Getting started with PostgreSQL

Gavin Sherry
`gavin@alcove.com.au`

Alcove Systems Engineering

January 16, 2007

Outline

Installing PostgreSQL and start

- Just do it through your package manager!

Fedora/RHEL/CentOS

```
# yum install postgresql
# service postgresql start
```

Debian/etc

```
# apt-get search postgresql
# /etc/init.d/postgresql-<version> start
```

Windows installer

- Just run the `.msi`
- `http://www.postgresql.org/download/`

Source

```
# ./configure ; make ; make install
# initdb /path/to/data
# pg_ctl -D /path/to/data start
```

It's that simple!

So, lets go

Example

```
$ psql postgres
Welcome to psql 8.2.1, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

postgres=# \d
No relations found.
postgres=# create table foo (i int, t text);
CREATE TABLE
postgres=# insert into foo values(1, 'Hello World');
INSERT 0 1
postgres=# select * from foo;
 i |      t
---+-------------
 1 | Hello World
(1 row)
```

Outline

FATAL...?

What's this?

psql: FATAL: IDENT authentication failed for user "gavin"

- Some packages tighten up default security
- What to do:

```
su - postgres
createuser gavin
```

- Alternatively, you could relax the authentication – but I wont tell you how!

Exploring

Example

```
# select version();
                 version
------------------------------------------
 PostgreSQL 8.2.1 ...
# select table_catalog, table_name, table_type from
   information_schema.tables where
   table_schema = 'public';
 table_catalog | table_name | table_type
---------------+------------+------------
 postgres      | foo        | BASE TABLE
(3 rows)
```

More exploring

Example

```
# select column_name, data_type from
   information_schema.columns
   where table_schema='public' and
   table_name = 'foo';
 column_name | data_type
-------------+-----------
 i           | integer
 t           | text
(2 rows)
```

Outline

A little project

- Say you want to develop a blog – “everyone’s doing it”
- You want to:
    1. Store articles
    2. Relate articles to blog users
    3. Keep statistics on blog posts

A little project

- Say you want to develop a blog – "everyone's doing it"
- You want to:
    1. Store articles
    2. Relate articles to blog users
    3. Keep statistics on blog posts

A little project

- Say you want to develop a blog – "everyone's doing it"
- You want to:
  1. Store articles
  2. Relate articles to blog users
  3. Keep statistics on blog posts

A little project

- Say you want to develop a blog – “everyone’s doing it”
- You want to:
    1. Store articles
    2. Relate articles to blog users
    3. Keep statistics on blog posts

blogdb – version 1

SQL

```
-- Create a table of users (this is a comment, btw)
CREATE TABLE users (usrid int, username text, email text);

-- insert a user (SQL isn’t case sensitive)
insert into users values(1, ’Gavin’,
        ’gavin@alcove.com.au’);

-- create the articles table
CREATE TABLE articles (artid int, title text, body text,
        dt timestamp, usrid int);

-- add a post
insert into articles values(1, ’First post’, ’Yay’,
        current_timestamp, 1);

-- get article count
SELECT count(*) FROM articles;
```

blogdb – version 1

SQL

```
-- Create a table of users (this is a comment, btw)
CREATE TABLE users (usrid int, username text, email text);


-- insert a user (SQL isn’t case sensitive)
insert into users values(1, ’Gavin’,
        ’gavin@alcove.com.au’);


-- create the articles table
CREATE TABLE articles (artid int, title text, body text,
        dt timestamp, usrid int);


-- add a post
insert into articles values(1, ’First post’, ’Yay’,
        current_timestamp, 1);


-- get article count
SELECT count(*) FROM articles;
```

blogdb – version 1

SQL

```
-- Create a table of users (this is a comment, btw)
CREATE TABLE users (usrid int, username text, email text);


-- insert a user (SQL isn’t case sensitive)
insert into users values(1, ’Gavin’,
        ’gavin@alcove.com.au’);


-- create the articles table
CREATE TABLE articles (artid int, title text, body text,
        dt timestamp, usrid int);


-- add a post
insert into articles values(1, ’First post’, ’Yay’,
        current_timestamp, 1);


-- get article count
SELECT count(*) FROM articles;
```

blogdb – version 1

SQL

```
-- Create a table of users (this is a comment, btw)
CREATE TABLE users (usrid int, username text, email text);


-- insert a user (SQL isn't case sensitive)
insert into users values(1, 'Gavin',
        'gavin@alcove.com.au');


-- create the articles table
CREATE TABLE articles (artid int, title text, body text,
        dt timestamp, usrid int);


-- add a post
insert into articles values(1, 'First post', 'Yay',
        current_timestamp, 1);


-- get article count
SELECT count(*) FROM articles;
```

blogdb – version 1

SQL

```
-- Create a table of users (this is a comment, btw)
CREATE TABLE users (usrid int, username text, email text);


-- insert a user (SQL isn't case sensitive)
insert into users values(1, 'Gavin',
        'gavin@alcove.com.au');


-- create the articles table
CREATE TABLE articles (artid int, title text, body text,
        dt timestamp, usrid int);


-- add a post
insert into articles values(1, 'First post', 'Yay',
        current_timestamp, 1);


-- get article count
SELECT count(*) FROM articles;
```

What’s wrong with version 1?

