

# Releasing Extensions on PGXN

David E. Wheeler  
PostgreSQL Experts, Inc.

PGCon  
May 20, 2011



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.



# Problem Solved

# Problem Solved

- Solved a database problem

# Problem Solved

- ⦿ Solved a database problem
- ⦿ Want to share

# Problem Solved

- ⦿ Solved a database problem
- ⦿ Want to share
- ⦿ Open source it

# Problem Solved

- ⦿ Solved a database problem
- ⦿ Want to share
- ⦿ Open source it
- ⦿ Where to distribute

# PGXN

# PGXN

“The PostgreSQL Extension Network is a central distribution system for open-source PostgreSQL extension libraries and utilities.”



# PGXN

PostgreSQL Extension Network

[recent](#) [users](#) [about](#) [faq](#)

In Documentation

analyze data types datatype explain explain analyze  
france hash key value key value pair md5 node ordered  
pair pair plan semantic version semver sha sha1  
statistics table tap tddd test driven database development  
testing unit testing variadic function version version  
number

PGXN, the PostgreSQL Extension network, is a central distribution system for open-source PostgreSQL extension libraries.

## Founders

**myYearbook**

**PGX**

**DALIBO**

## Patrons



**Enova Financial**

## Benefactors

→ [Etsy](#)

→ [US PostgreSQL Association](#)



# PGXN

PostgreSQL Extension Network

[recent](#) [users](#) [about](#) [faq](#)

wheeler

In Documentation

[PGXN Search](#)

analyze data types datatype explain explain analyze  
france hash key value key value pair md5 node ordered  
pair pair plan semantic version semver sha sha1  
statistics table tap tddd test driven database development  
testing unit testing variadic function version version  
number

PGXN, the PostgreSQL Extension network, is a central distribution system for open-source PostgreSQL extension libraries.

## Founders

**myYearbook**

**PGX**

**DALIBO**

## Patrons



**Enova Financial**

## Benefactors

→ [Etsy](#)

→ [US PostgreSQL Association](#)



# PGXN

PostgreSQL Extension Network

[recent](#) [users](#) [about](#) [faq](#)

wheeler

In Documentation

PGXN Search

analyze data types datatype explain explain analyze  
france hash key value key value pair md5 node ordered  
pair pair plan semantic version semver sha sha1  
statistics table tap tddd test driven database development  
testing unit testing variadic function version version  
number

PGXN, the PostgreSQL Extension network, is a central distribution system for open-source PostgreSQL extension libraries.

## Founders

**myYearbook**

**PGX**

**DALIBO**

## Patrons



**Enova Financial**

## Benefactors

→ [Etsy](#)

→ [US PostgreSQL Association](#)



# PGXN

PostgreSQL Extension Network

wheeler

in Documentation

PGXN Search

recent

users

about

faq

1-4 of 4 found

## [pair 0.1.2](#)

Author David E. Wheeler Copyright and License Copyright (c) 2010-2011 David E. Wheeler. This module is free software; you can redistribute it and/or modify it under the PostgreSQL License.

[pair 0.1.2](#) • 2011-04-20 • David E. Wheeler

## [semver 0.2.1](#)

Author David E. Wheeler Sam Vilain Copyright and License Copyright (c) 2010-2011 David E. Wheeler and Sam Vilain. This module is free software; you can redistribute it and/or modify it under the...

[semver 0.2.1](#) • 2011-04-20 • David E. Wheeler

## [explanation 0.2.0](#)

Author David E. Wheeler, PostgreSQL Experts, Inc.. Copyright and License Copyright (c) 2010-2011, Marchex. All rights reserved. Redistribution and use in source and binary forms, with or without...

[explanation 0.2.0](#) • 2011-02-21 • David E. Wheeler

## [pgTAP 0.25.0](#)

\_nick = 'theory'; SELECT row\_eq(testrow, ROW(1, theory, 'David Wheeler')::users); Compares the contents of a single row to a record.  
Works on PostgreSQL 8.1 and higher.

[pgTAP 0.25.0](#) • 2011-02-02 • David E. Wheeler



# PGXN

PostgreSQL Extension Network

wheeler

in Documentation

PGXN Search

recent

users

about

faq

1-4 of 4 found

## pair 0.1.2

Author David E. Wheeler Copyright and License Copyright (c) 2010-2011 David E. Wheeler. This module is free software; you can redistribute it and/or modify it under the PostgreSQL License.

pair 0.1.2 • 2011-04-20 • David E. Wheeler

## semver 0.2.1

Author David E. Wheeler Sam Vilain Copyright and License Copyright (c) 2010-2011 David E. Wheeler and Sam Vilain. This module is free software; you can redistribute it and/or modify it under the...

semver 0.2.1 • 2011-04-20 • David E. Wheeler

## explanation 0.2.0

Author David E. Wheeler, PostgreSQL Experts, Inc.. Copyright and License Copyright (c) 2010-2011, Marchex. All rights reserved. Redistribution and use in source and binary forms, with or without...

explanation 0.2.0 • 2011-02-21 • David E. Wheeler

## pgTAP 0.25.0

\_nick = 'theory'; SELECT row\_eq(testrow, ROW(1, theory, 'David Wheeler')::users); Compares the contents of a single row to a record.  
Works on PostgreSQL 8.1 and higher.

pgTAP 0.25.0 • 2011-02-02 • David E. Wheeler

**PGXN**

PostgreSQL Extension Network

**Contents**

- [explanation 0.2.0](#)
  - [Synopsis](#)
  - [Usage](#)
  - [Specifying Columns](#)
  - [Example](#)
  - [Author](#)
  - [Copyright and License](#)

# explanation 0.2.0

This extension adds a new function, `explanation()`, to your database. Pass it a string that executes a query and the function runs `EXPLAIN` on the query and returns the results as a table. Each node in the plan is represented by a single row, and child nodes refer to the unique identifier of their parents. The results, that is, are organized into a proximity tree.

## Synopsis

Plan a simple query:

```
SELECT node_type, strategy, actual_startup_time, actual_total_time
  FROM explanation(
    query := $$ SELECT * FROM pg_class WHERE relname = 'users' $$,
    analyzed := true
  );
```

## Output:

node_type	strategy	actual_startup_time	actual_total_time
Index Scan		00:00:00.000017	00:00:00.000017

**PGXN**

PostgreSQL Extension Network

**Contents**

- [explanation 0.2.0](#)
  - [Synopsis](#)
  - [Usage](#)
  - [Specifying Columns](#)
  - [Example](#)
  - [Author](#)
  - [Copyright and License](#)

# explanation 0.2.0

This extension adds a new function, `explanation()`, to your database. Pass it a string that executes a query and the function runs `EXPLAIN` on the query and returns the results as a table. Each node in the plan is represented by a single row, and child nodes refer to the unique identifier of their parents. The results, that is, are organized into a proximity tree.

## Synopsis

Plan a simple query:

```
SELECT node_type, strategy, actual_startup_time, actual_total_time
  FROM explanation(
    query := $$ SELECT * FROM pg_class WHERE relname = 'users' $$,
    analyzed := true
  );
```

## Output:

node_type	strategy	actual_startup_time	actual_total_time
Index Scan		00:00:00.000017	00:00:00.000017

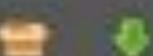


# PGXN

PostgreSQL Extension Network

[theory](#) > [explanation](#)[recent](#) [users](#) [about](#) [faq](#)

## explanation

**This Release:** explanation 0.2.0**Date:** 2011-02-21**Status:** Stable**Abstract:** Turn an explain plan into a table of nodes organized as a proximity tree**Description:** Sometimes you want to be able to save an explain plan for later analysis and querying. This extension does that for you.**Released By:** theory**License:** The (three-clause) BSD License**Resources:** [git](#) • [repo](#) • [bugs](#)**Special Files:** [Changes](#) • [README.md](#) • [META.json](#) • [Makefile](#)**Tags:** [explain](#) • [explain analyze](#) • [analyze](#) • [table](#) • [statistics](#) • [node](#) • [plan](#)

### Extensions

#### explanation 0.2.0

Turn an explain plan into a table of nodes organized as a proximity tree

#### README

# explanation

This Release: explanation 0.2.0

Date: 2011-02-21

Status: Stable

Abstract: Turn an explain plan into a table of nodes organized as a proximity tree

Description: Sometimes you want to be able to save an explain plan for later analysis and querying. This extension does that for you.

Released By: theory

License: The (three-clause) BSD License

Resources: [git](#) • [repo](#) • [bugs](#)

Special Files: [Changes](#) • [README.md](#) • [META.json](#) • [Makefile](#)

Tags: [explain](#) • [explain analyze](#) • [analyze](#) • [table](#) • [statistics](#) • [node](#) • [plan](#)

## Extensions

### explanation 0.2.0

Turn an explain plan into a table of nodes organized as a proximity tree

## README

# explanation 0.2.0

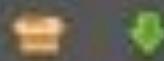
This extension adds a new function, `explanation()`, to your database. Pass it a string that executes a query and the function runs `EXPLAIN` on the query and returns the results as a table. Each node in the plan is represented by a single row, and child nodes refer to the unique identifier of their parents. The results, that is, are organized into a proximity tree.

## Installation

To build it, just do this:

[theory](#) > [explanation](#)[recent](#) [users](#) [about](#) [faq](#)

## explanation

**This Release:** explanation 0.2.0**Date:** 2011-02-21**Status:** Stable**Abstract:** Turn an explain plan into a table of nodes organized as a proximity tree**Description:** Sometimes you want to be able to save an explain plan for later analysis and querying. This extension does that for you,**Released By:** theory**License:** The (three-clause) BSD License**Resources:** [git](#) • [repo](#) • [bugs](#)**Special Files:** [Changes](#) • [README.md](#) • [META.json](#) • [Makefile](#)**Tags:** [explain](#) • [explain analyze](#) • [analyze](#) • [table](#) • [statistics](#) • [node](#) • [plan](#)

### Extensions

#### [explanation 0.2.0](#)

Turn an explain plan into a table of nodes organized as a proximity tree

### README

## explanation 0.2.0



# PGXN

PostgreSQL Extension Network

[recent](#) [users](#) [about](#) [faq](#)

## David E. Wheeler

**Nickname:** theory**Email:** [devide@justatheory.com](mailto:devide@justatheory.com)**URL:** <http://justatheory.com/>**Twitter:** theory

### Distributions

<a href="#"></a>	<a href="#">explanation 0.2.0</a>	Turn an explain plan into a table of nodes organized as a proximity tree	2011-02-21	<a href="#"></a>	<a href="#"></a>
<a href="#"></a>	<a href="#">pair 0.1.2</a>	A key/value pair data type	2011-04-20	<a href="#"></a>	<a href="#"></a>
<a href="#"></a>	<a href="#">pgTAP 0.25.0</a>	Unit testing for PostgreSQL	2011-02-02	<a href="#"></a>	<a href="#"></a>
<a href="#"></a>	<a href="#">semver 0.2.1</a>	A semantic version data type	2011-04-20	<a href="#"></a>	<a href="#"></a>

# Your Solution

# Your Solution

- You've solved a problem

# Your Solution

- ⦿ You've solved a problem
- ⦿ Using database objects

# Your Solution

- ⦿ You've solved a problem
- ⦿ Using database objects
- ⦿ Packaged like contrib

# Your Solution

- ⦿ You've solved a problem
- ⦿ Using database objects
- ⦿ Packaged like contrib
- ⦿ Want to open-source it

# Your Solution

- You've solved a problem
- Using database objects
- Packaged like contrib
- Want to open-source it
- How to distribute on PGXN?

# Your Solution

- ⦿ You've solved a problem
- ⦿ Using database objects
- ⦿ Packaged like contrib
- ⦿ Want to open-source it
- ⦿ How to distribute on PGXN?
- ⦿ Just one thing:

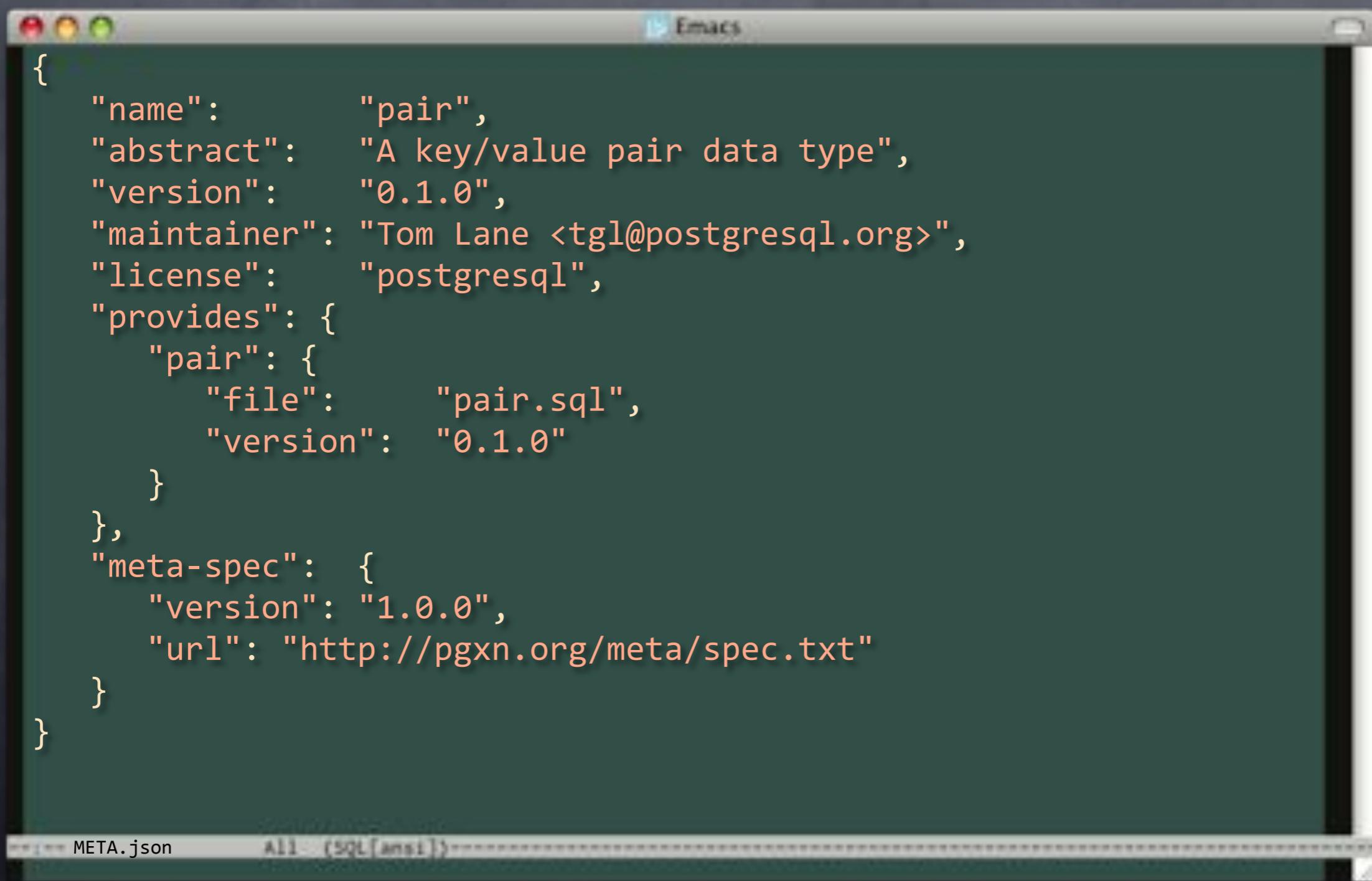
# Your Solution

- ⦿ You've solved a problem
- ⦿ Using database objects
- ⦿ Packaged like contrib
- ⦿ Want to open-source it
- ⦿ How to distribute on PGXN?
- ⦿ Just one thing:
- ⦿ **META.json**

# META.json



# META.json

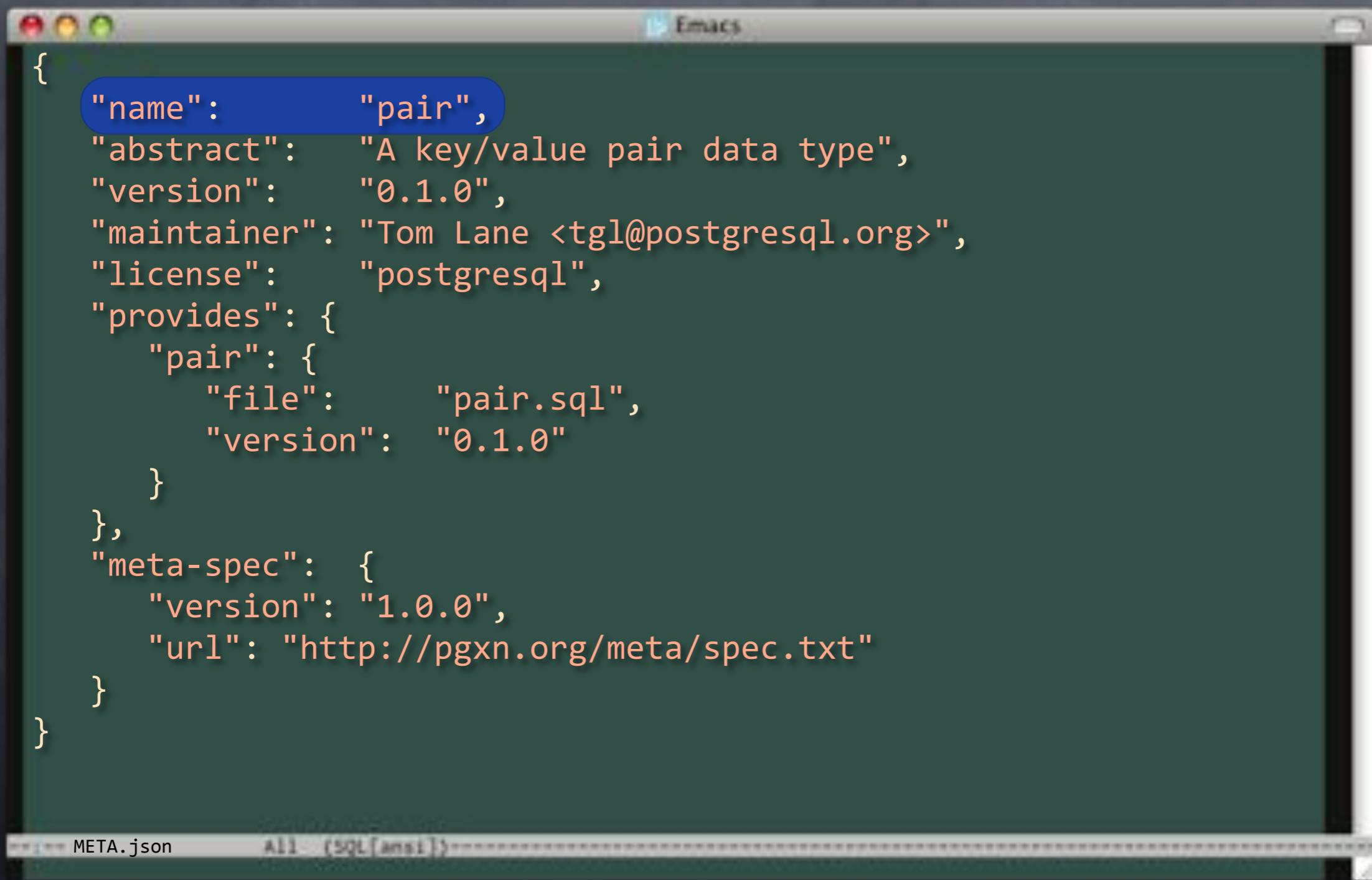


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains a JSON object representing a package specification. The code is as follows:

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

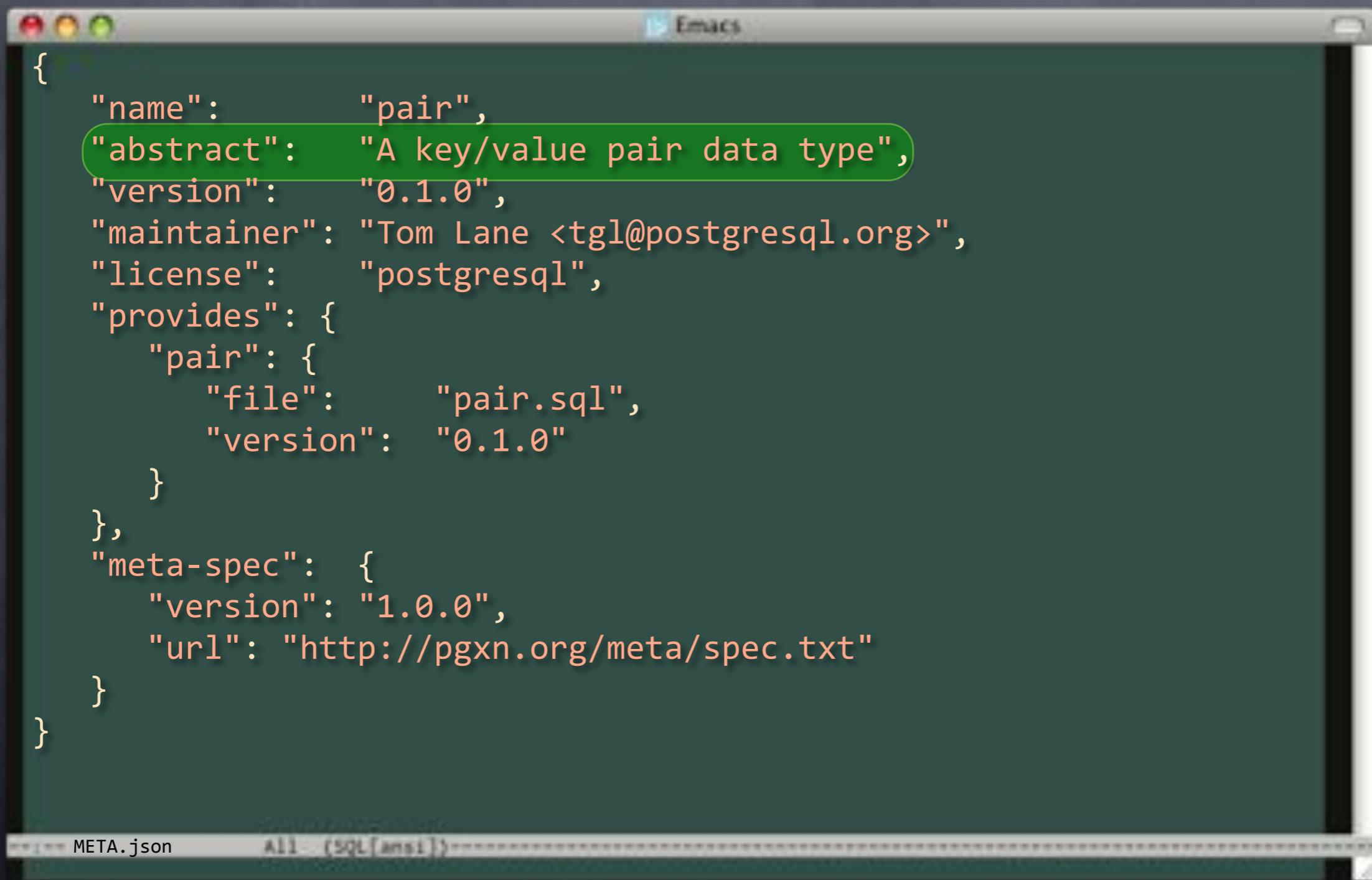
The status bar at the bottom shows "META.json" and "All (SQL [ansi])".

