to see

• But anyway, here are some examples:

\[
\text{select name, description from product where price > 1000 and weight > 2.5;}
\]

\[
\text{select name from product where description like '% computer%'; -- % is used by \texttt{like} as a wildcard}
\]

• By the way, “--” is the comment delimiter, so the
example above can be entered in its entirety into \texttt{psql}
— it just ignores everything after “--”

• Parentheses, \texttt{and}, \texttt{or}, and \texttt{not} function as you would
expect within the \texttt{where} clause
Other SQL Things to Try

• As you can see, the basics aren’t so bad. Other commands to look up and try out:
  ◇ update — changes existing data
  ◇ delete — erases data
  ◇ alter table — modifies a table’s structure
  ◇ Here’s a fun one: try explain, followed by a select statement, e.g. explain select * from product

• Use “\h” liberally if needed, or look it up in the book; the Web will also have a lot of information

Useful PostgreSQL Functions

• SQL handles activities “within” a database — PostgreSQL (or the overall database manager software) handles other housekeeping activities

• Backups, copies, transfers: the pg_dump command can copy an entire database to a regular file, to which you can do pretty much anything

• Scripting: you’ve seen psql in “interactive” mode; you can also use it for batch/script jobs
  1. Write your sequence of commands as a text file
  2. Feed it to psql — you can either redirect (< or |), use the –f option, or use the \i command from the psql prompt