- No data integrity
- No permissions
- `users` is a common table name – what about other applications with a `users` table?

What's wrong with version 1?

- No data integrity
- No permissions
- `users` is a common table name – what about other applications with a `users` table?

What’s wrong with version 1?

- No data integrity
- No permissions
- `users` is a common table name – what about other applications with a `users` table?

What’s wrong with version 1?

- No data integrity
- No permissions
- `users` is a common table name – what about other applications with a `users` table?

What’s wrong with version 1?

- No data integrity
- No permissions
- `users` is a common table name – what about other applications with a `users` table?

Data integrity

- There are levels of integrity
- Referential integrity
    - Enforce a relationship
- Guard against duplication
    - Each row must have a key
- Some data must look a certain way – like dates and times
- Some data cannot be 'undefined' or `NULL`

blogdb – version 2

SQL

```
CREATE TABLE users (usrid serial primary key,
        username text NOT NULL,
        email text NOT NULL,
        unique(username));

INSERT INTO users (username, email) values(1, 'Gavin',
        'gavin@alcove.com.au');

CREATE TABLE articles (artid serial primary key,
        title text NOT NULL,
        body text NOT NULL,
        dt timestamp default current_timestamp,
        usrid int references users(usrid));
```

blogdb – version 2

SQL

```
CREATE TABLE users (usrid serial primary key,
        username text NOT NULL,
        email text NOT NULL,
        unique(username));

INSERT INTO users (username, email) values(1, ’Gavin’,
        ’gavin@alcove.com.au’);

CREATE TABLE articles (artid serial primary key,
        title text NOT NULL,
        body text NOT NULL,
        dt timestamp default current_timestamp,
        usrid int references users(usrid));
```

blogdb – version 2

SQL

```
CREATE TABLE users (usrid serial primary key,
        username text NOT NULL,
        email text NOT NULL,
        unique(username));

INSERT INTO users (username, email) values(1, 'Gavin',
        'gavin@alcove.com.au');

CREATE TABLE articles (artid serial primary key,
        title text NOT NULL,
        body text NOT NULL,
        dt timestamp default current_timestamp,
        usrid int references users(usrid));
```

Permissions and privileges

- We speak of `CREATE ROLE` and `GRANT`

SQL

```
CREATE ROLE blogread LOGIN;
CREATE ROLE blogwrite LOGIN;
...
GRANT SELECT ON articles,users
    TO blogread;
GRANT ALL ON articles,users
    TO blobwrite;
```

Name space issues

- This is why SQL supports "schemas"

SQL

```
CREATE SCHEMA blog;
SET search_path = blog;
CREATE ...

SELECT count(*) FROM blog.articles;
```

Further exercises

- Data integrity for email addresses – See `CREATE DOMAIN`
- Speed up `count(*)` in the presence of a large `articles` table – see the advanced lecture today

Outline

More on PostgreSQL

- What's with the name? → **Post** (in) **gres**, with **SQL** support
  - Hey, it used to be named POSTQUEL
  - Then again, it used to have a hell of a lot of Lisp in it
- Academic project in the 80s, one of **Michael Stonebraker's** projects at Berkeley
- BSD license – do what you want with the source (don't sue us!)
- Now developed by a group of enthusiasts

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, ...
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- ...

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Buzzword compliance

What have we got?

- JDBC, .NET, ODBC, PHP, Python, Ruby, . . .
- Replication
- Views, Indexes, Schemas, Foreign keys
- Partitioning
- Two phase commit
- Transations and nested transactions!
- Functions/stored procedures in SQL, Java, Ruby, Python, and much more
- Online backups
- . . .

Who uses PostgreSQL?

- Businesses, big and small
- Hobbyists and developers
- Big operators like
  - Fujitsu
  - Sun
  - HP
  - Apple
  - Cisco
  - AC Neilson
  - Skype
  - Just about every Australian government department
  - Safeway (North America)
  - .org, .info, .in, . . .
  - Sony
- But it's even more popular with small businesses

What else can you do?

- Report writing: Jasper, Crystal, . . .
- Wiki, CMS backend
- All the other OSS jazz – forums, photo galleries, . . .
- Bugzilla backend
- LDAP backend
- J2EE backend
- Ruby on rails, out of the box
- GIS – 100% standards compliant!
- Statistical analysis – full integration of 'R'
- Data warehousing

Outline

Resources for learning more

- The PostgreSQL manual - `http://www.postgresql.org/docs`
- Books (some free) - `http://www.postgresql.org/docs/books/`
- GUI console - `http://www.pgadmin.org`
- Mailing lists - `http://www.postgresql.org/`
- IRC - `irc.freenode.net`, `#postgresql`
  - My nick is swm
- SydPUG - `http://pugs.postgresql.org/sydpug/`

Stuff for the boss

- Case studies:
  `http://www.postgresql.org/about/casestudies/`
- Featured users:
  `http://www.postgresql.org/about/users/`
- A glossy brochure:
  `http://alcove.com.au/postgresql.pdf`