# META.json



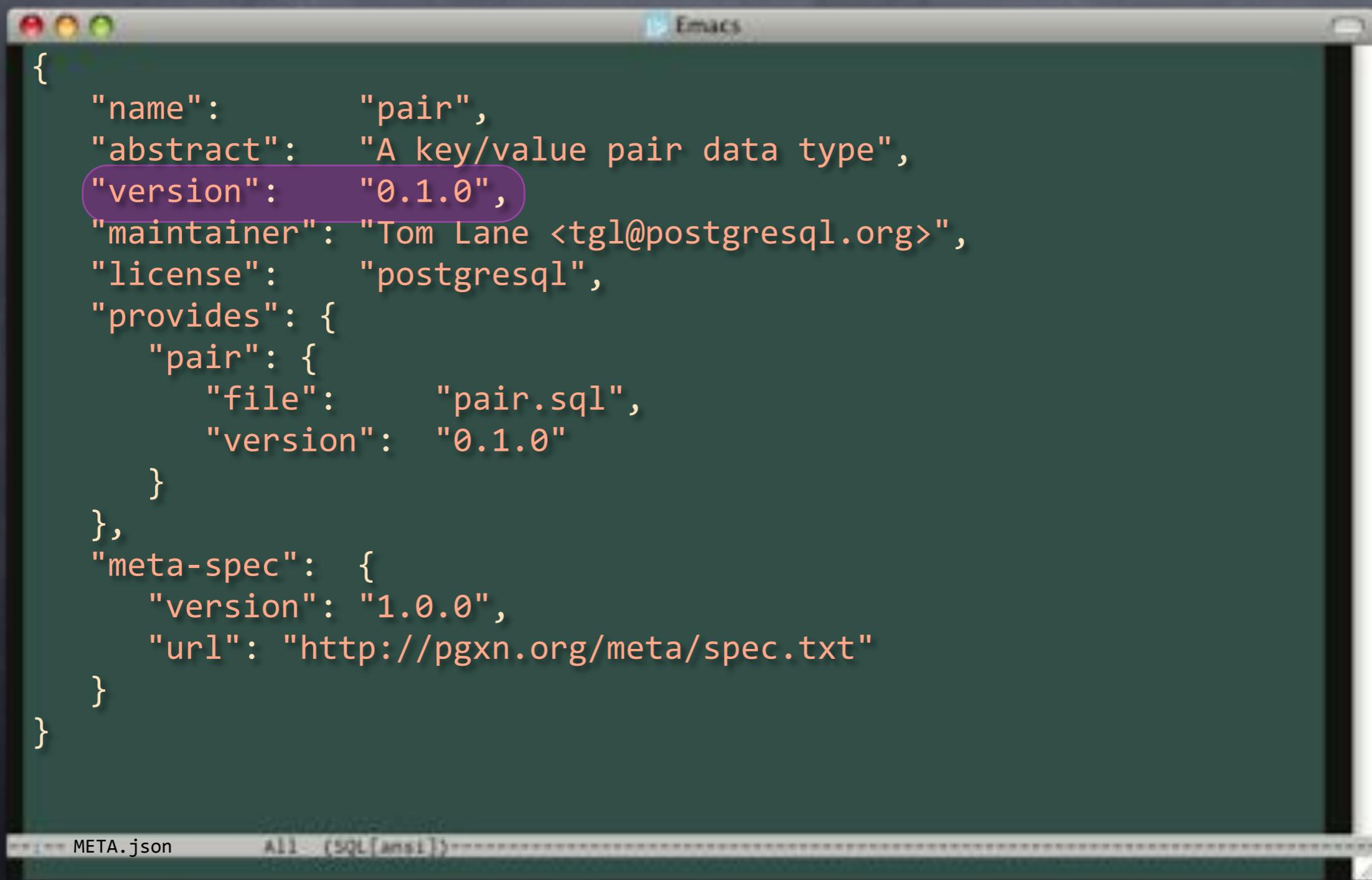
```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

# META.json



```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

# META.json



```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

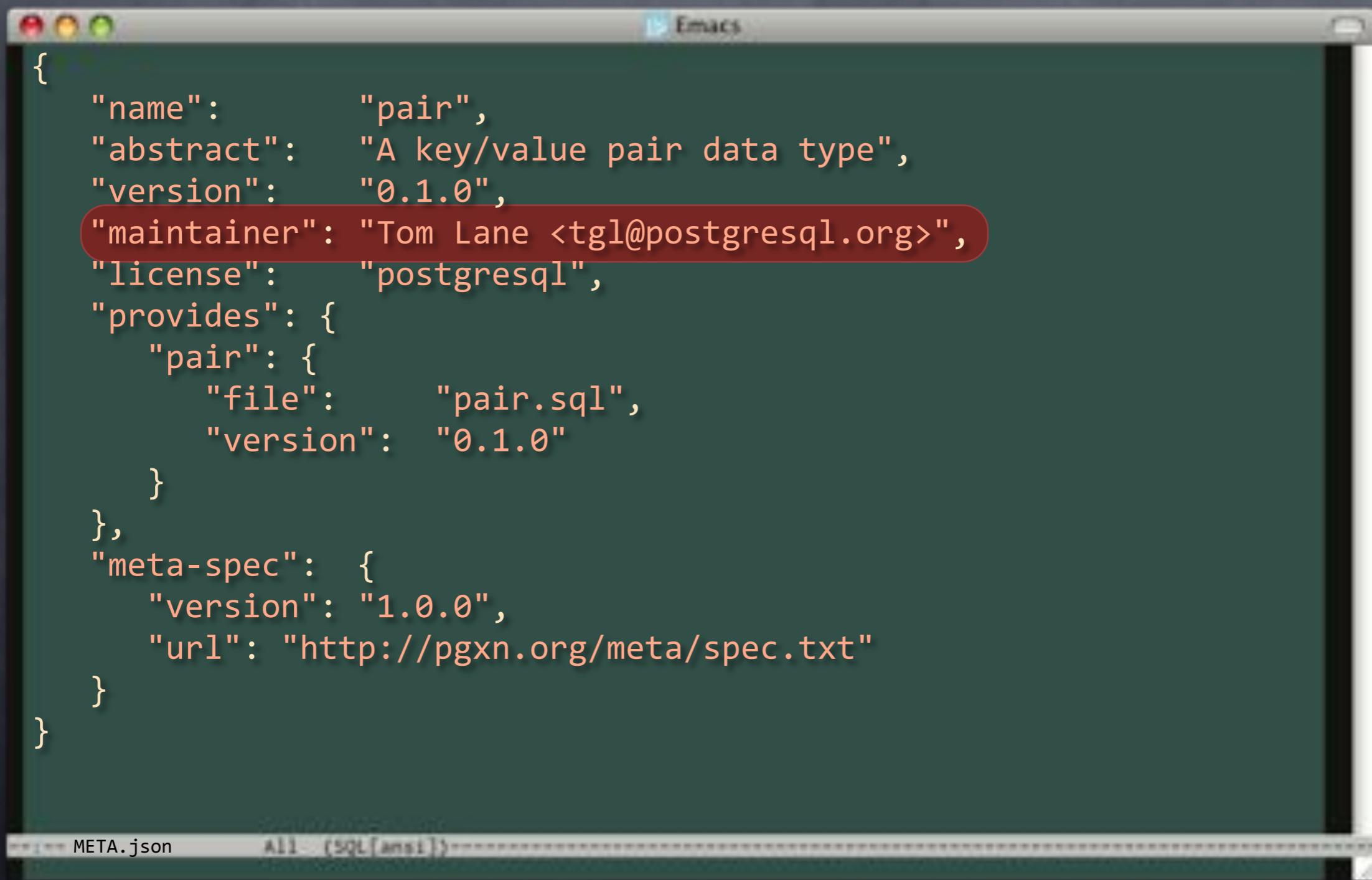
# META.json



```
{  
  "name": "pa...  
  "abstract": "A key/v...  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

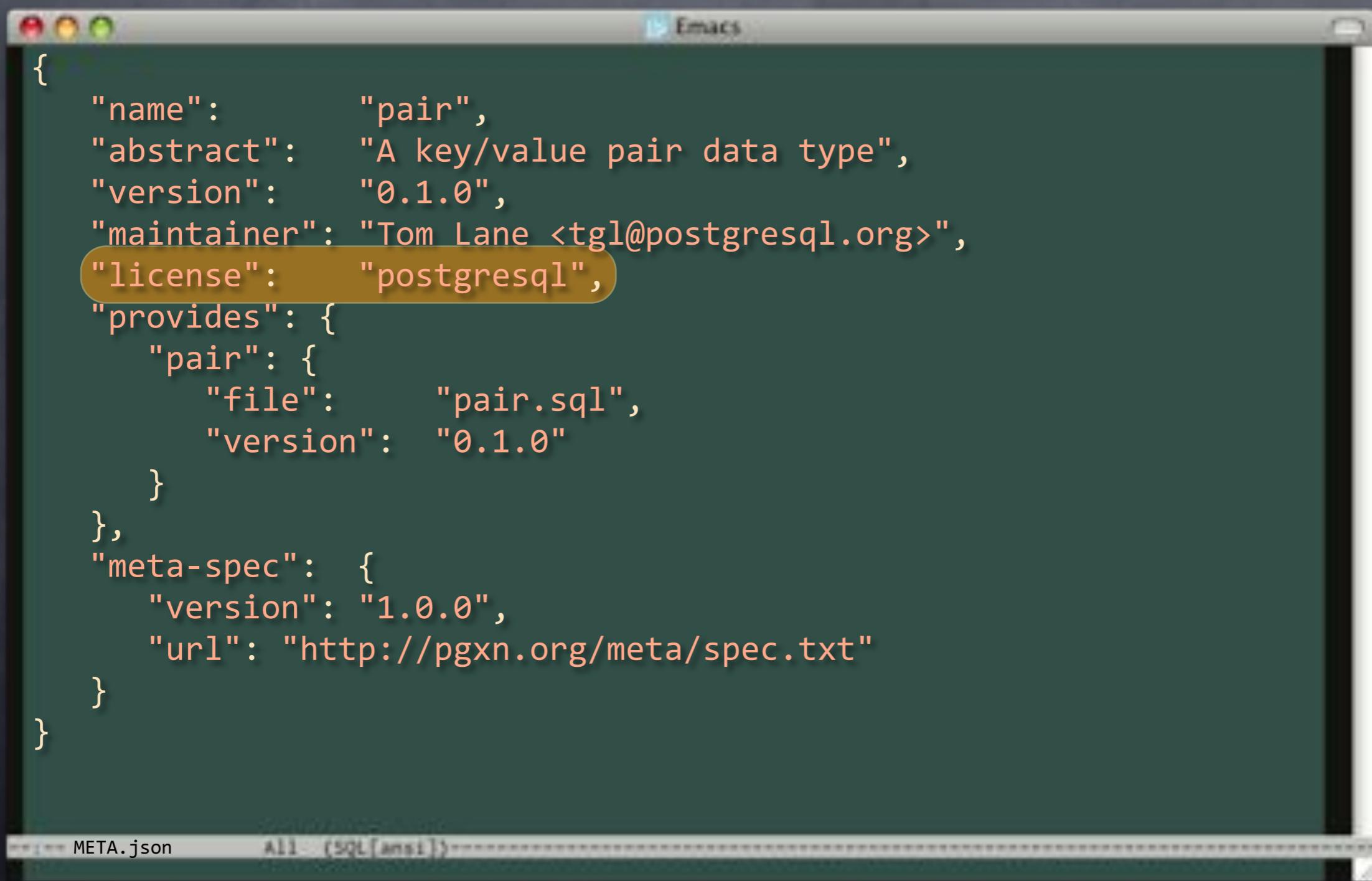
The image shows a screenshot of a terminal window displaying a `META.json` file. The file contains JSON data with several fields: `name`, `abstract`, `version` (which is highlighted with a purple oval), `maintainer`, `license`, `provides`, and `meta-spec`. A blue speech bubble with the text `semver.org` is positioned above the `version` field. A tooltip-like arrow points from the `version` field towards the speech bubble. The terminal window has a dark background and a green title bar.

# META.json



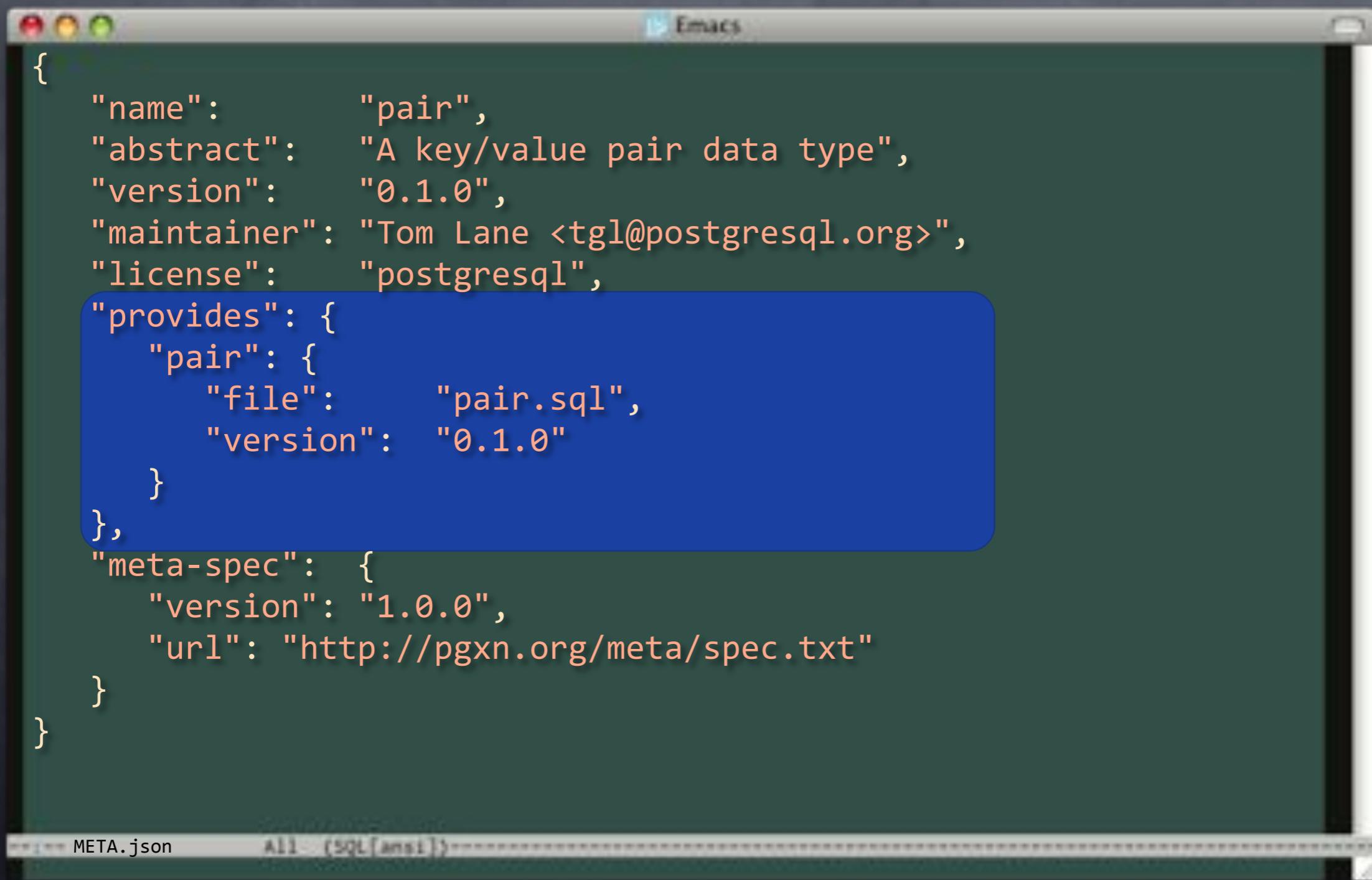
```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

# META.json



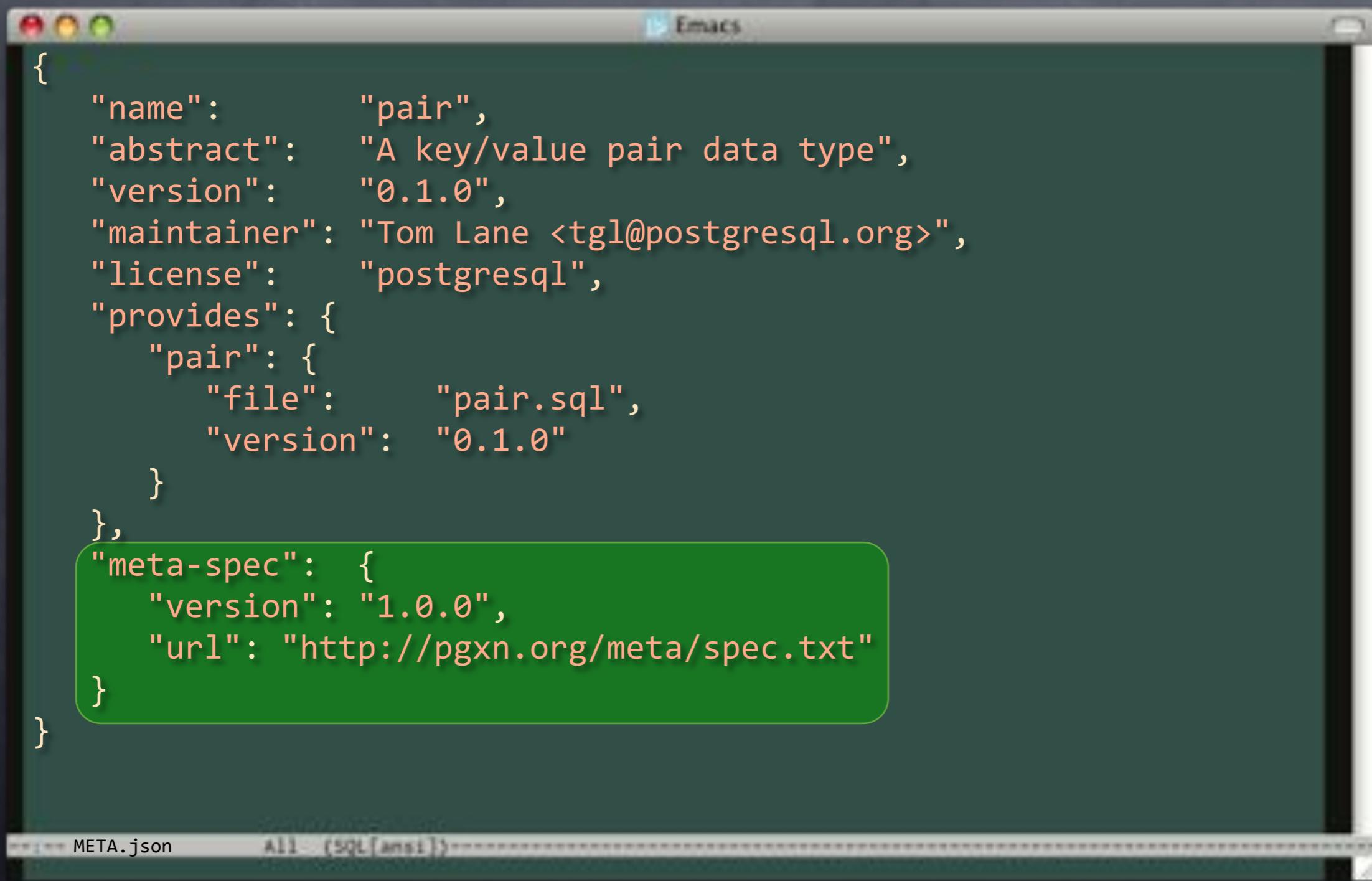
```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

# META.json



```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

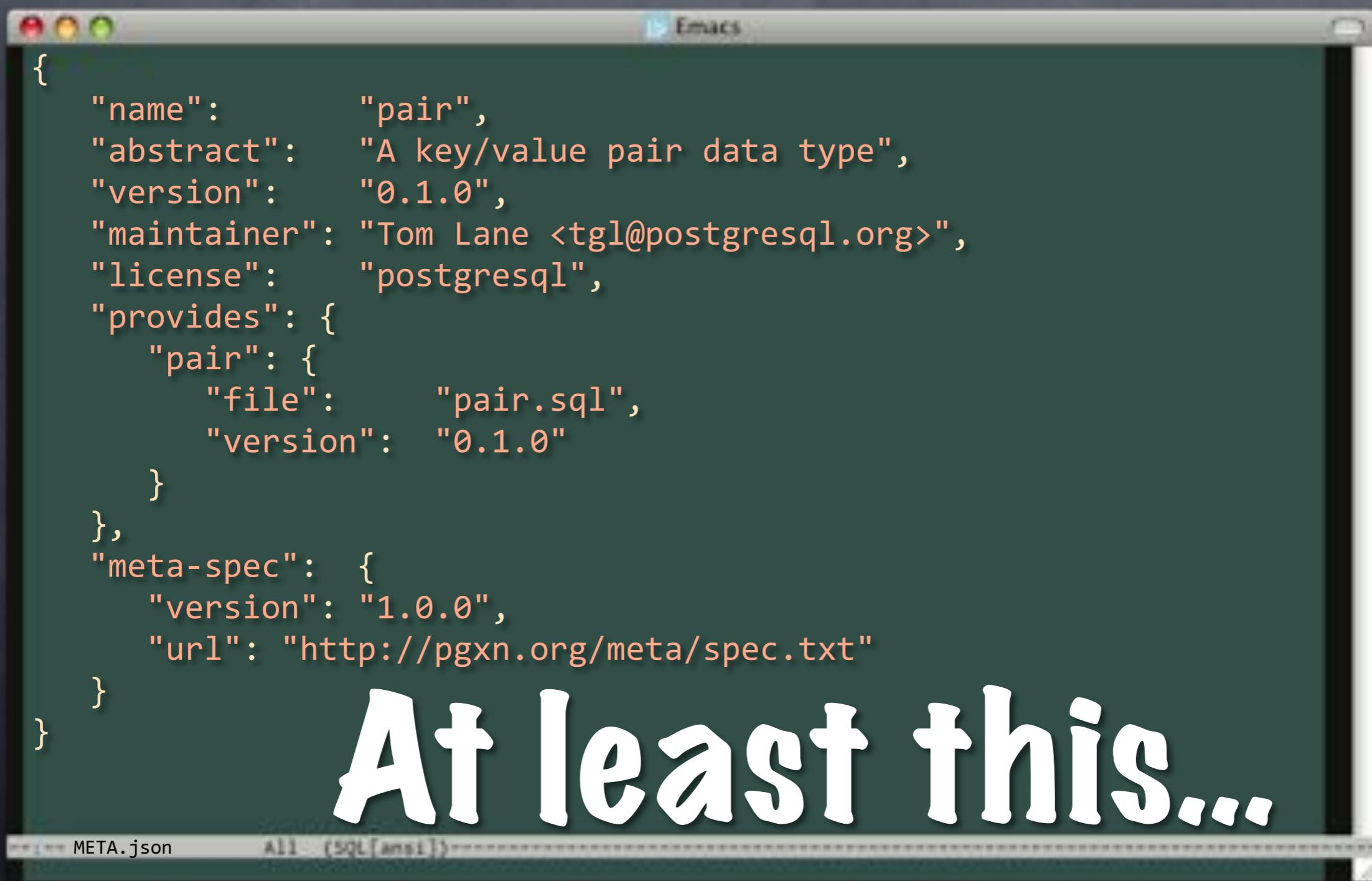
# META.json



```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

The screenshot shows an Emacs window with a dark background and light-colored text. The title bar says 'Emacs'. The buffer contains a JSON object representing a package named 'pair'. It includes fields for abstract, version, maintainer, license, provides, and meta-spec. The 'meta-spec' field is highlighted with a green rounded rectangle. The status bar at the bottom shows 'META.json' and 'All (SQL [ansi])'.

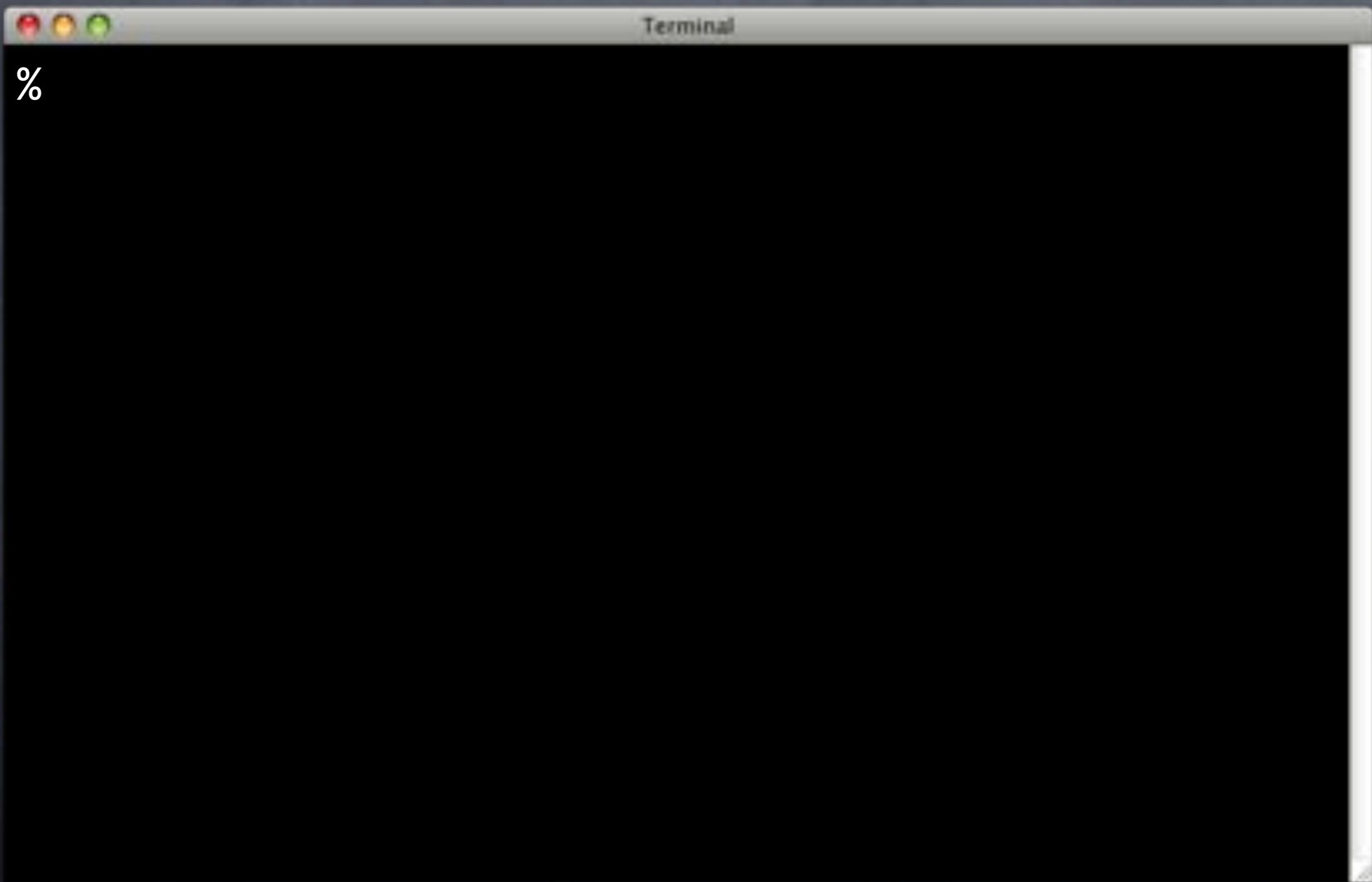
# META.json

A screenshot of the Emacs text editor showing a file named 'META.json'. The buffer contains JSON code defining a package named 'pair'. The code includes fields for name, abstract, version, maintainer, license, provides, and meta-spec. The 'provides' field contains a single entry for 'pair' with a file named 'pair.sql' and a version of '0.1.0'. The 'meta-spec' field specifies a version of '1.0.0' and a URL of 'http://pgxn.org/meta/spec.txt'.

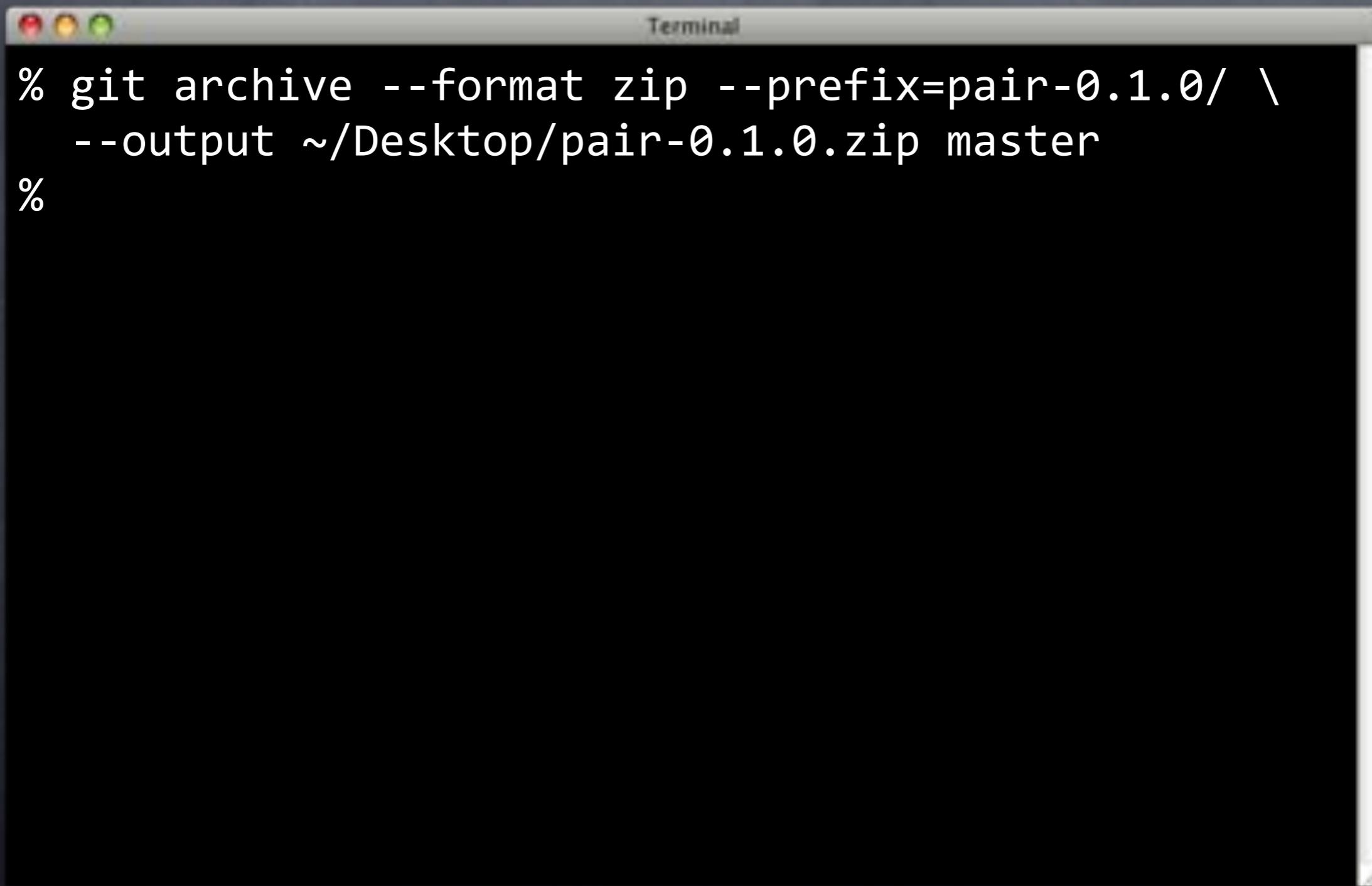
```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0"  
    }  
  },  
  "meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
  }  
}
```

At least this...

# Package it Up!



# Package it Up!



```
% git archive --format zip --prefix=pair-0.1.0/ \
--output ~/Desktop/pair-0.1.0.zip master
```

# Package it Up!



```
% git archive --format zip --prefix=pair-0.1.0/ \
--output ~/Desktop/pair-0.1.0.zip master
```

Easy, eh?



Perf+

References+

Personal+

Read Later

Bookmark

Google Docs

Atlas

Readability

TerraPass

bitly Sidebar

Kick ass

# Welcome

PGXN Manager is a Webapp that allows you to upload PostgreSQL extension distributions and have them be distributed to the PostgreSQL Extension Network. See "About" for details on how to get started.

PGXN Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.



## PGXN Manager

(Release it on PGXN!)

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)



# Welcome

PGXN Manager is a Webapp that allows you to upload PostgreSQL extension distributions and have them be distributed to the PostgreSQL Extension Network. See "About" for details on how to get started.

PGXN Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

## PGXN Manager

(Release it on PGXN!)

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)

# Request an Account

Want to distribute your PostgreSQL extensions on PGXN? Register here to request an account. We'll get it approved post haste.

## The Essentials

Name: Barack Obama

What does your mother call you?

Email: you@example.com

Where can we get hold of you?

URI: http://blog.example.com/

Got a blog or personal site?

Nickname: bobama

By what name would you like to be known? Letters, numbers, and dashes only, please.

Twitter: @barackobama

Got a Twitter account? Tell us the username and your uploads will be tweeted!

## Your Plans

Why: I would like to release the following killer extensions on PGXN:

- foo
- bar
- baz

So what are your plans for PGXN? What do you wanna release?

Pretty Please!



PGXN Manager  
Release it on PGXN!

Log In

Request Account (Mouse cursor)

Reset Password

About

How To

Contact

PGXN Manager — Request an account and start releasing distributions

http://manager.pgxn.org/account/register

DuckDuckGo

# Request an Account

Want to distribute your PostgreSQL extensions on PGXN? Register here to request an account. We'll get it approved post haste.

**The Essentials**

Name: Tom Lane  
What does your mother call you?

Email: tgl@postgresql.org  
Where can we get hold of you?

URI: <http://postgresql.org/~tgl/>  
Got a blog or personal site?

Nickname: tomlane  
By what name would you like to be known? Letters, numbers, and dashes only, please.

Twitter: @tomlane  
Got a Twitter account? Tell us the username and your uploads will be tweeted!

**Your Plans**

Why: I've got some killer extensions in development that I think will be useful to everyone, including:

- \* pair: an ordered pair data type
- \* PL/Brainfuck: just what it sounds like

So what are your plans for PGXN? What do you wanna release?

Pretty Please!



PGXN Manager  
Release it on PGXN!

Log In

Request Account

Reset Password

About

How To

Contact

PGXN Manager — Request an account and start releasing distributions

http://manager.pgxn.org/account/register

DuckDuckGo

# Request an Account

Want to distribute your PostgreSQL extensions on PGXN? Register here to request an account. We'll get it approved post haste.

**The Essentials**

Name: Tom Lane  
What does your mother call you?

Email: tgl@postgresql.org  
Where can we get hold of you?

URI: http://postgresql.org/~tgl/  
Got a blog or personal site?

Nickname: tomlane  
By what name would you like to be known? Letters, numbers, and dashes only, please.

Twitter: @tomlane  
Got a Twitter account? Tell us the username and your uploads will be tweeted!

**Your Plans**

Why: I've got some killer extensions in development that I think will be useful to everyone, including:

- \* pair: an ordered pair data type
- \* PL/Brainfuck: just what it sounds like

So what are your plans for PGXN? What do you wanna release?

Pretty Please!

[Perl](#)[References](#)[Personal](#)[Read Later](#)[Bookmark](#)[Google Docs](#)[Atlas](#)[Readability](#)[TerraPass](#)[Bitly Sidebar](#)[Kick ass](#)

# Thanks

Thanks for requesting a PGXN account, tomlane. We'll get back to you once the hangover has worn off.

PGXN Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

## PGXN Manager

[Release It! on PGXN](#)

[Log In](#)

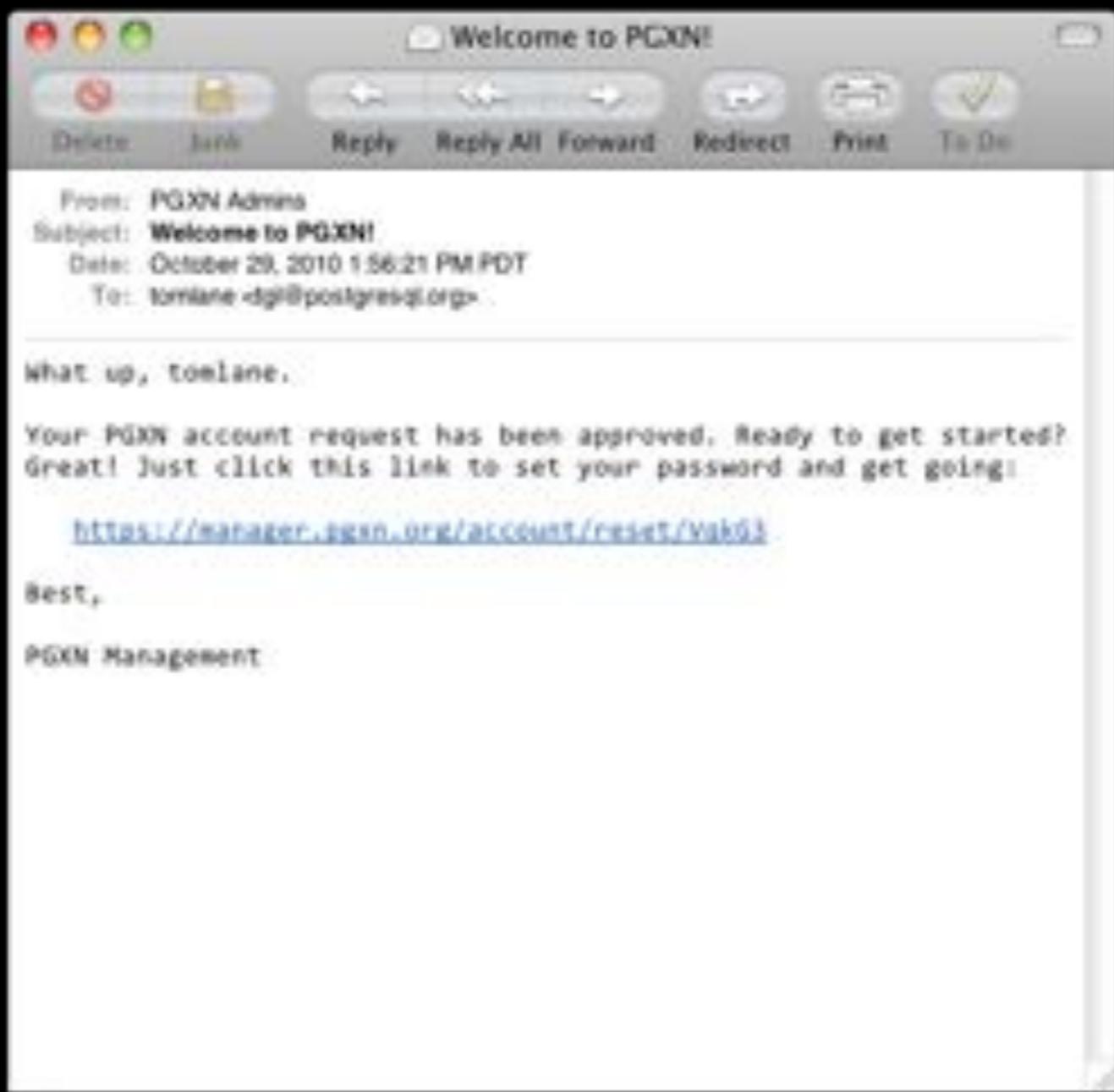
[Request Account](#)

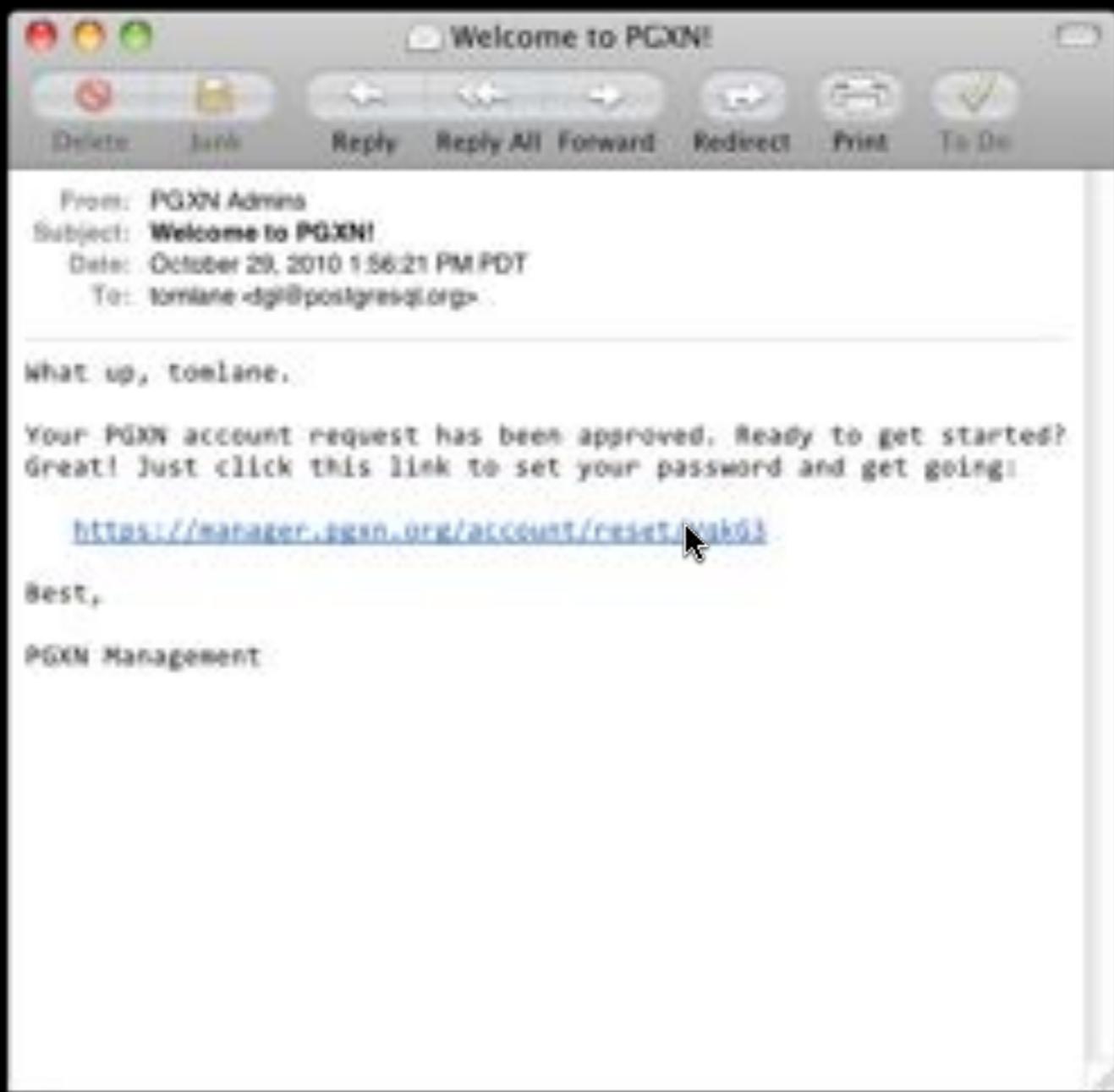
[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)





# Reset Your PGXN Password

Please choose a password to use for your PGXN account.

## Change Password

New Password:

Verify Password:

[Change](#)

PGXM Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

**PGXN Manager**  
Release it on PGHQ!

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)

# Reset Your PGXN Password

Please choose a password to use for your PGXN account.

## Change Password

New Password:

Verify Password:

[Change](#)

PGIM Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

**PGXN Manager**

Release it on PGHQ!

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)

# Reset Your PGXN Password

Please choose a password to use for your PGXN account.

## Change Password

New Password:

Verify Password:

[Change](#)

PGIM Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

**PGXN Manager**

Release it on PGHQ!

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)

[Perl+](#)[References+](#)[Personal+](#)[Read Later](#)[Bookmark](#)[Google Docs](#)[Atlas](#)[Readability](#)[TerraPass](#)[bitly Sidebar](#)[Kick ass](#)

# Password Changed



WOOH! Your password has been changed. So what are you waiting for? [Go log in!](#)

PGXN Manager v1.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

**PGXN Manager**

Release it on PGHQ!

[Log In](#)

[Request Account](#)

[Reset Password](#)

[About](#)

[How To](#)

[Contact](#)

[Perf+](#)[References+](#)[Personal+](#)[Read Later](#)[Bookmark](#)[Google Docs](#)[Atlas](#)[Readability](#)[TerraPass](#)[Bitly Sidebar](#)[Kick ass](#)

## PGXN Manager

Release it on PGXN!

[Log In](#)[Request Account](#)[Reset Password](#)[About](#)[How To](#)[Contact](#)

This site is asking you to login. Please provide your username and password.

Domain: manager.pgxn.org:443

Realm: PGXN Users Only

Username:

Password:

[Cancel](#)[Log In](#)

Are you waiting for? Go log in!

Perl+

References+

Personal+

Read Later

Bookmark

Google Docs

Atlas

Readability

TerraPass

bitly Sidebar

Kick ass



## PGXN Manager

Release It on PGXN!

[Log In](#)[Request Account](#)[Reset Password](#)[About](#)[How To](#)[Contact](#)

This site is asking you to login. Please provide your username and password.

Domain: manager.pgxn.org:443

Realm: PGXN Users Only

Username: Password: [Cancel](#)[Log In](#)

Are you waiting for? Go log in!



PGXN Manager — Distribute PostgreSQL extensions on our world-wide network

https://manager.pgxn.org/ DuckDuckGo

Perl+ References+ Personal+ Read Later Bookmark Google Docs Atlas Readability ToraPass Bitly Sidebar Kick ass



# Welcome

PGXN Manager is a Webapp that allows you to upload PostgreSQL extension distributions and have them be distributed to the PostgreSQL Extension Network. See "About" for details on how to get started.

PGXN Manager v0.4.4 · © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

Upload a Distribution  
Your Distributions  
Show Permissions  
Edit Account  
Change Password  
  
About  
How To  
Contact

PGXN Manager — Distribute PostgreSQL extensions on our world-wide network

https://manager.pgxn.org/ DuckDuckGo

Perl+ References+ Personal+ Read Later Bookmark Google Docs Atlas Readability ToraPass Bitly Sidebar Kick ass



# Welcome

PGXN Manager is a Webapp that allows you to upload PostgreSQL extension distributions and have them be distributed to the PostgreSQL Extension Network. See "About" for details on how to get started.

PGXN Manager v0.4.4 · © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

PGXN Manager · Release it on PGHQ!

Upload a Distribution

Your Distributions

Show Permissions

Edit Account

Change Password

About

How To

Contact

[Perf+](#)[References+](#)[Personal+](#)[Read Later](#) [Bookmark](#) [Google Docs](#) [Atlas](#) [Readability](#) [TerraPass](#) [Bitly Sidebar](#) [Kick ass](#)

# Upload a Distribution

So you've developed a PGXN extension and what to distribute it on PGXN. This is the place to upload it! Just find your distribution archive (.zip, .tgz, etc..) in the upload field below and you'll be good to go.

Don't know what this means? Want to know how to create great PostgreSQL extensions and distribute them to your fellow PostgreSQL enthusiasts via PGXN? Take a gander at our [How to](#) for all the juicy details. It's not hard, we promise.

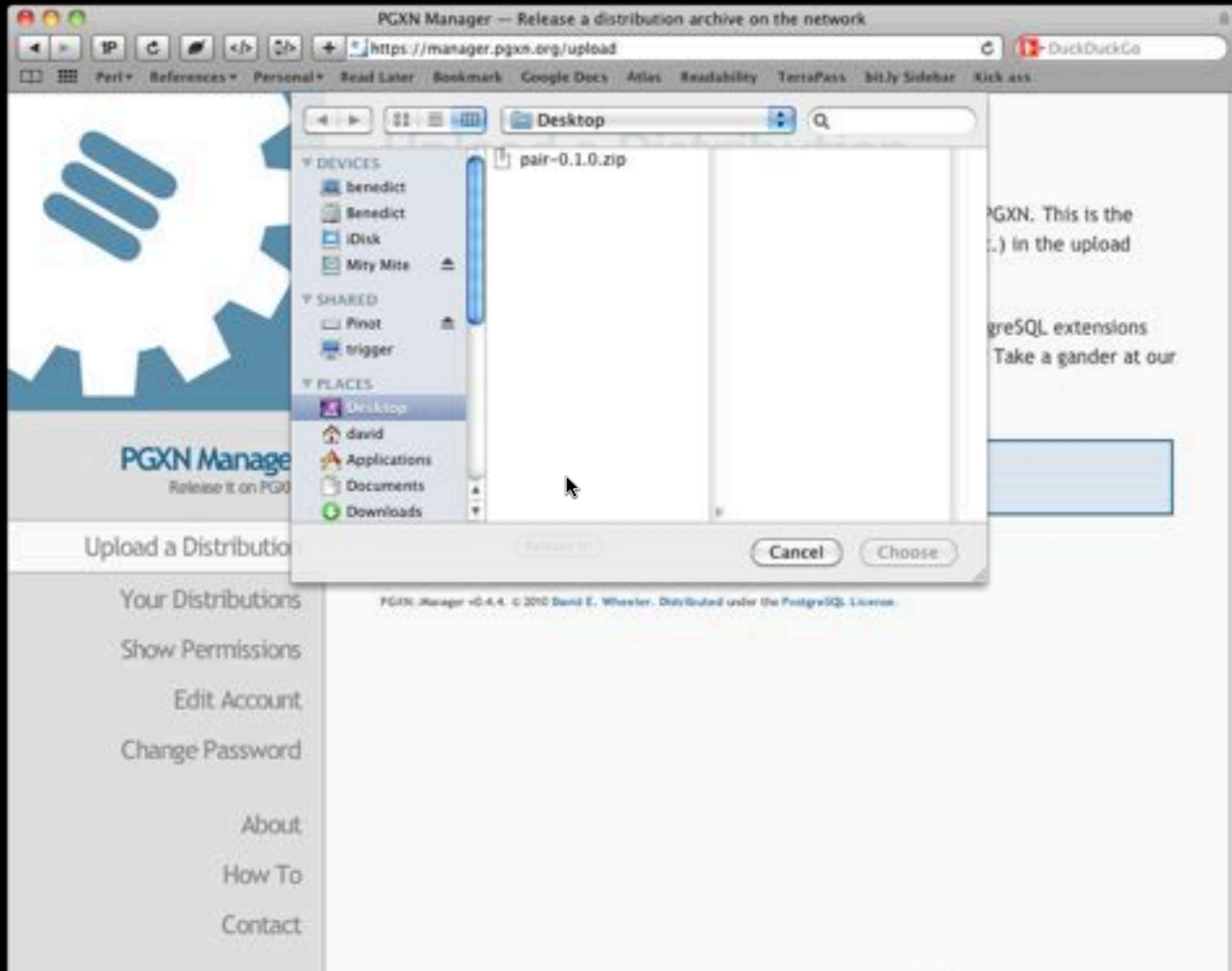
## Upload a Distribution Archive

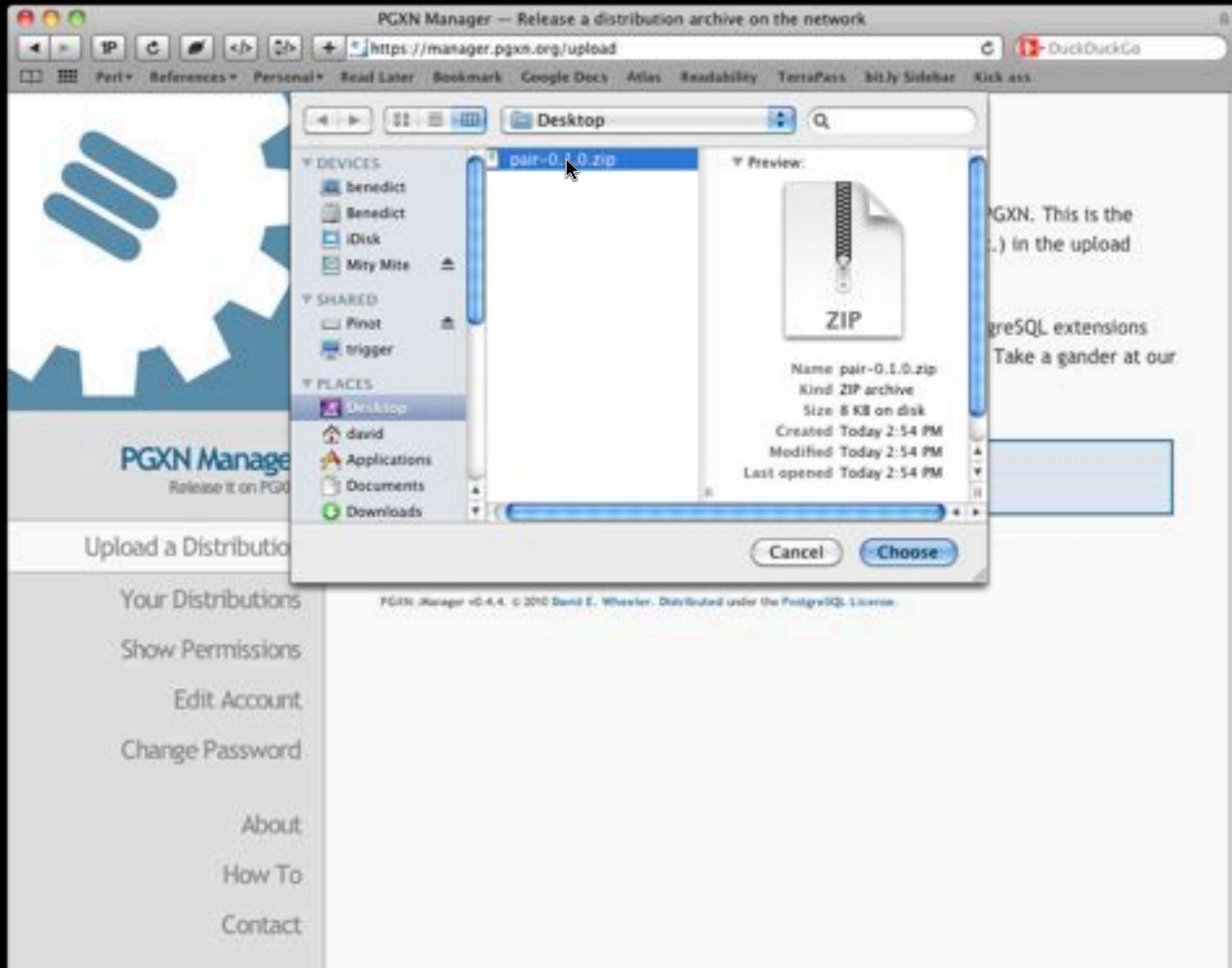
Archive:  no file selected

PGXN Manager v0.4.4 - © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

[Privacy Policy](#) | [Terms of Service](#) | [Feedback](#) | [Help](#) | [Log Out](#)

[Upload a Distribution](#)[Your Distributions](#)[Show Permissions](#)[Edit Account](#)[Change Password](#)[About](#)[How To](#)[Contact](#)





[Perf+](#)[References+](#)[Personal+](#)[Read Later](#)[Bookmark](#)[Google Docs](#)[Atlas](#)[Readability](#)[TerraPass](#)[Bitly Sidebar](#)[Kick ass](#)

# Upload a Distribution

So you've developed a PGXN extension and what to distribute it on PGXN. This is the place to upload it! Just find your distribution archive (.zip, .tgz, etc..) in the upload field below and you'll be good to go.

Don't know what this means? Want to know how to create great PostgreSQL extensions and distribute them to your fellow PostgreSQL enthusiasts via PGXN? Take a gander at our [How to](#) for all the juicy details. It's not hard, we promise.

## Upload a Distribution Archive

Archive:  pair-0.1.0.zip

PGXN Manager v0.4.4 - © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

[Upload a Distribution](#)[Your Distributions](#)[Show Permissions](#)[Edit Account](#)[Change Password](#)[About](#)[How To](#)[Contact](#)

[Perf+](#)[References+](#)[Personal+](#)[Read Later](#) [Bookmark](#) [Google Docs](#) [Atlas](#) [Readability](#) [TerraPass](#) [Bitly Sidebar](#) [Kick ass](#)

# Upload a Distribution

So you've developed a PGXN extension and what to distribute it on PGXN. This is the place to upload it! Just find your distribution archive (.zip, .tgz, etc..) in the upload field below and you'll be good to go.

Don't know what this means? Want to know how to create great PostgreSQL extensions and distribute them to your fellow PostgreSQL enthusiasts via PGXN? Take a gander at our [How to](#) for all the juicy details. It's not hard, we promise.

## Upload a Distribution Archive

Archive:  pair-0.1.0.zip

[Release It!](#)

PGXN Manager v0.4.4 - © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

[Privacy Policy](#) | [Terms of Service](#) | [Feedback](#)

[Upload a Distribution](#)

[Your Distributions](#)

[Show Permissions](#)

[Edit Account](#)

[Change Password](#)

[About](#)

[How To](#)

[Contact](#)

PGON Manager — pair-0.1.0

https://manager.pgxn.org/distributions/pair/0.1.0

DuckDuckGo

PGON Manager — pair-0.1.0

A key/value pair data type

Congratulations! This distribution has been released on PGXN.

Owner: tomlane

Status: stable

SHA1: 1a6aec4af6db420ed6c2e3fb338c15cbf5b8689

Extensions: • pair 0.1.0

PGXN Manager v0.4.4, © 2010 David E. Wheeler. Distributed under the PostgreSQL License.

Archive

README

Metadata

Upload a Distribution

Your Distributions

Show Permissions

Edit Account

Change Password

About

How To

Contact





# pair

This Release: pair 0.1.0

Date: 2011-04-21

Status: Stable

Abstract: A key/value pair data type

Released By: tomlane

License: The PostgreSQL License

Special Files: [Makefile](#) • [README.pair](#) • [META.json](#)

## Extensions

pair 0.1.0

## README

pair 0.1.0

-----

This library contains a single PostgreSQL extension, a key/value pair data type called 'pair', along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named 'k' and 'v'.

The 'pair' data type was created as an inspiration, as documented in [\(this blog post\)](#)(<http://justatheory.com/computers/databases/postgresql/key-value-pairs.html>). Give it a read if you're interested in the context of its creation.

To build it, just do this:

```
make  
make installcheck  
make install
```

If you encounter an error such as:

# pair

This Release: pair 0.1.0

Date: 2011-04-21

Status: Stable

Abstract: A key/value pair data type

Released By: tomlane

License: The PostgreSQL License

Special Files: [Makefile](#) • [README.pair](#) • [META.json](#)

No resources,  
tags, or long  
description

## Extensions

### pair 0.1.0

## README

### pair 0.1.0

XXXXXXXXXX

This library contains a single PostgreSQL extension, a key/value pair data type called 'pair', along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named 'k' and 'v'.

The 'pair' data type was created as an inspiration, as documented in [\(this blog post\)](#)(<http://justatheory.com/computers/databases/postgresql/key-value-pairs.html>). Give it a read if you're interested in the context of its creation.

To build it, just do this:

```
make
make installcheck
make install
```

If you encounter an error such as:



# pair

This Release: pair 0.1.0

Date: 2011-04-21

Status: Stable

Abstract: A key/value pair data type

Released By: tomlane

License: The PostgreSQL License

Special Files: [Makefile](#) • [README.pair](#) • [META.json](#)

## Extensions

pair 0.1.0

# No documentation link

## README

pair 0.1.0

.....

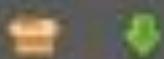
This library contains a single PostgreSQL extension, a key/value pair data type called 'pair', along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named 'k' and 'v'.

The 'pair' data type was created as an inspiration, as documented in [\(this blog post\)](#)(<http://justatheory.com/computers/databases/postgresql/key-value-pairs.html>). Give it a read if you're interested in the context of its creation.

To build it, just do this:

```
make  
make installcheck  
make install
```

If you encounter an error such as:



# pair

This Release: pair 0.1.0

Date: 2011-04-21

Status: Stable

Abstract: A key/value pair data type

Released By: tomlane

License: The PostgreSQL License

Special Files: [Makefile](#) • [README.pair](#) • [META.json](#)

## Extensions

pair 0.1.0

[README](#)

pair 0.1.0

XXXXXXXXXX

This library contains a single PostgreSQL extension, a key/value pair data type called 'pair', along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named 'k' and 'v'.

The 'pair' data type was created as an inspiration, as documented in [\(this blog post\)](#)(<http://justatheory.com/computers/databases/postgresql/key-value-pairs.html>). Give it a read if you're interested in the context of its creation.

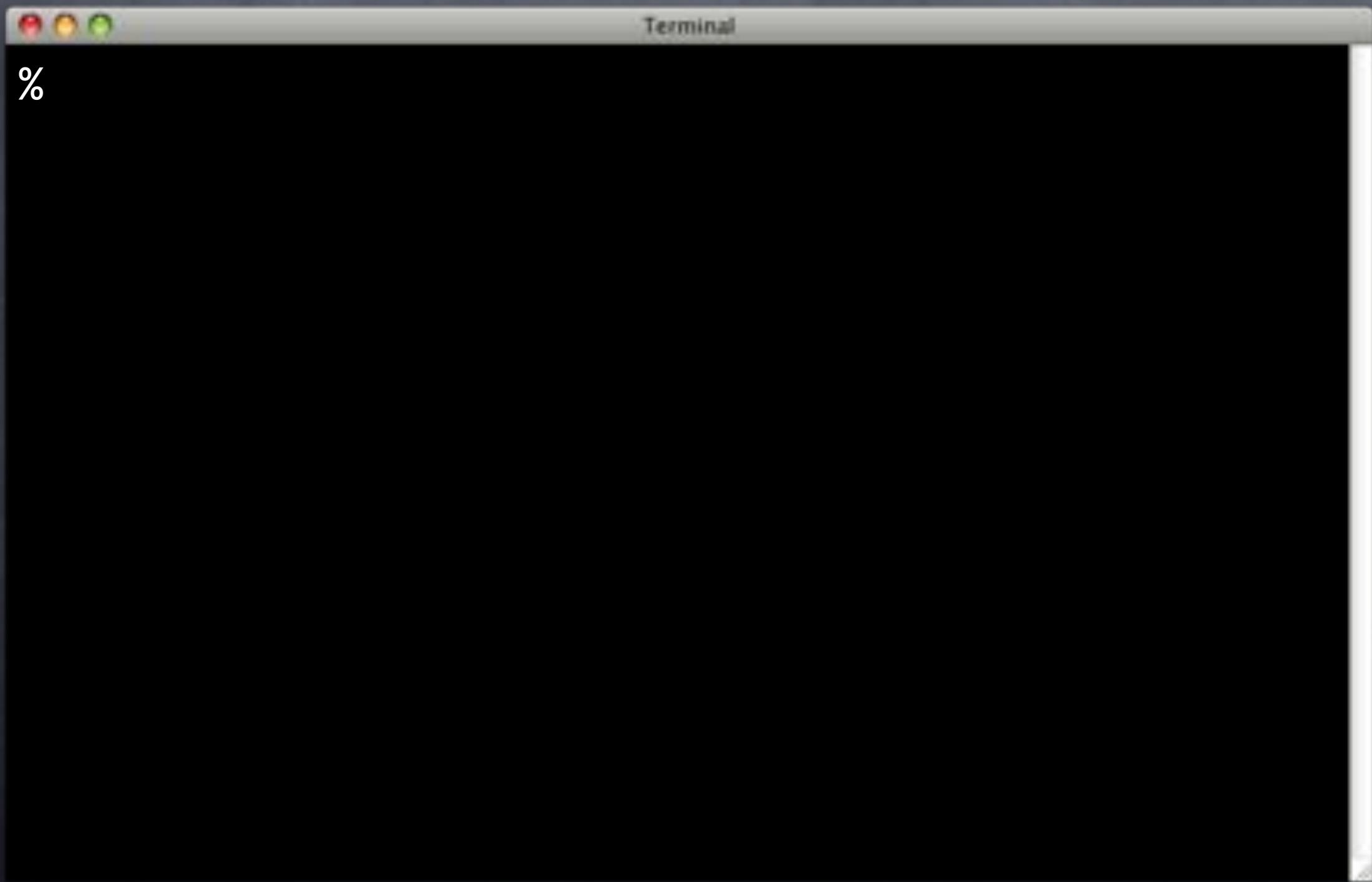
To build it, just do this:

```
make  
make installcheck  
make install
```

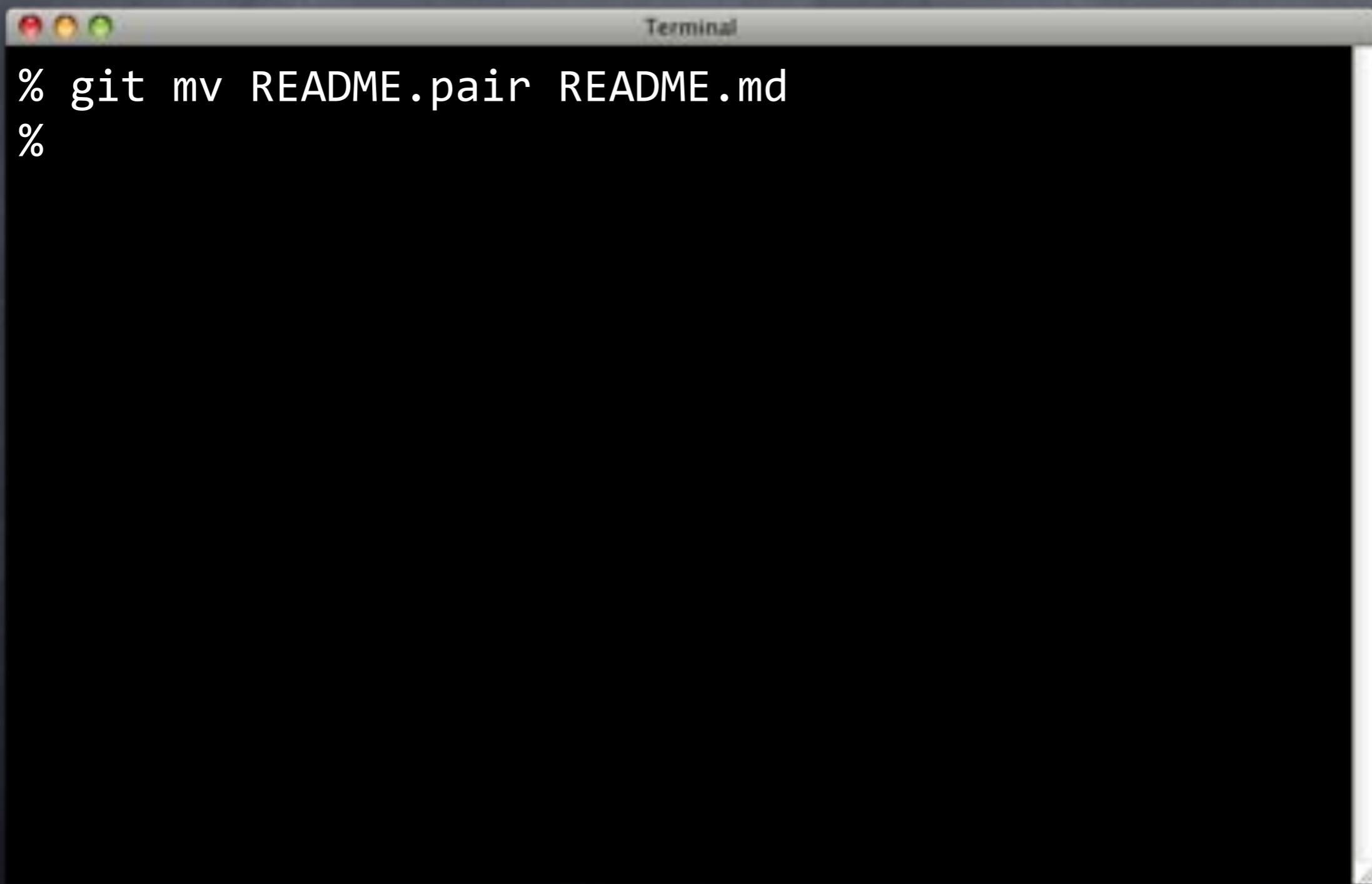
If you encounter an error such as:

Plain text  
**README**

# Add README Extension



# Add README Extension



```
Terminal  
% git mv README.pair README.md  
%
```

# Add README Extension



```
% git mv README.pair README.md  
%
```

Easy, eh?

# PGXN Markup

# PGXN Markup

- HTML
- Markdown
- MultiMarkdown
- Pod
- Textile
- Trac
- MediaWiki

# PGXN Markup

- HTML
- Markdown
- MultiMarkdown
- Pod
- Textile
- Trac
- MediaWiki
- Text::Markup (fork me!)

# PGXN Markup

- HTML
- Markdown
- MultiMarkdown
- Pod
- Text::Markup (fork me!)
- Write some Docs!
- Textile
- Trac
- MediaWiki

# Add Documentation

# Add Documentation

- Use any supported markup

# Add Documentation

- Use any supported markup
- Put wherever you like

# Add Documentation

- Use any supported markup
- Put wherever you like
- Recommend doc/



pair 0.1.1

=====

## Synopsis

-----

```
% CREATE EXTENSION pair;  
CREATE EXTENSION
```

```
% SELECT 'foo' ~> 'bar';  
pair
```

-----

```
(foo,bar)
```

## Description

-----

This library contains a single PostgreSQL extension, a key/value pair data type called “pair”, along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named `k` and `v`.

pair 0.1.1

=====

Synopsis

-----

```
% CREATE EXTENSION pair;  
CREATE EXTENSION
```

```
% SELECT 'foo' ~> 'bar';  
pair
```

-----

```
(foo,bar)
```

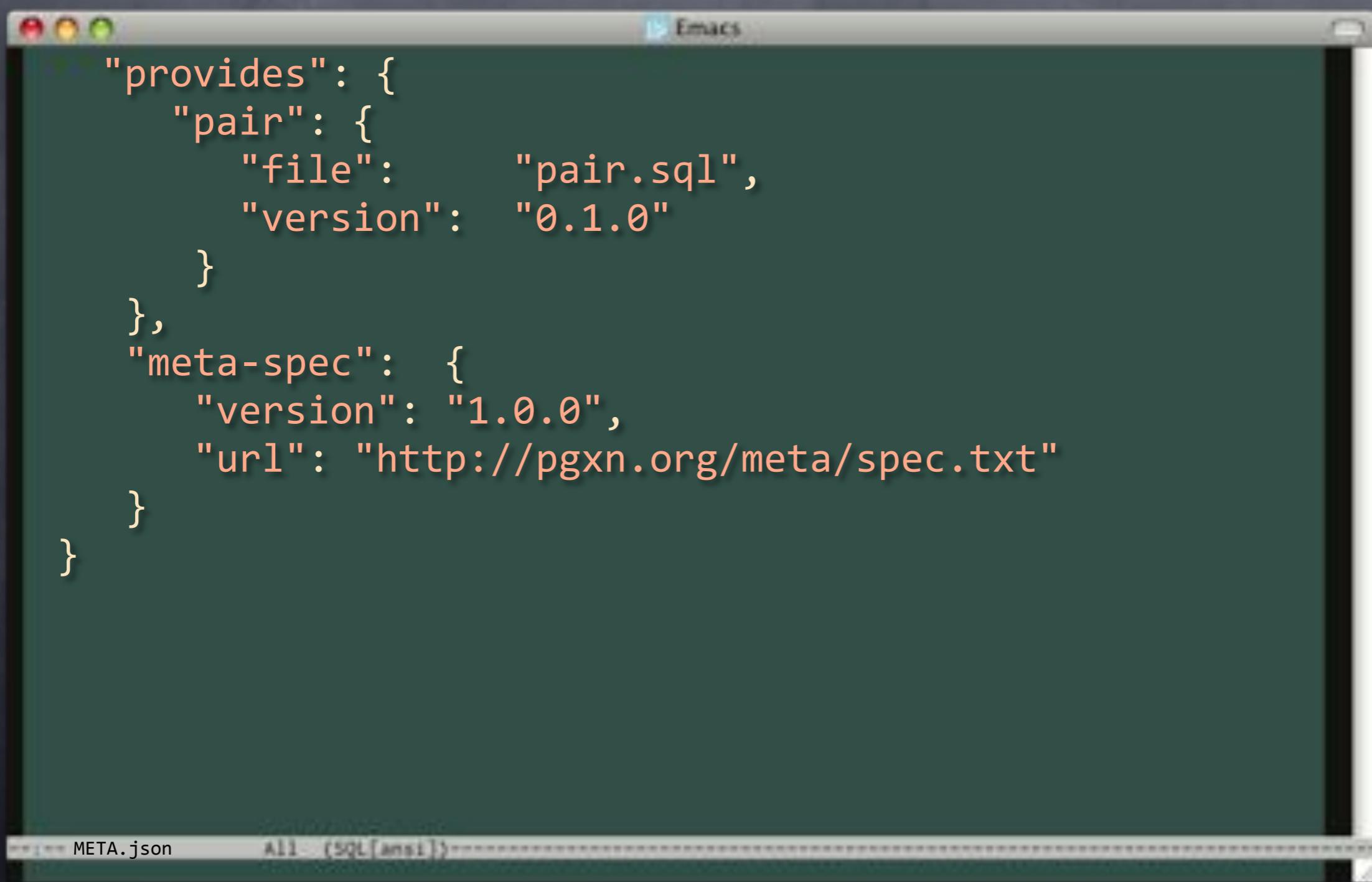
Description

-----

This library contains a single PostgreSQL extension, a key/value pair data type called “pair”, along with a convenience function for constructing key/value pairs. It’s just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named `k` and `v`.

Must be UTF-8  
or use a  
BOM

# Improve Provides

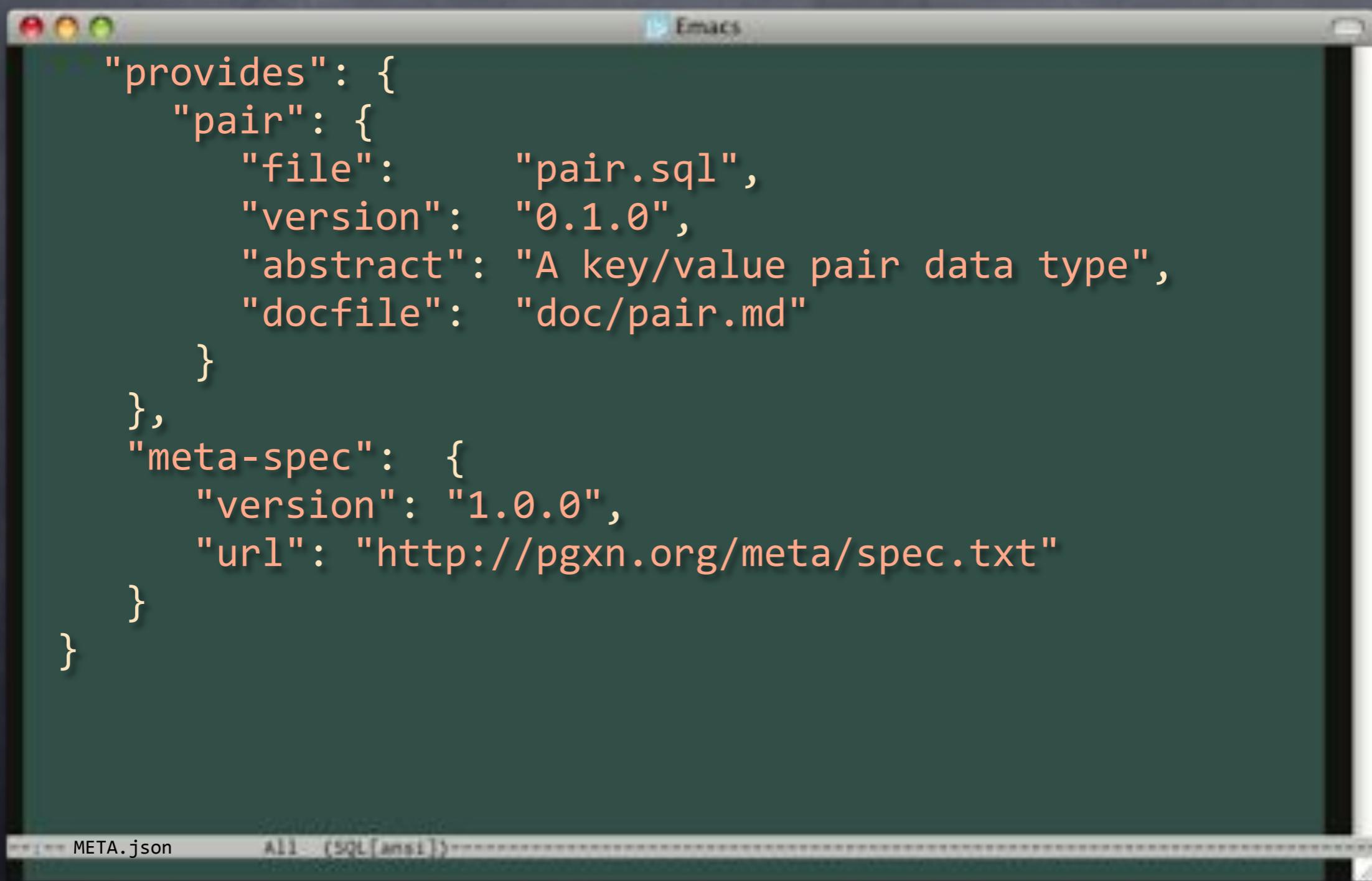


A screenshot of the Emacs text editor showing a JSON configuration file named `META.json`. The file contains the following code:

```
"provides": {  
    "pair": {  
        "file": "pair.sql",  
        "version": "0.1.0"  
    }  
},  
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
}  
}
```

The Emacs window title bar says "Emacs". The status bar at the bottom shows "META.json" and "All- (SQL [ansi])".

# Improve Provides

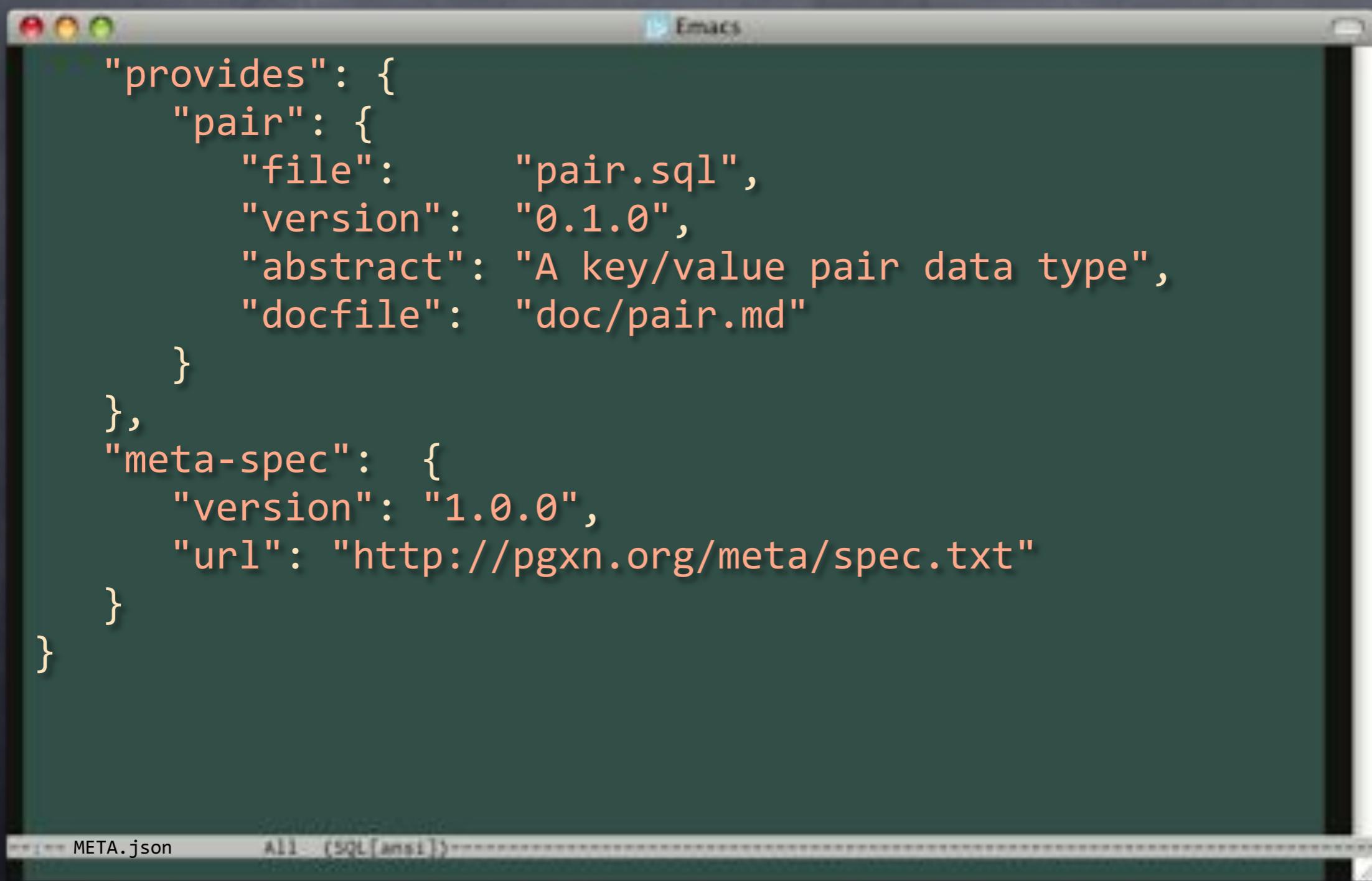


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following JSON code:

```
"provides": {  
    "pair": {  
        "file": "pair.sql",  
        "version": "0.1.0",  
        "abstract": "A key/value pair data type",  
        "docfile": "doc/pair.md"  
    }  
},  
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
}  
}
```

The status bar at the bottom left shows "META.json" and "All (SQL [ansi])".

# Add Description

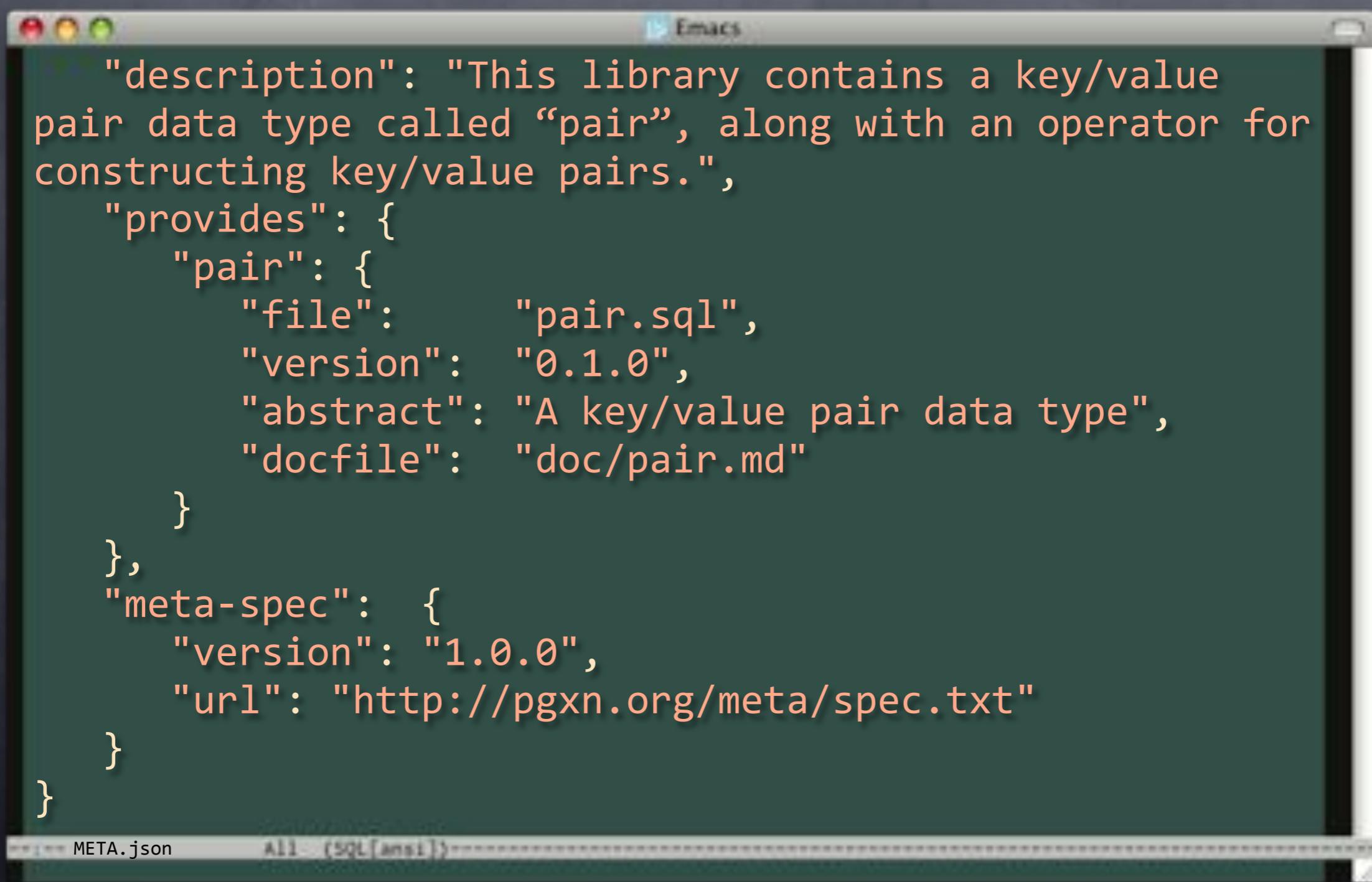


The screenshot shows an Emacs window with a dark green background and white text. The title bar says "Emacs". The buffer contains the following JSON code:

```
"provides": {  
    "pair": {  
        "file": "pair.sql",  
        "version": "0.1.0",  
        "abstract": "A key/value pair data type",  
        "docfile": "doc/pair.md"  
    }  
},  
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
}  
}
```

The status bar at the bottom left shows "META.json" and "All- (SQL [ansi])".

# Add Description

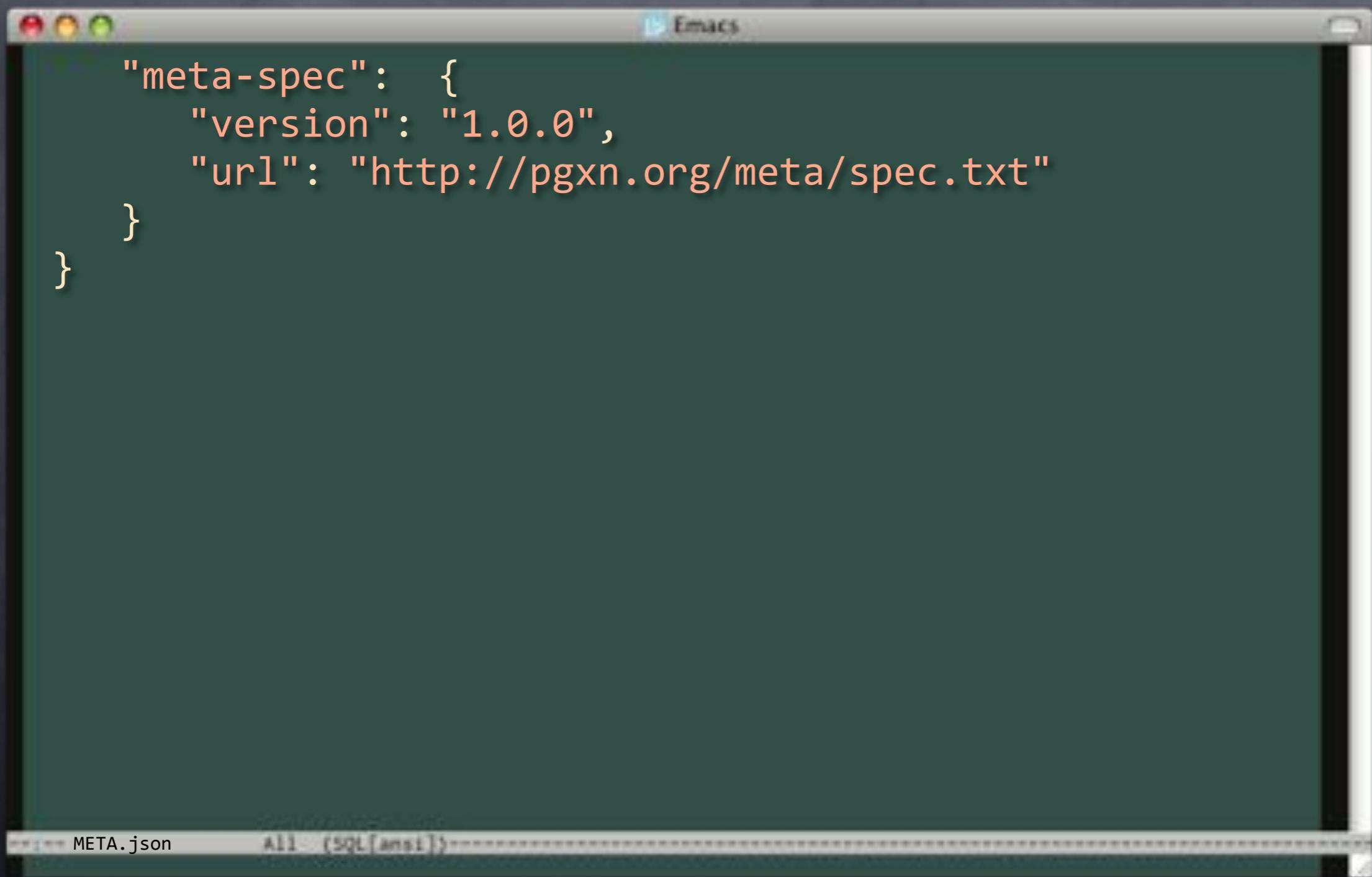


The screenshot shows a window titled "Emacs" displaying a JSON configuration file named "META.json". The file contains the following data:

```
"description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
"provides": {  
    "pair": {  
        "file": "pair.sql",  
        "version": "0.1.0",  
        "abstract": "A key/value pair data type",  
        "docfile": "doc/pair.md"  
    }  
},  
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
}  
}
```

The status bar at the bottom of the Emacs window shows "META.json" and "All (SQL [ansi])".

# Add Tags



The screenshot shows an Emacs window with a dark green background and white text. The title bar says "Emacs". The buffer contains the following JSON code:

```
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
}  
}
```

The status bar at the bottom left shows "META.json" and "All- (SQL [ansi])".

# Add Tags

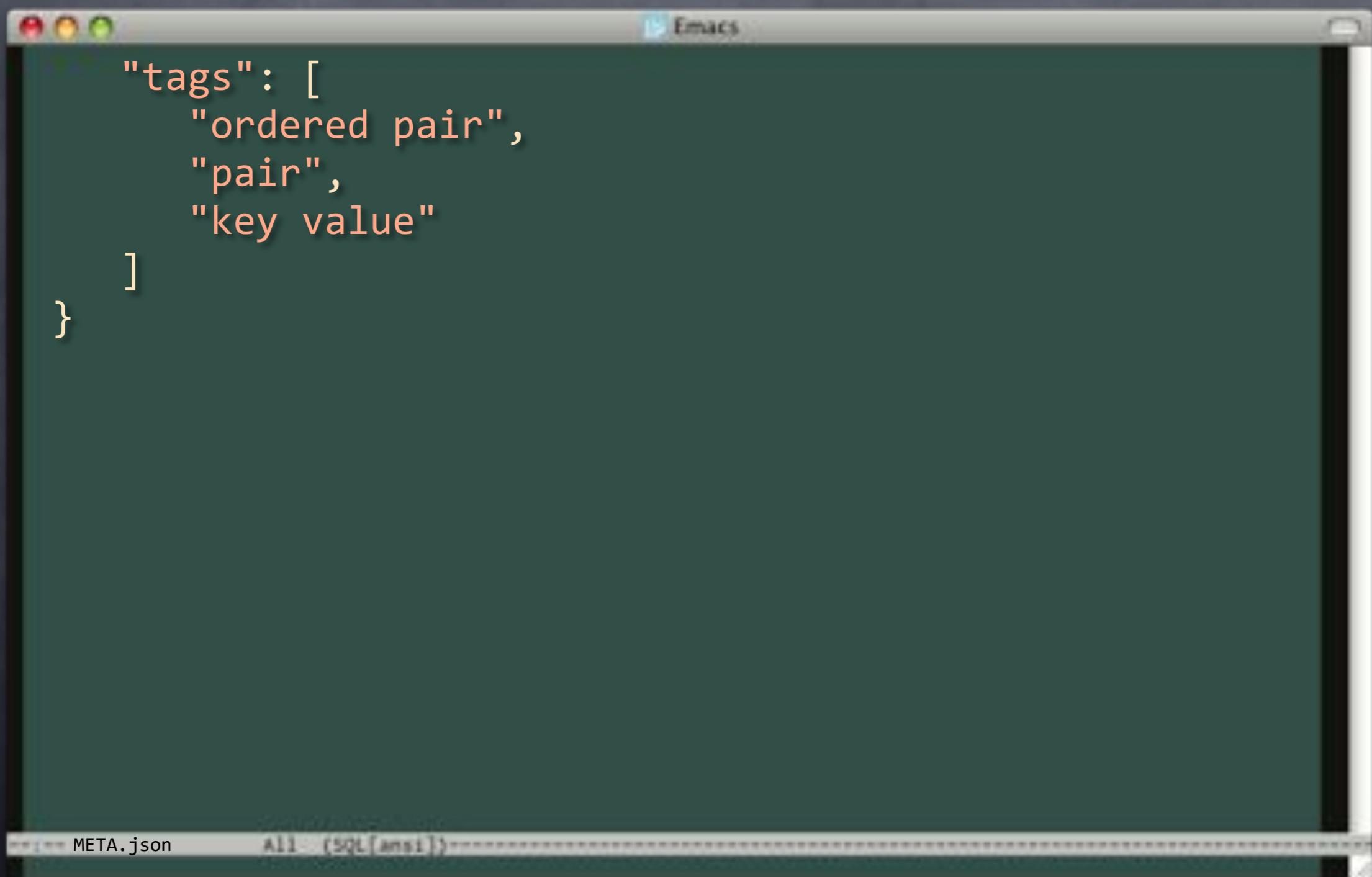


The screenshot shows an Emacs window with a dark green background and white text. The title bar says "Emacs". The buffer contains the following JSON code:

```
"meta-spec": {  
    "version": "1.0.0",  
    "url": "http://pgxn.org/meta/spec.txt"  
},  
"tags": [  
    "ordered pair",  
    "pair",  
    "key value"  
]  
}
```

The file name "META.json" is visible at the bottom left, and the mode line at the bottom right shows "All- (SQL [ansi])".

# Add Resources

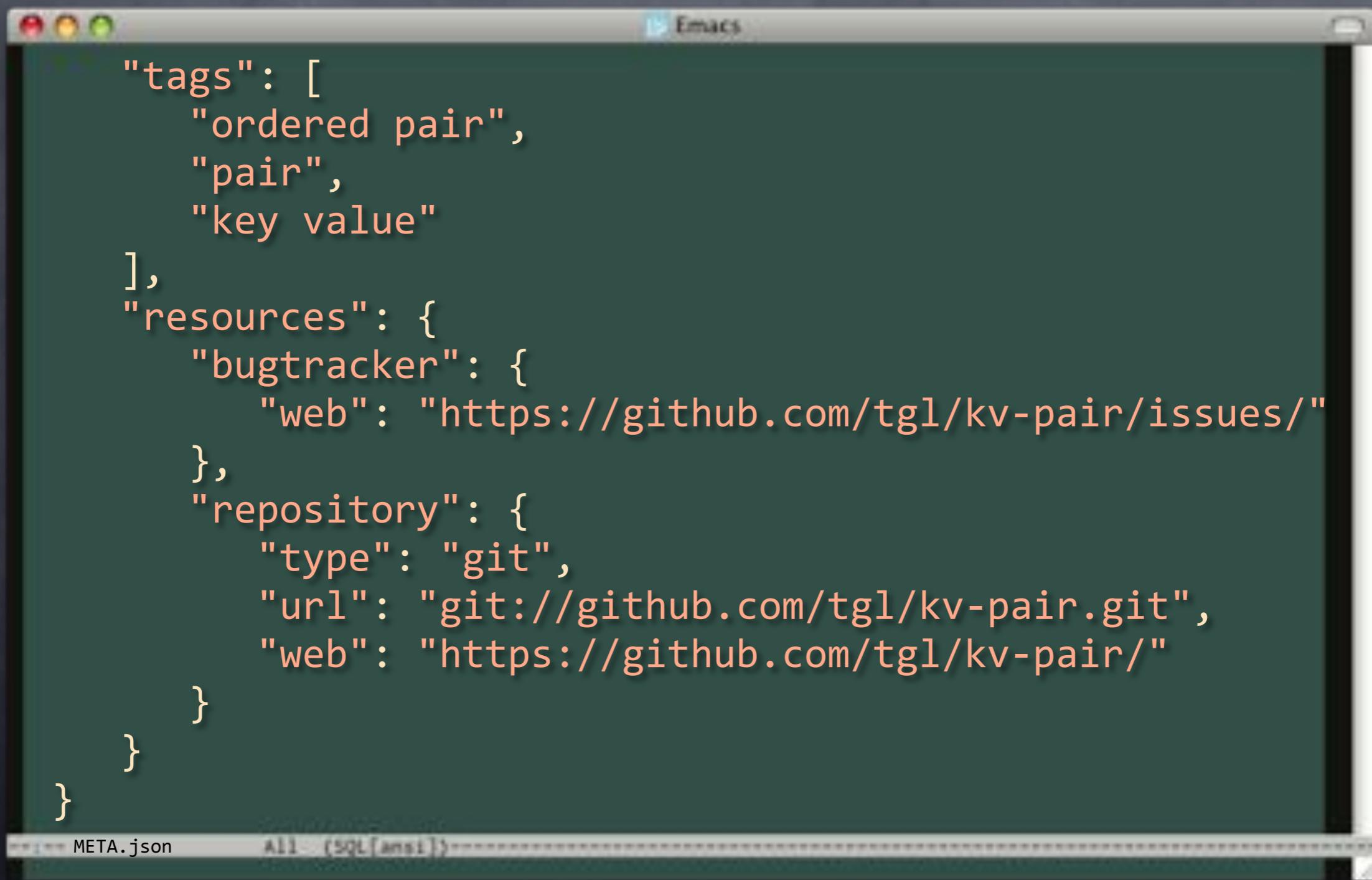


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following JSON code:

```
"tags": [  
    "ordered pair",  
    "pair",  
    "key value"  
]  
}
```

The file name "META.json" is visible at the bottom left, and the mode "All- (SQL [ansi])" is shown at the bottom right.

# Add Resources

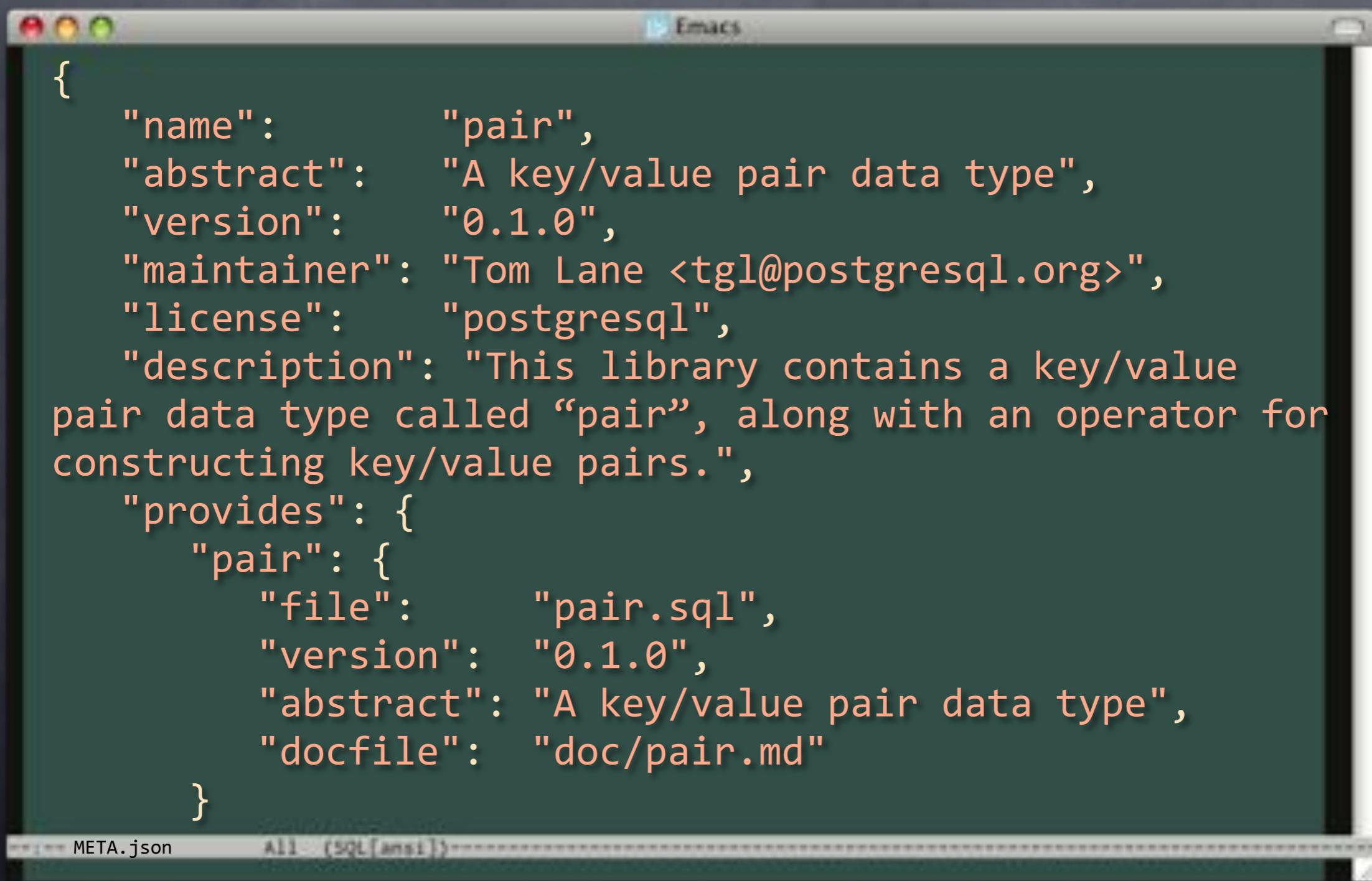


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following JSON configuration code:

```
"tags": [
    "ordered pair",
    "pair",
    "key value"
],
"resources": {
    "bugtracker": {
        "web": "https://github.com/tgl/kv-pair/issues/"
    },
    "repository": {
        "type": "git",
        "url": "git://github.com/tgl/kv-pair.git",
        "web": "https://github.com/tgl/kv-pair/"
    }
}
```

The file is named "META.json" as indicated at the bottom of the window.

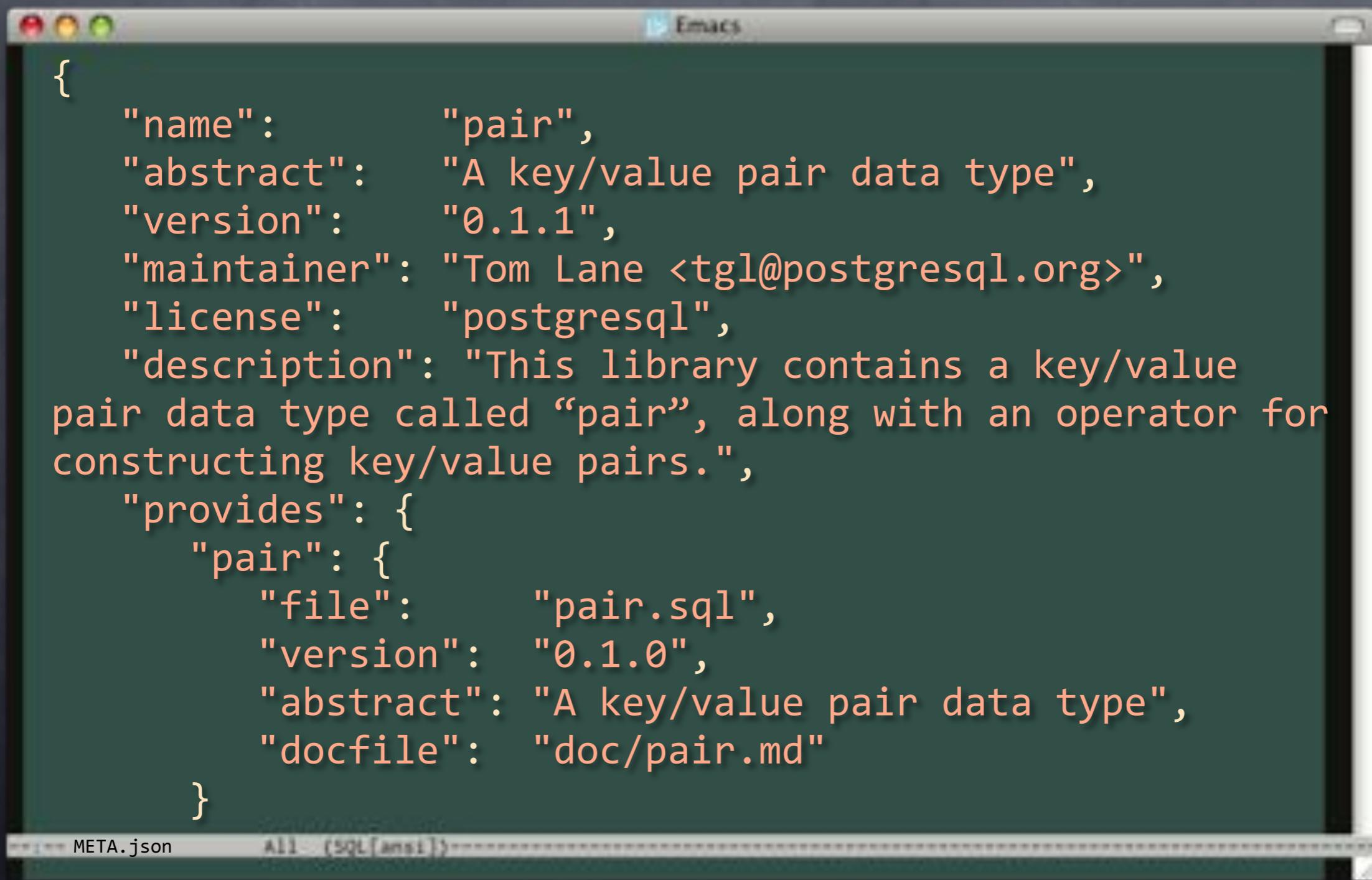
# Update Version



A screenshot of the Emacs text editor showing a JSON configuration file named `META.json`. The file contains metadata for a "pair" data type library. The JSON structure includes fields for name, abstract, version, maintainer, license, description, provides, and pair. The "description" field provides a detailed explanation of the library's purpose. The "provides" field indicates that the "pair" provider includes a `pair.sql` file, version 0.1.0, and an abstract description of the key/value pair data type.

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.0",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

# Update Version



A screenshot of the Emacs text editor showing a JSON configuration file named `META.json`. The file contains metadata for a "pair" data type library. The JSON structure includes fields for name, abstract, version, maintainer, license, description, provides, and pair. The "pair" provider entry specifies a file named `pair.sql`, version `0.1.0`, and an abstract description of a key/value pair data type.

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.1",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

# Update Version



```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "version": "0.1.1",  
  "maintainer": "Tom Lane <tcl@sss.pw.usf.edu>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

META.json

# Update Version

A screenshot of an Emacs window displaying a JSON configuration file named `META.json`. The file contains the following data:

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "version": "0.1.1",  
  "maintainer": "Tom Lane <tcl@sss.pw.usna.edu>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

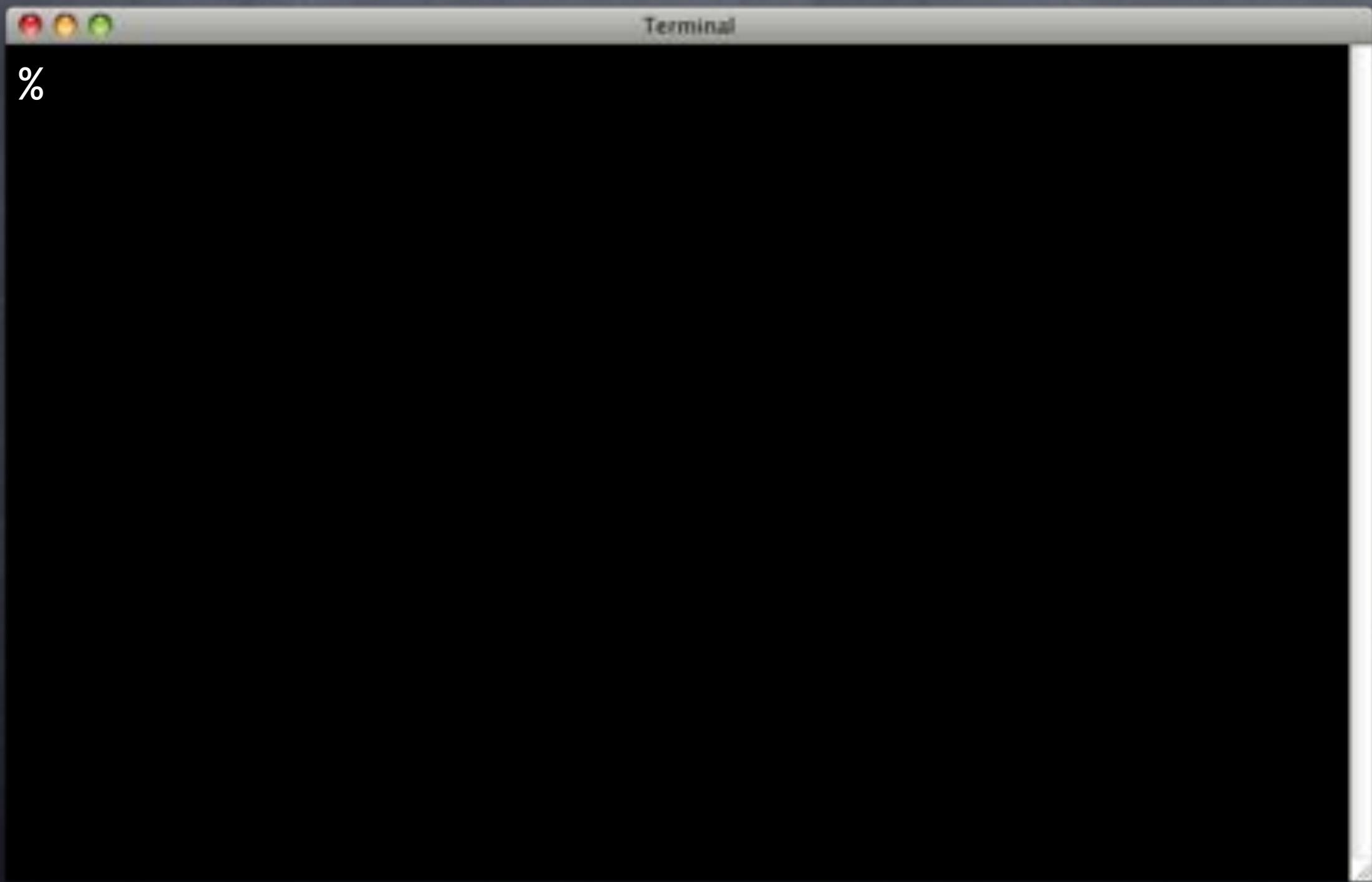
Two blue speech bubbles with white text are overlaid on the JSON data:

- A bubble pointing to the `version` field in the top object says **Must be unique.**
- A bubble pointing to the `version` field in the `pair` object says **Unchanged!**

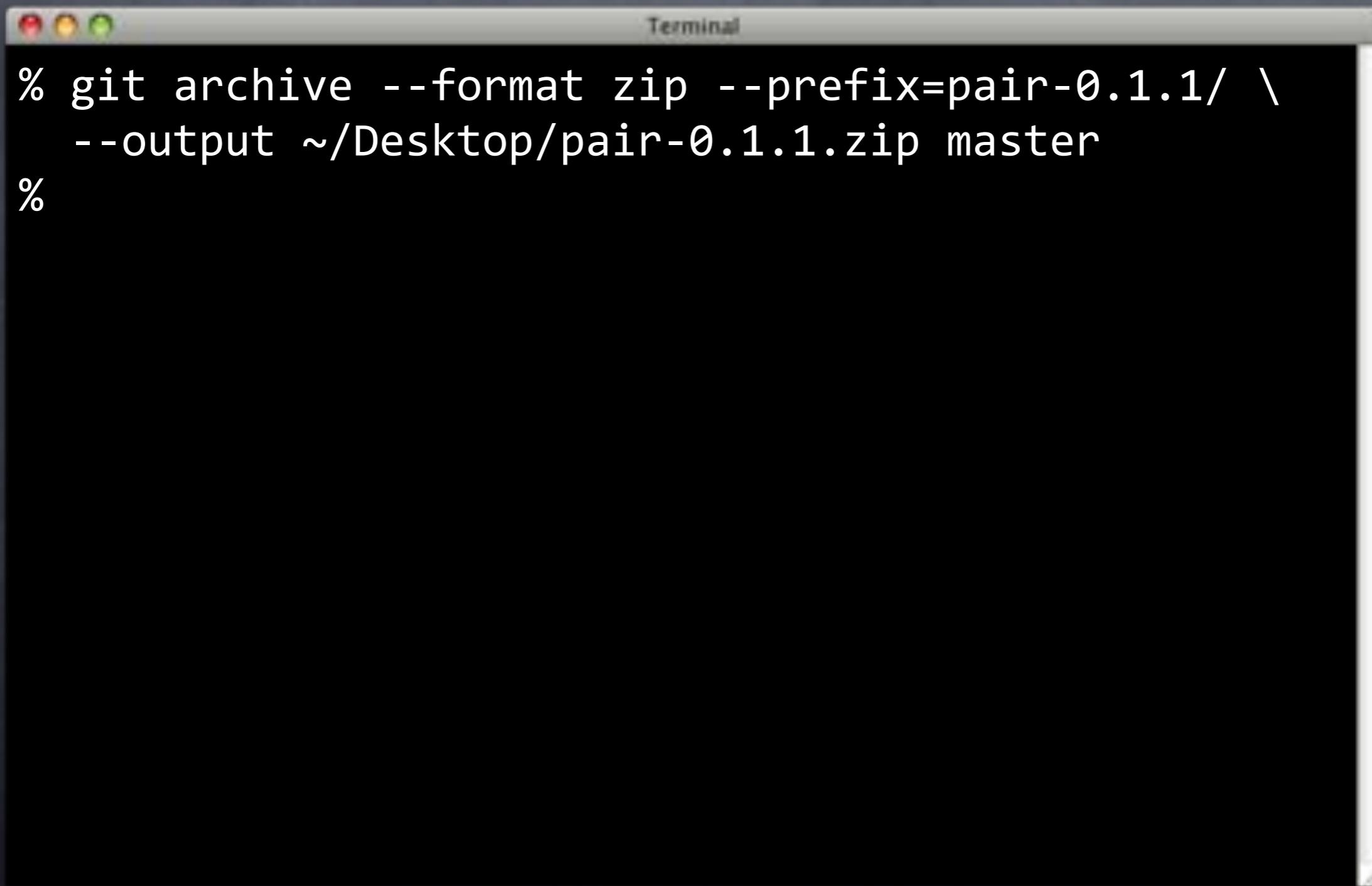
# PGXN Meta Spec

<http://pgxn.org/spec/>

# Package it up



# Package it up



```
Terminal  
% git archive --format zip --prefix=pair-0.1.1/ \  
--output ~/Desktop/pair-0.1.1.zip master  
%
```

# Release it



# PGXN

PostgreSQL Extension Network

tomlane > pair

[recent](#) [users](#) [about](#) [faq](#)

## pair



**This Release:** pair 0.1.1

**Date:** 2010-10-29

**Status:** Stable

**Other Releases:** pair 0.1.1 – 2010-10-29 ↗

**Abstract:** A key/value pair data type

**Description:** This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs.

**Released By:** tomlane

**License:** The PostgreSQL License

**Resources:** [git](#) • [repo](#) • [bugs](#)

**Special Files:** [Changes](#) • [Makefile](#) • [README.md](#) • [META.json](#)

**Tags:** [ordered pair](#) • [pair](#) • [key value](#)

## Extensions

pair 0.1.1

A key/value pair data type

# pair



This Release: pair 0.1.1

Date: 2010-10-29

Status: Stable

Other Releases: pair 0.1.1 — 2010-10-29

Abstract: A key/value pair data type

Description: This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs.

Released By: tomlane

License: The PostgreSQL License

Resources: git • repo • bugs

Special Files: Changes • Makefile • README.md • META.json

Tags: ordered pair • pair • key value

## Extensions

### pair 0.1.1

A key/value pair data type

## README

# pair 0.1.

This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "K" and "V".

The pair data type was created as an inspiration, as documented in this blog post. Give it a read if you're interested in the context of its creation.

# pair

This Release: pair 0.1.1

Date: 2010-10-29

Status: Stable

Other Releases: pair 0.1.1 – 2010-10-29

Abstract: A key/value pair data type

Description: This library contains a single PostgreSQL convenience function for constructing

Released By: tomlane

License: The PostgreSQL License

Resources: git • repo • bugs

Special Files: Changes • Makefile • README.md • META.json

Tags: ordered pair • pair • key value

Lots of great metadata!

## Extensions

### pair 0.1.1

A key/value pair data type

## README

# pair 0.1.

This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "K" and "V".

The pair data type was created as an inspiration, as documented in this blog post. Give it a read if you're interested in the context of its creation.

# pair

This Release: pair 0.1.1

Date: 2010-10-29

Status: Stable

Other Releases: pair 0.1.1 — 2010-10-29

Abstract: A key/value pair data type

Description: This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs.

Released By: tomlane

License: The PostgreSQL License

Resources: git • repo • bugs

Special Files: Changes • Makefile • README.md • META.json

Abstract and doc link

Extensions

pair 0.1.1

A key/value pair data type

README

## pair 0.1.

This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "K" and "V".

The pair data type was created as an inspiration, as documented in this blog post. Give it a read if you're interested in the context of its creation.

# pair



This Release: pair 0.1.1

Date: 2010-10-29

Status: Stable

Other Releases: pair 0.1.1 — 2010-10-29

Abstract: A key/value pair data type

Description: This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs.

Released By: tomlane

License: The PostgreSQL License

Resources: git • repo • bugs

Special Files: Changes • Makefile • README.md • META.json

Tags: ordered pair • pair • key value

## Extensions

### pair 0.1.1

A key/value pair data type

### README

## pair 0.1.

This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "K" and "V".

The pair data type was created as an inspiration, as documented in this blog post. Give it a read if you're interested in the context of its creation.



README in  
HTML!

# pair



This Release: pair 0.1.1

Date: 2010-10-29

Status: Stable

Other Releases: pair 0.1.1 — 2010-10-29

Abstract: A key/value pair data type

Description: This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs.

Released By: tomlane

License: The PostgreSQL License

Resources: git • repo • bugs

Special Files: Changes • Makefile • README.md • META.json

Tags: ordered pair • pair • key value

## Extensions

### pair 0.1.1

A key/value pair data type

### README

## pair 0.1.

This library contains a single PostgreSQL extension, a key/value pair data type called "pair", along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "K" and "V".

The pair data type was created as an inspiration, as documented in this blog post. Give it a read if you're interested in the context of its creation.



README in  
HTML!



# PGXN

PostgreSQL Extension Network

tomlane > pair > pair

[recent](#) [users](#) [about](#) [faq](#)

## Contents

- pair 0.1.1
  - Synopsis
  - Description
  - Usage
  - Support
  - Author
  - Copyright and License

## pair 0.1.1

### Synopsis

```
% CREATE EXTENSION pair;
CREATE EXTENSION pair;

% SELECT 'foo'::text::pair
          AS pair
;
-----+
| (foo,bar) |
-----+
```

### Description

This library contains a single PostgreSQL extension, a key/value pair data type called `pair`, along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "`k`" and "`v`".

So what's it good for? Well, the main idea is if you have a custom function to which you'd like to be able to pass any number of key/value pairs. You could use `hstore` of course, but maybe it's overkill, or you need to guarantee the order in which the pairs are passed. If so, then this extension is for you.

The `pair` data type was created as an inspiration, as documented in this [tiny post](#). Give it a read!



# PGXN

PostgreSQL Extension Network

tomlane &gt; pair &gt; pair

recent users about faq

## Contents

- pair 0.1.1
  - Synopsis
  - Description
  - Usage
  - Support
  - Author
  - Copyright and License

## pair 0.1.1

### Synopsis

```
% CREATE EXTENSION pair;
CREATE EXTENSION pair;

% SELECT 'foo'::text::pair
          AS pair
;
-----+
 (foo,bar)
```

### Description

This library contains a single PostgreSQL extension, a key/value pair data type called `pair`, along with a convenience function for constructing key/value pairs. It's just a simple thing, really: a two-value composite type that can store any type of value in its slots, which are named "`k`" and "`v`".

So what's it good for? Well, the main idea is if you have a custom function to which you'd like to be able to pass any number of key/value pairs. You could use `hstore` of course, but maybe it's overkill, or you need to guarantee the order in which the pairs are passed. If so, then this extension is for you.

# Niiiiice.

# What Else?

# What Else?

- Recommended file layout

# What Else?

- Recommended file layout
- Standard Makefile format

# What Else?

- Recommended file layout
- Standard Makefile format
- 9.1 CREATE EXTENSION support

# What Else?

- Recommended file layout
- Standard Makefile format
- 9.1 CREATE EXTENSION support
  - With compatibility!

# File Recommendations

# File Recommendations

- SQL source in sql/

# File Recommendations

- SQL source in sql/
- C source in src/

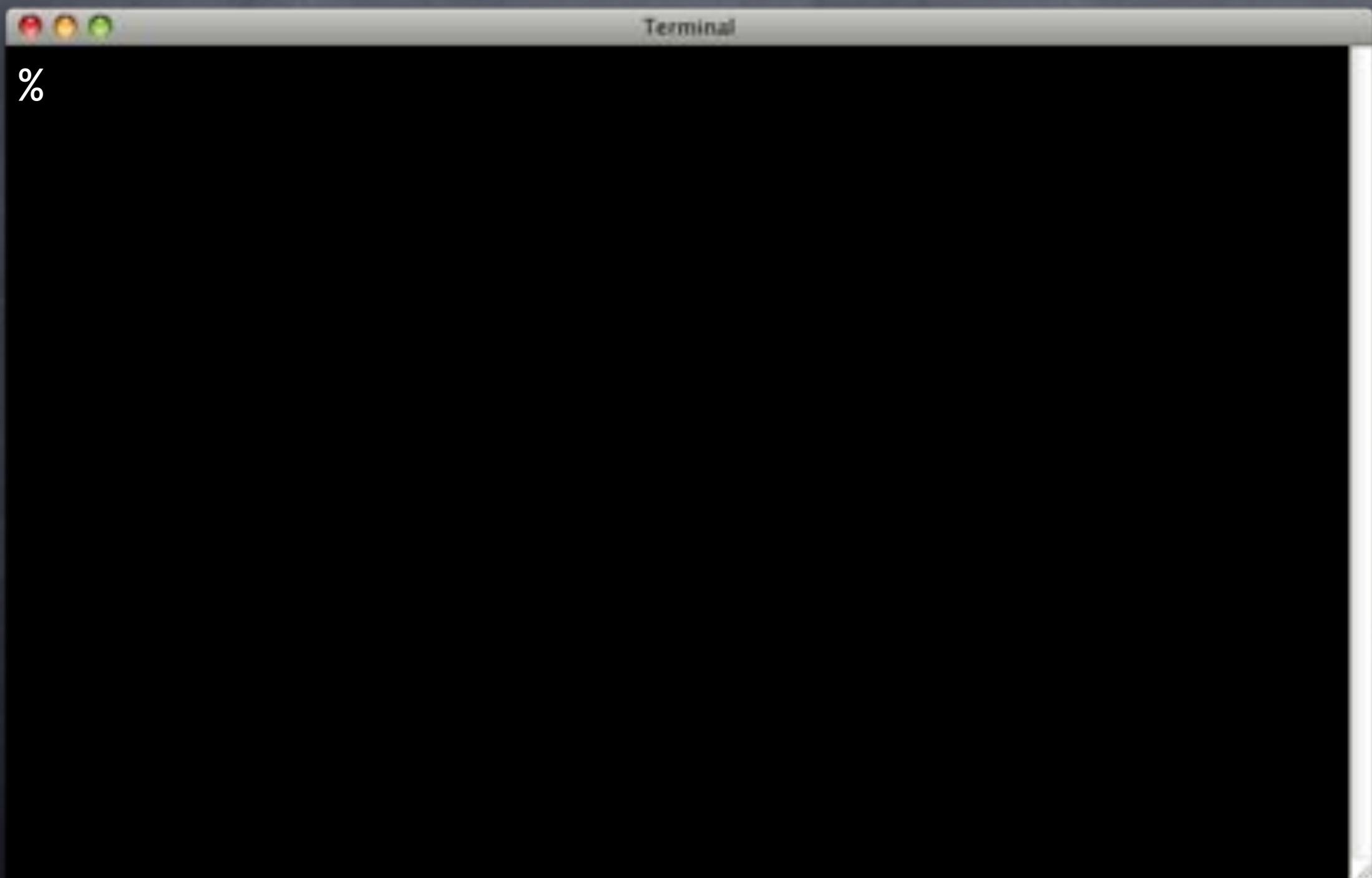
# File Recommendations

- SQL source in sql/
- C source in src/
- Tests in test/

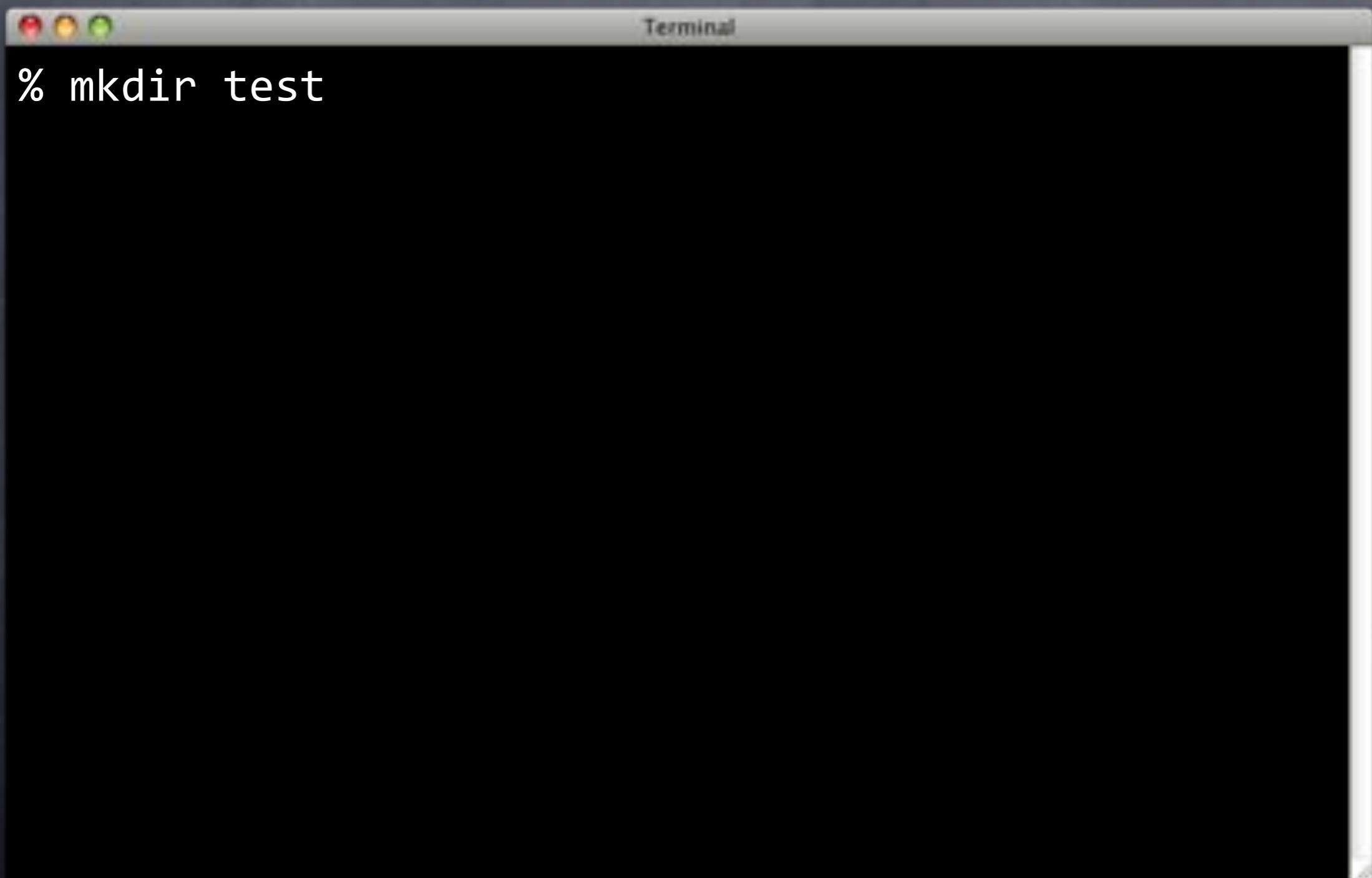
# File Recommendations

- SQL source in `sql/`
- C source in `src/`
- Tests in `test/`
- Documentation in `doc/`

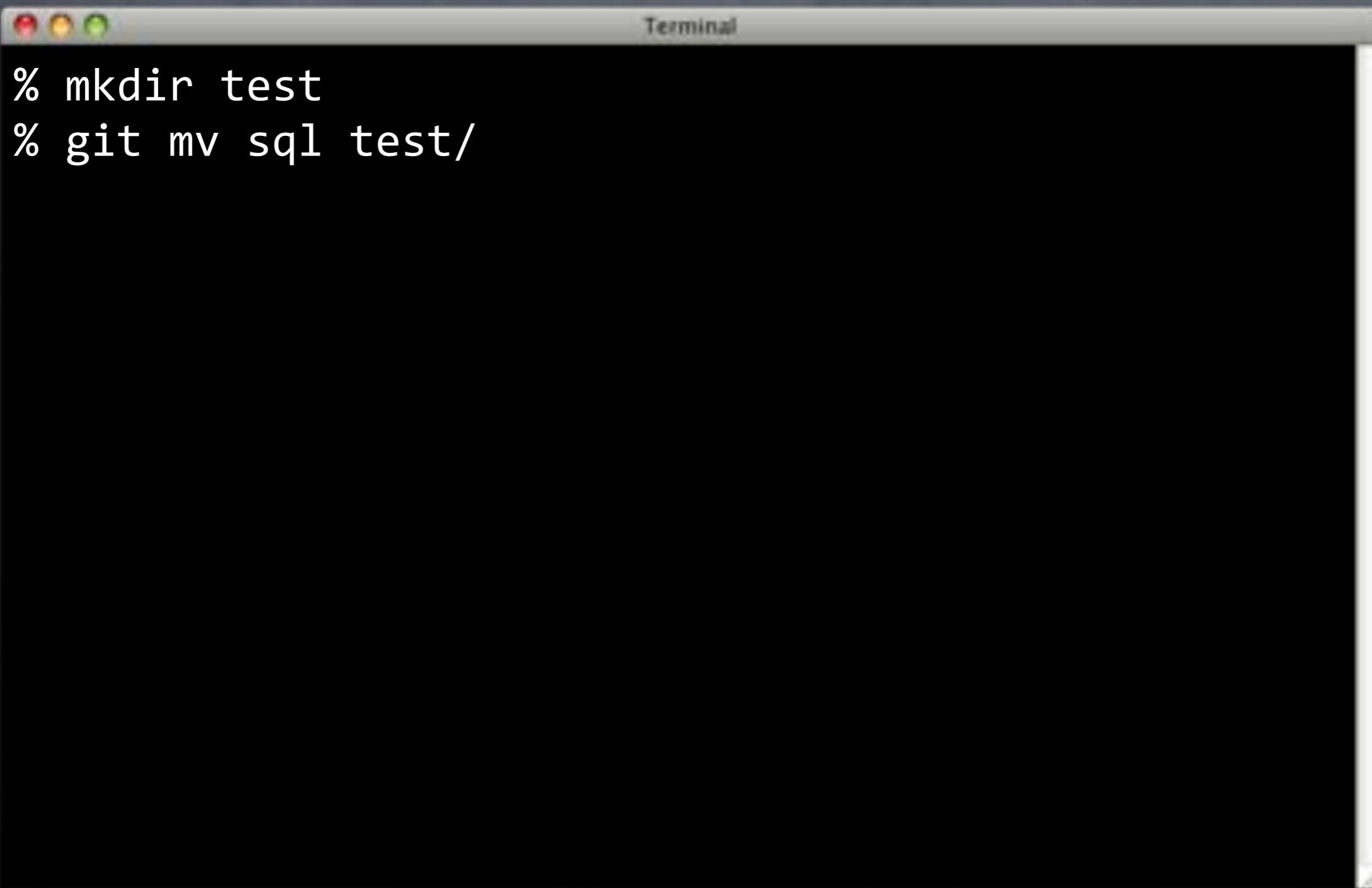
# Rearrange



# Rearrange

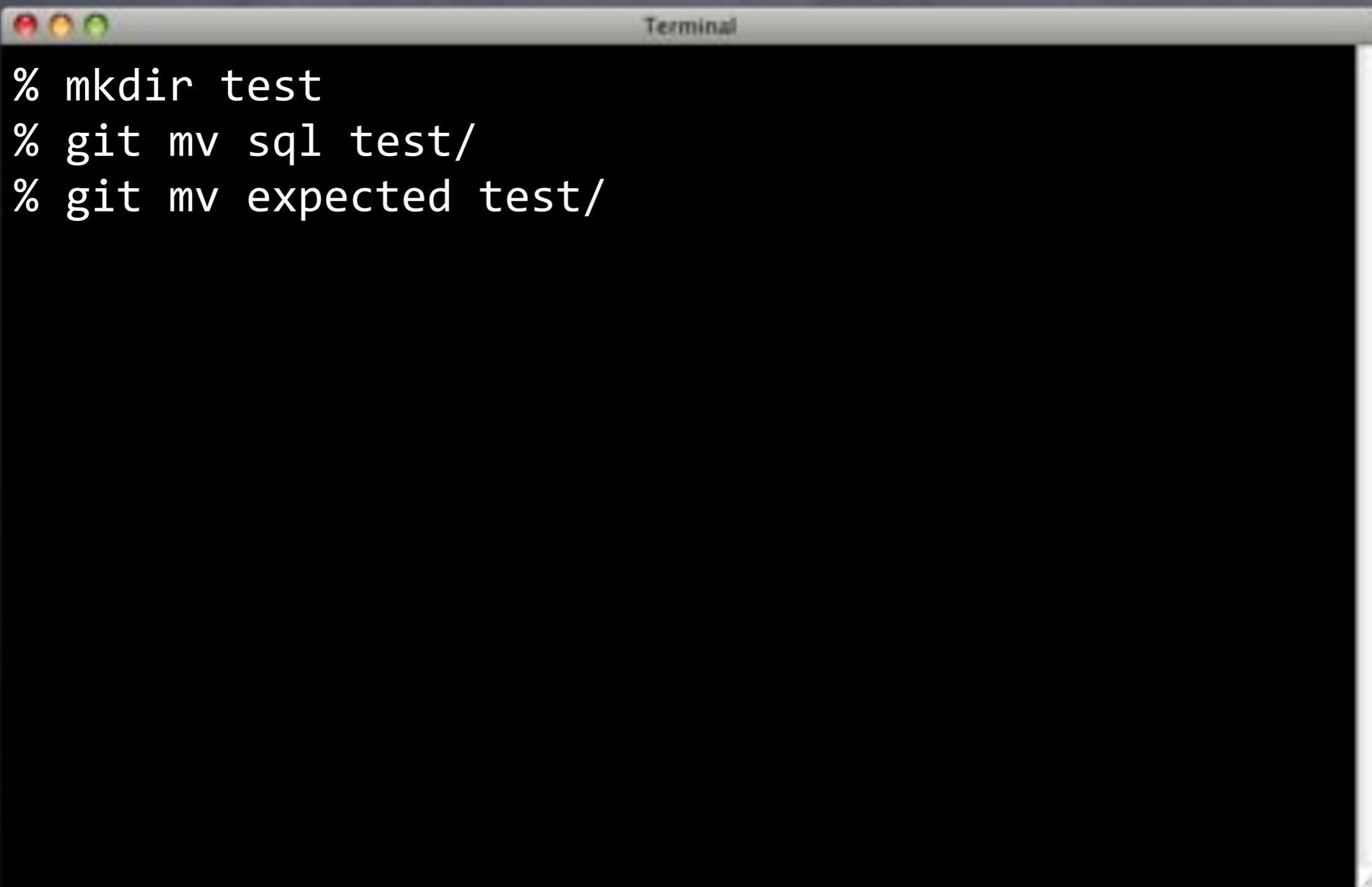


# Rearrange



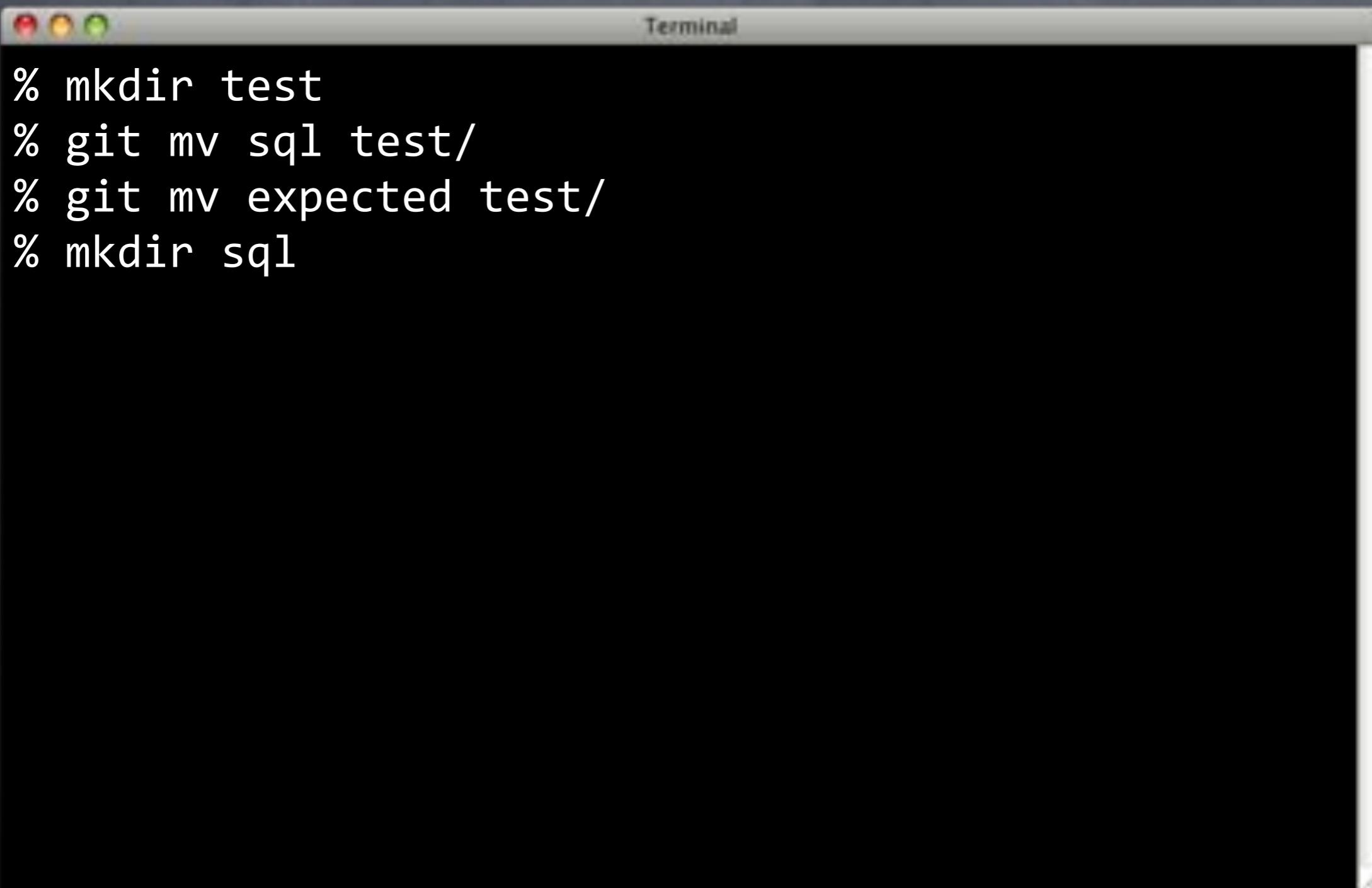
```
% mkdir test
% git mv sql test/
```

# Rearrange



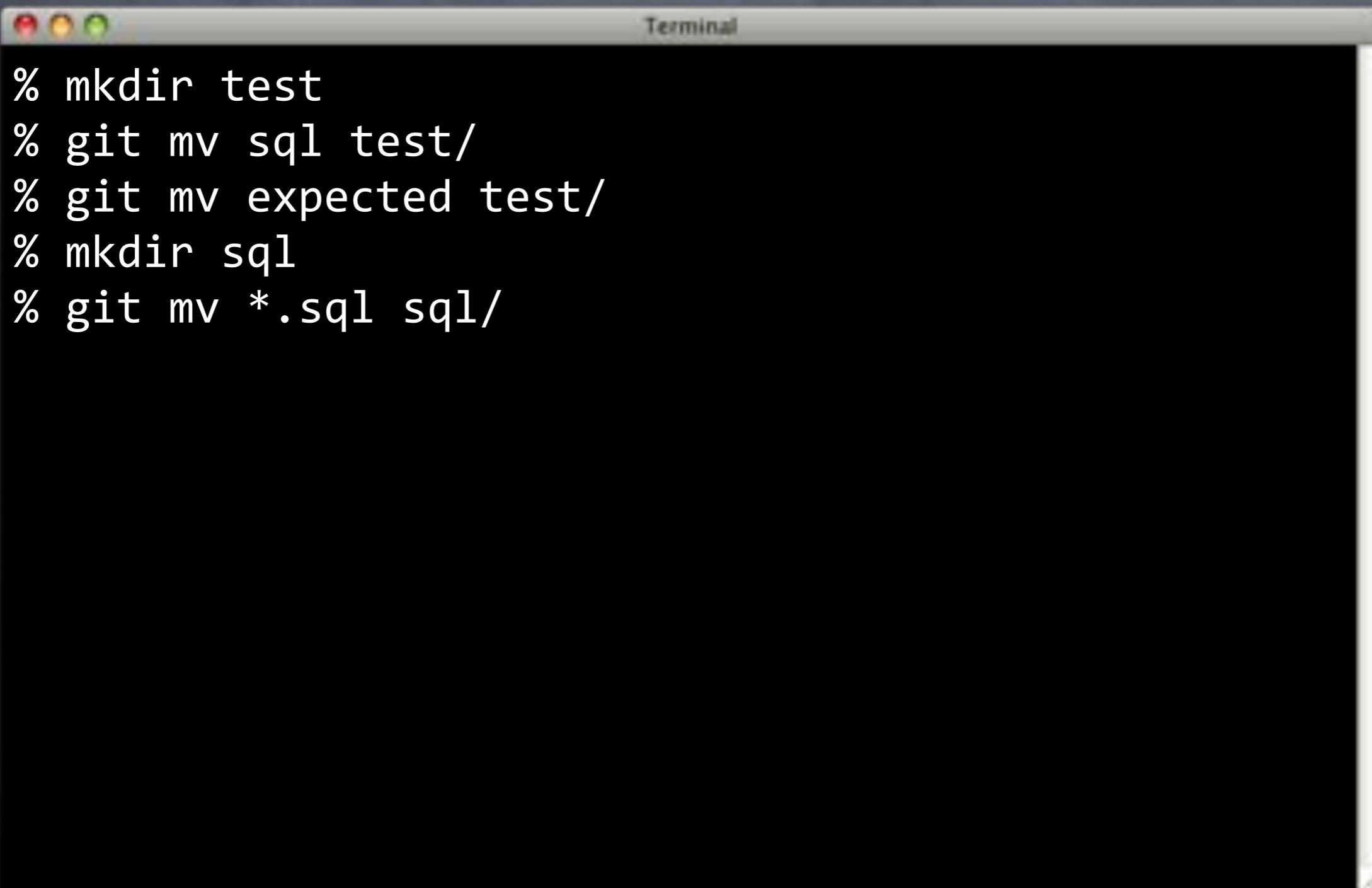
```
% mkdir test
% git mv sql test/
% git mv expected test/
```

# Rearrange



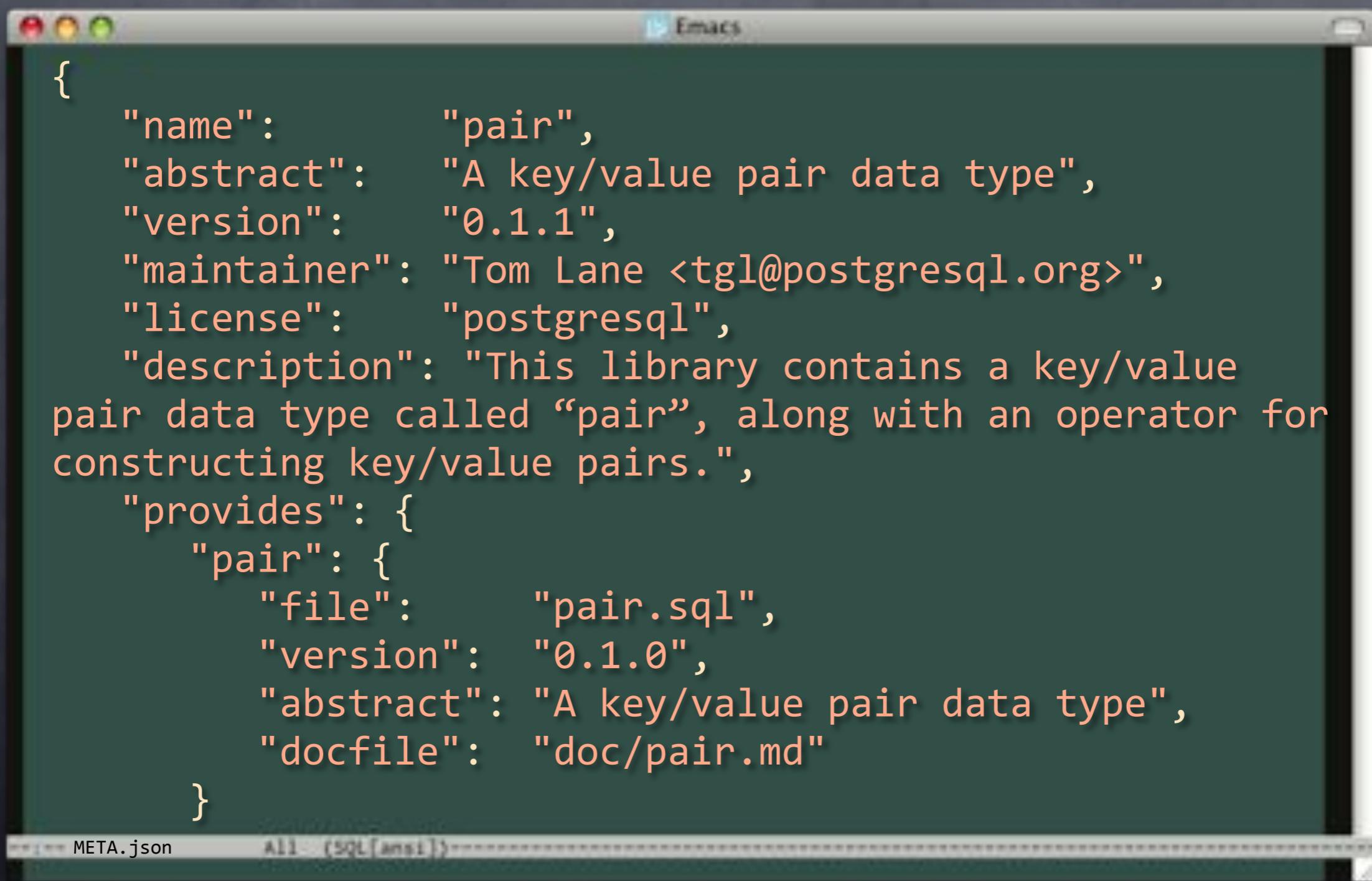
```
% mkdir test
% git mv sql test/
% git mv expected test/
% mkdir sql
```

# Rearrange



```
% mkdir test
% git mv sql test/
% git mv expected test/
% mkdir sql
% git mv *.sql sql/
```

# Update Path

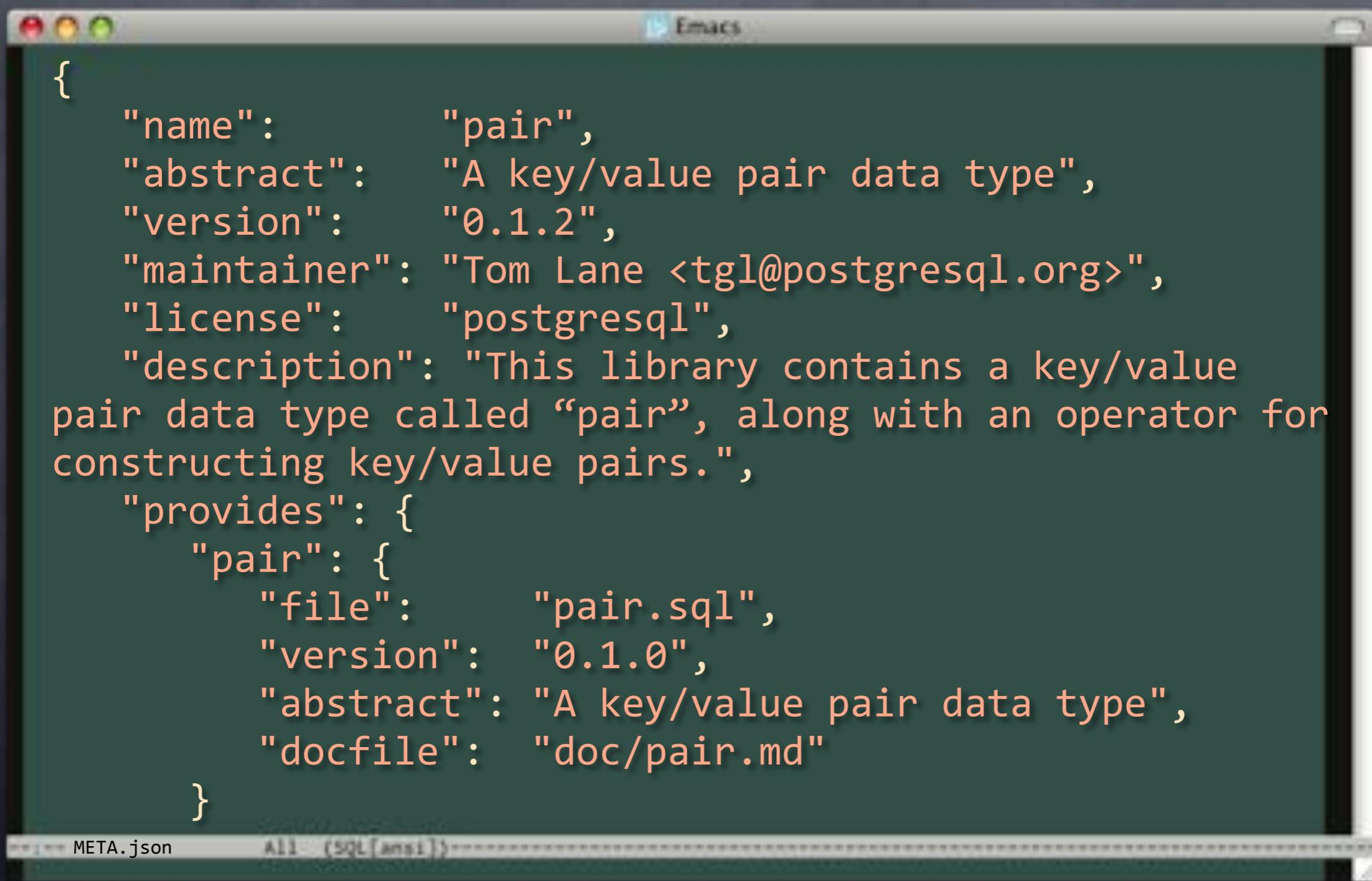


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains a JSON object with the following structure:

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.1",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

The file is named "META.json" as indicated in the bottom left corner of the window.

# Update Path

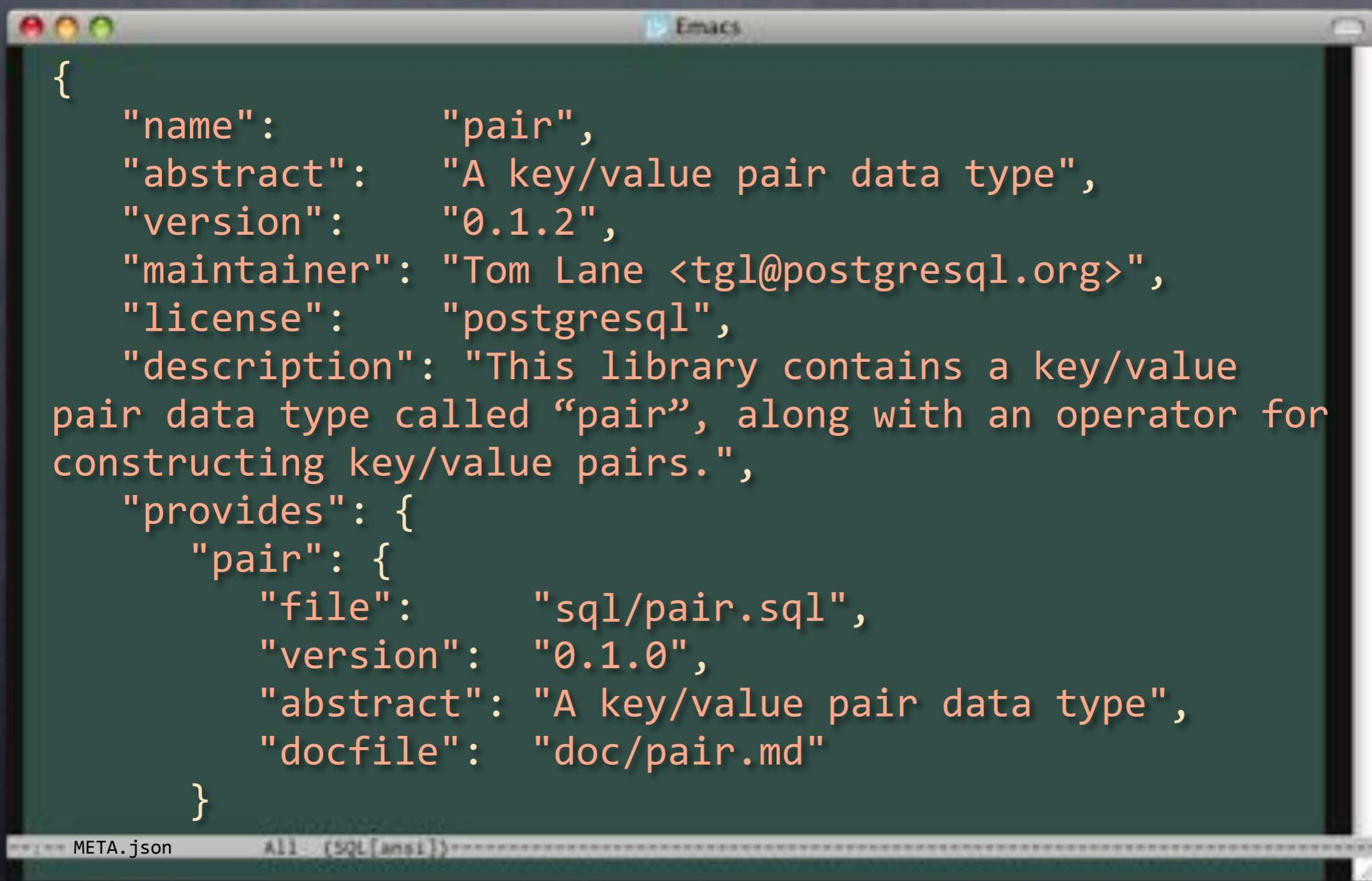


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains a JSON object with the following structure:

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.2",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

The file is named "META.json" as indicated in the bottom left corner of the window.

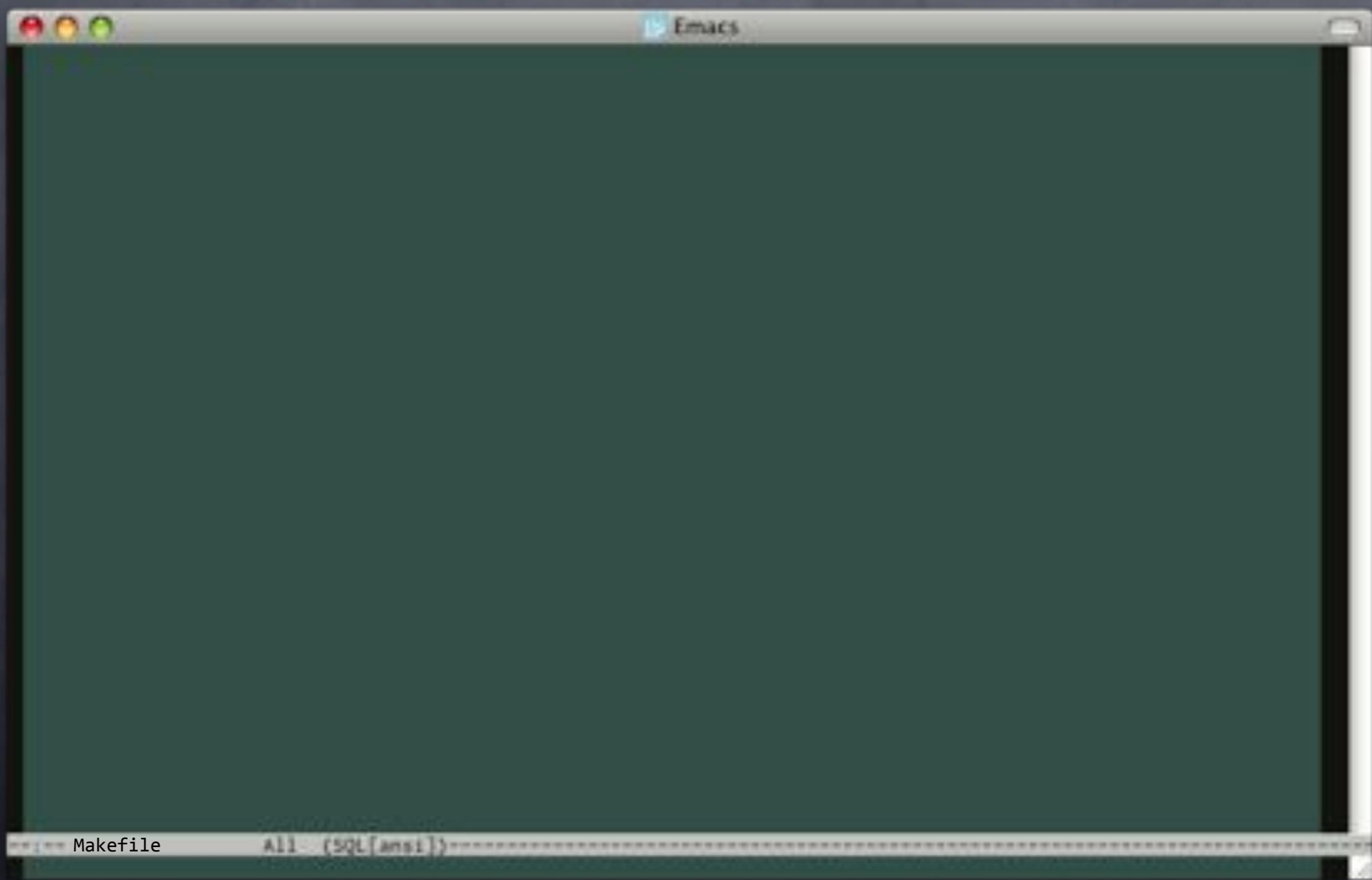
# Update Path



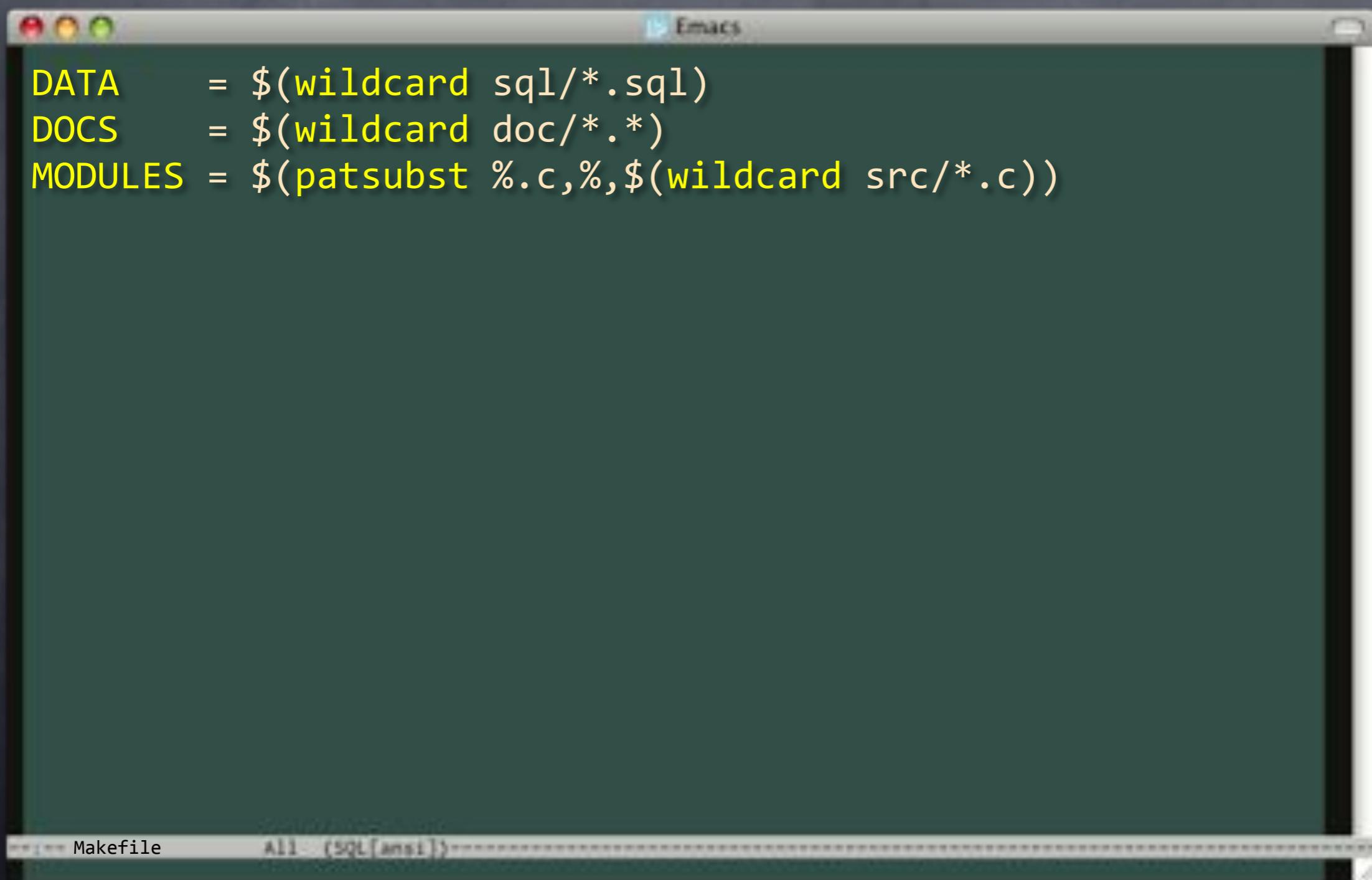
The image shows a screenshot of an Emacs window with a dark background. The title bar reads "Emacs". The buffer contains a JSON object representing a library named "pair". The "description" field includes a detailed explanation of the library's purpose. The "provides" field contains another JSON object for the "pair" module, which includes a "file" field pointing to "sql/pair.sql".

```
{  
  "name": "pair",  
  "abstract": "A key/value pair data type",  
  "version": "0.1.2",  
  "maintainer": "Tom Lane <tgl@postgresql.org>",  
  "license": "postgresql",  
  "description": "This library contains a key/value pair data type called \"pair\", along with an operator for constructing key/value pairs.",  
  "provides": {  
    "pair": {  
      "file": "sql/pair.sql",  
      "version": "0.1.0",  
      "abstract": "A key/value pair data type",  
      "docfile": "doc/pair.md"  
    }  
  }  
}
```

# Makefile



# Makefile

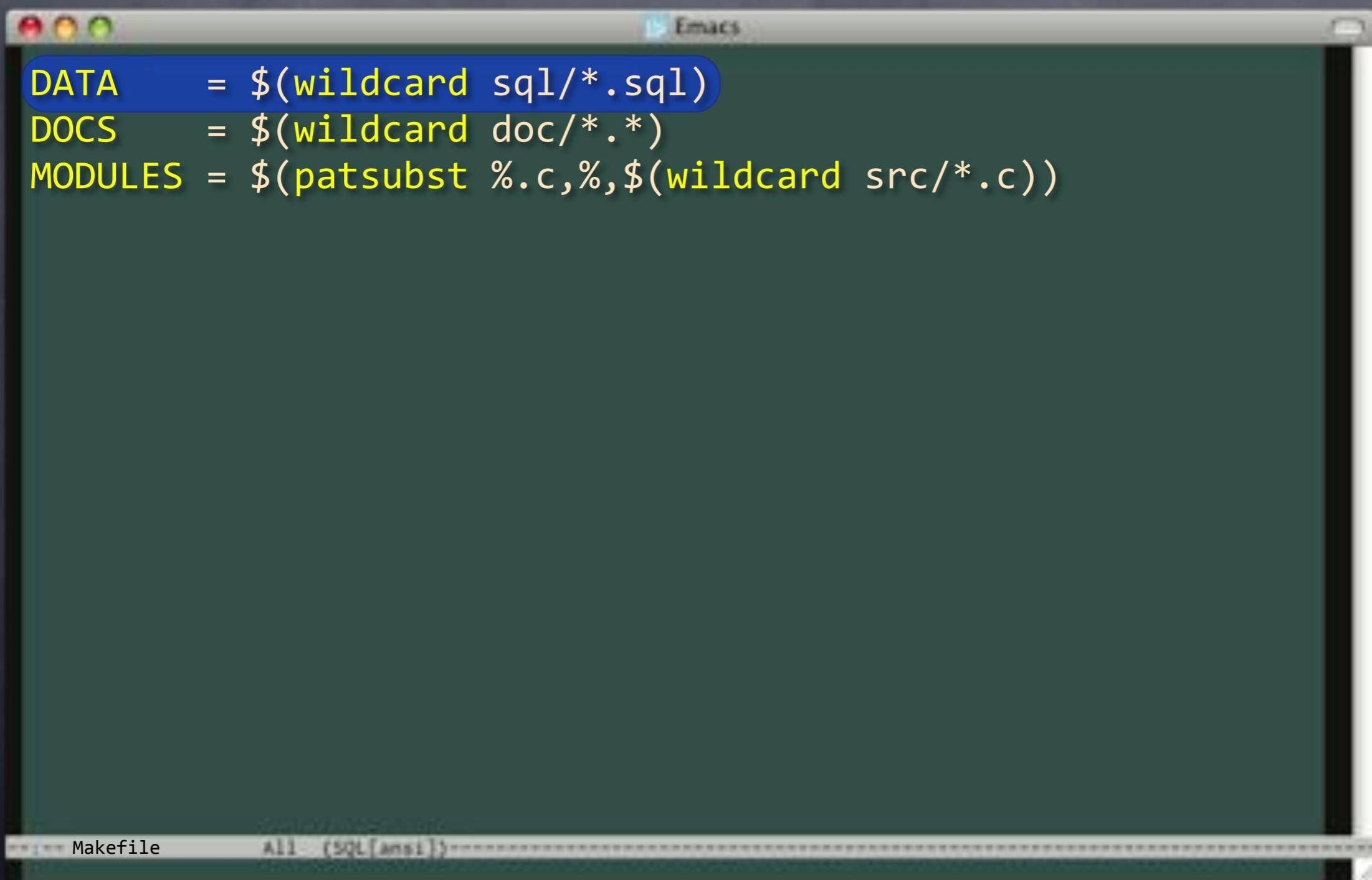


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following Makefile code:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))
```

The status bar at the bottom left shows "Makefile" and "All - (SQL[ansi])".

# Makefile



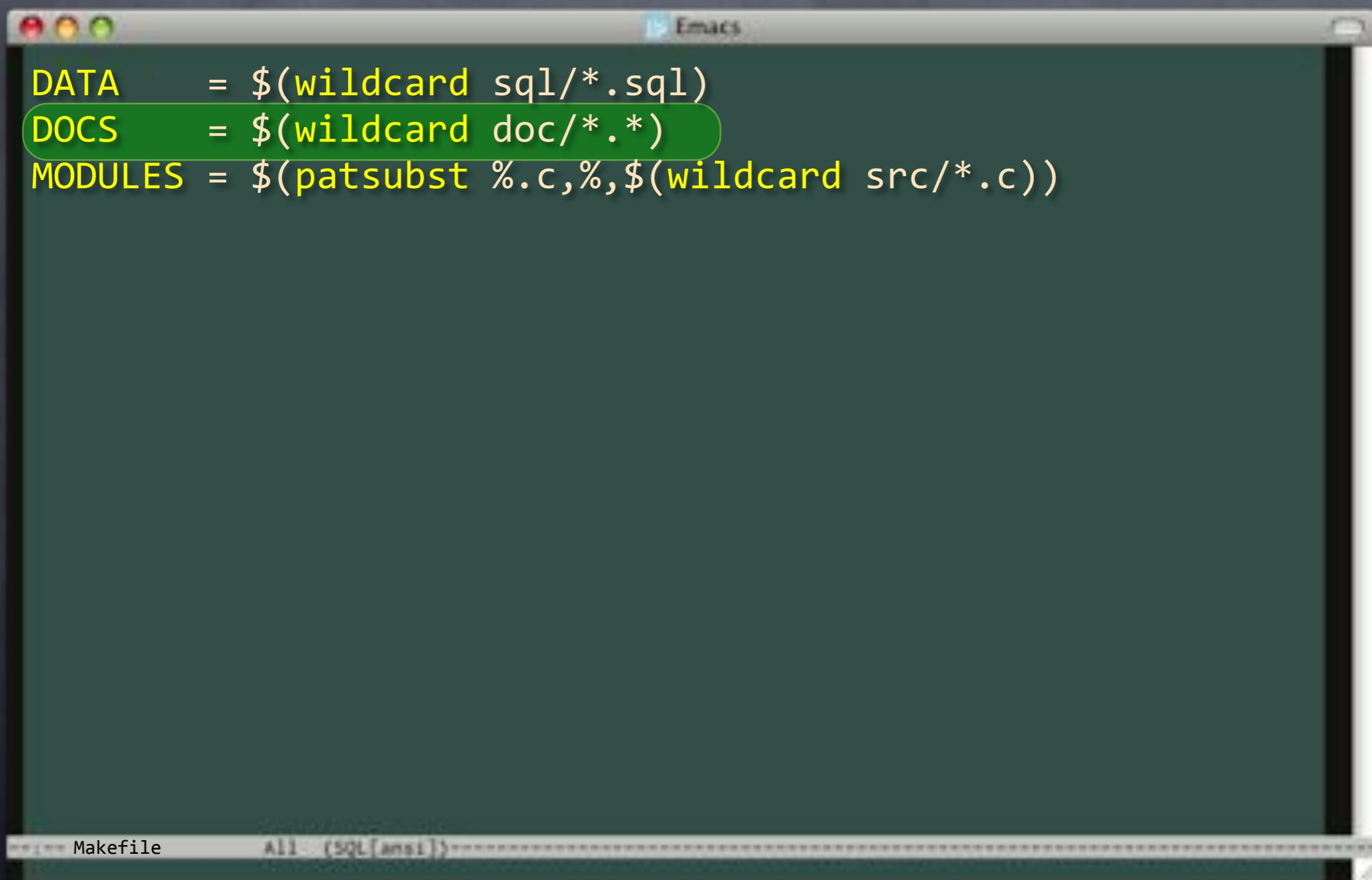
The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following Makefile code:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))
```

The first line, "DATA = \$(wildcard sql/\*.sql)", is highlighted with a blue rectangle.

At the bottom of the window, the status bar displays "Makefile" on the left and "All- (SQL[ansi])" on the right.

# Makefile

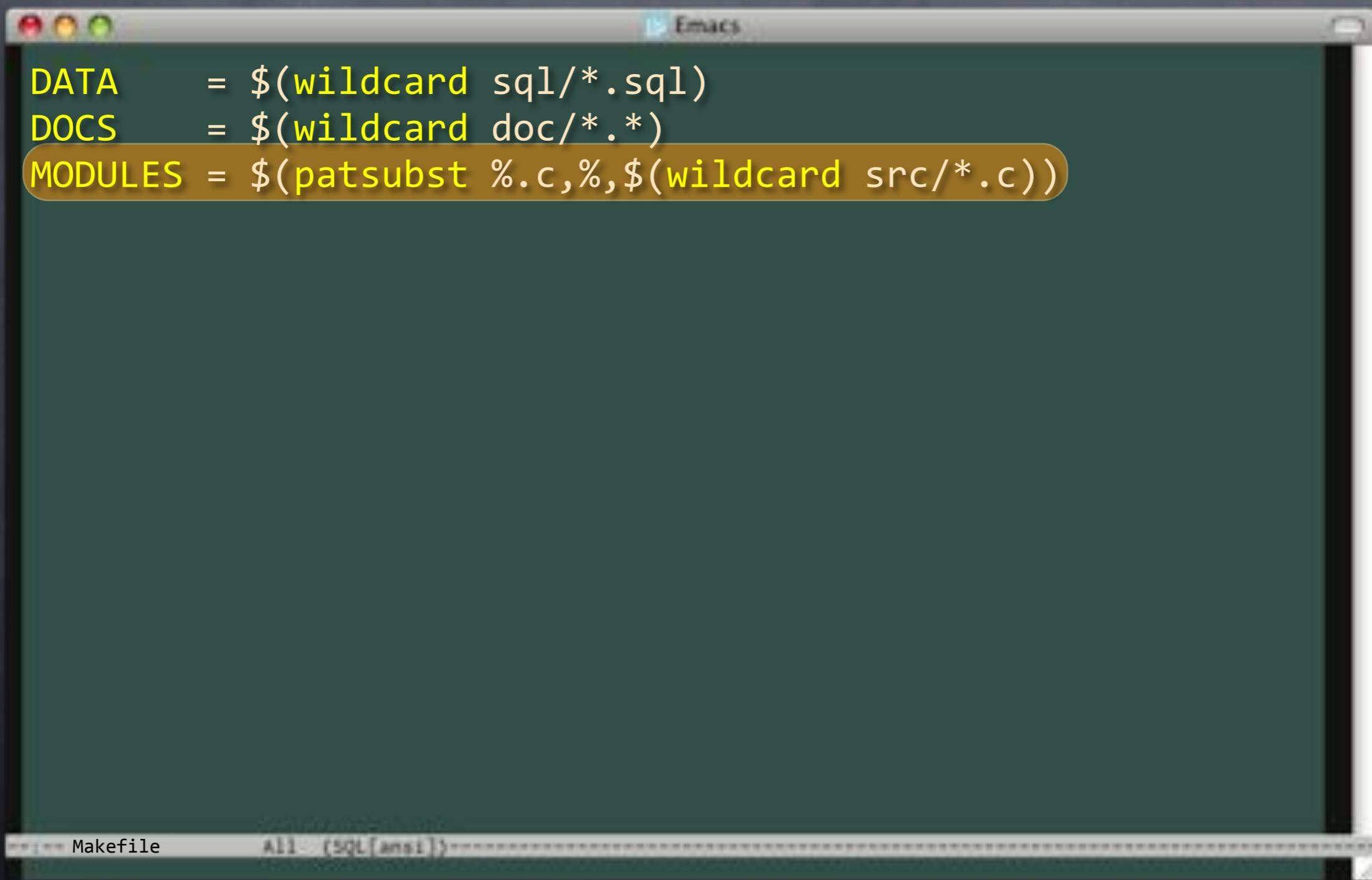


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following Makefile code:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))
```

The word "MODULES" is highlighted with a green oval. The status bar at the bottom left shows "Makefile" and "All - (SQL[ansi])".

# Makefile



The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following Makefile code:

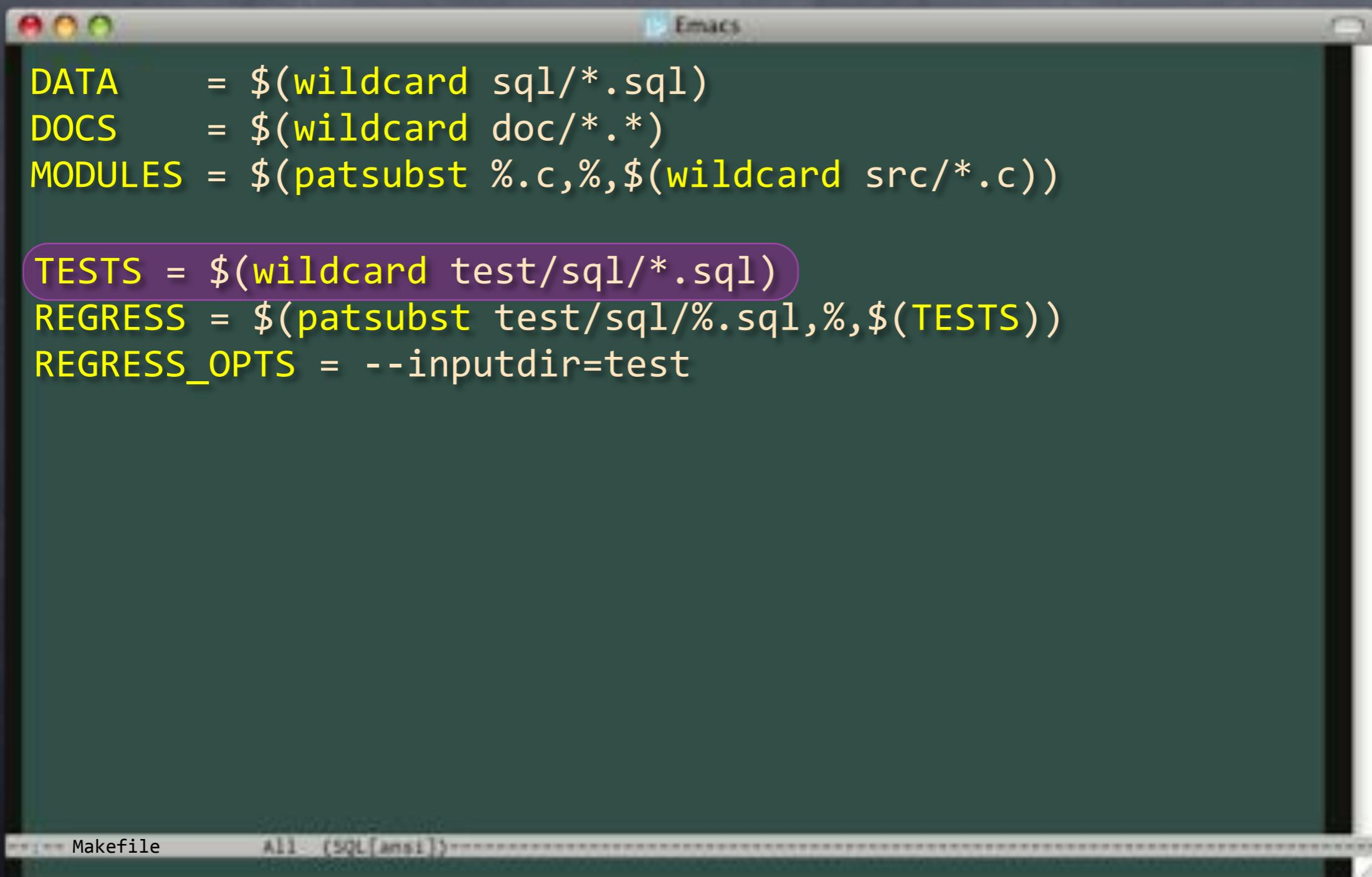
```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))
```

The third line, "MODULES", is highlighted with a yellow rectangle. The status bar at the bottom left shows "Makefile" and "All - (SQL[ansi])".

# Makefile

```
Emacs  
DATA      = $(wildcard sql/*.sql)  
DOCS      = $(wildcard doc/*.* )  
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))  
  
TESTS    = $(wildcard test/sql/*.sql)  
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))  
REGRESS_OPTS = --inputdir=test
```

# Makefile



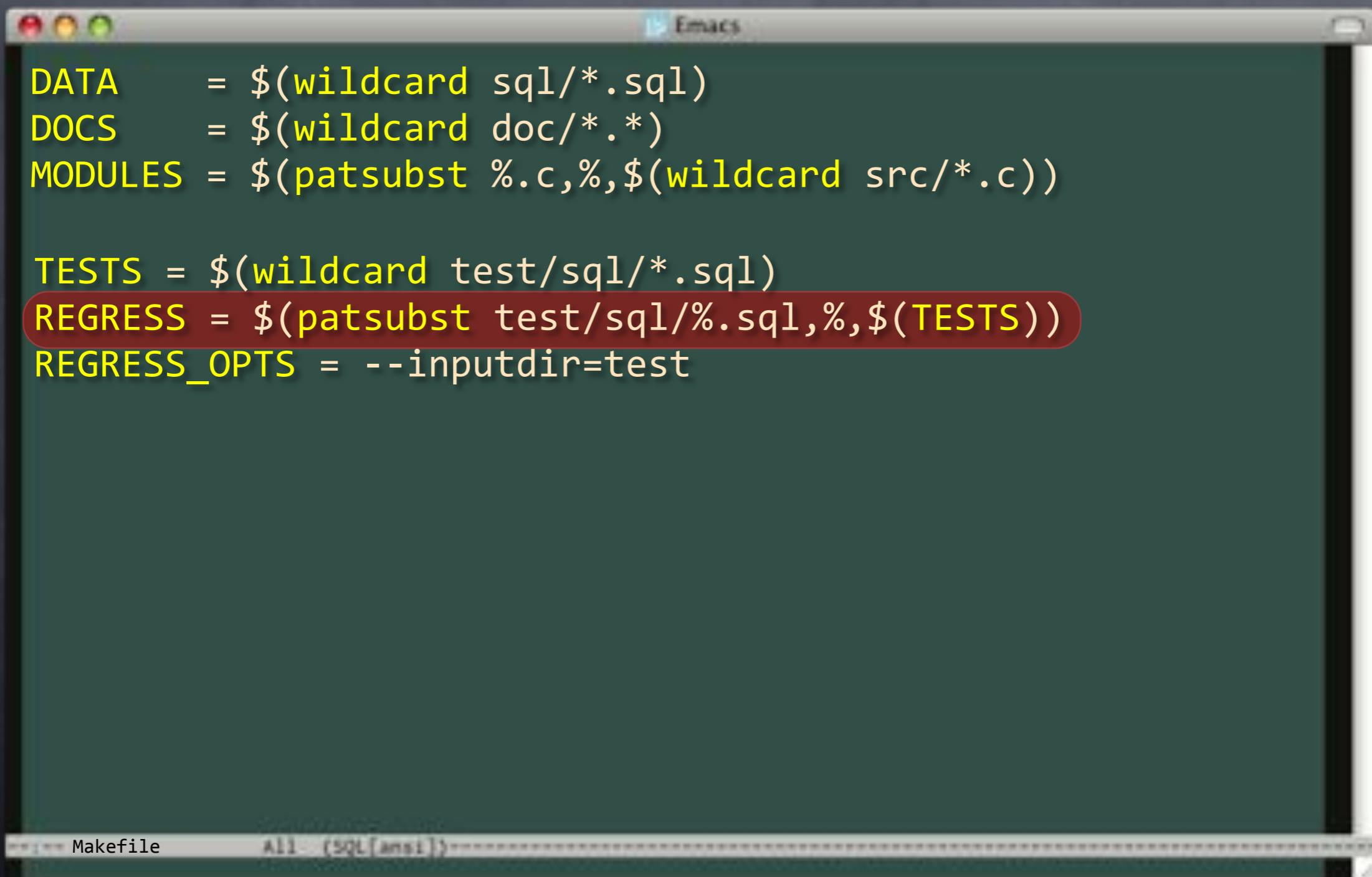
The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains a Makefile with the following content:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))

TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test
```

The line "TESTS = \$(wildcard test/sql/\*.sql)" is highlighted with a purple oval around it.

# Makefile



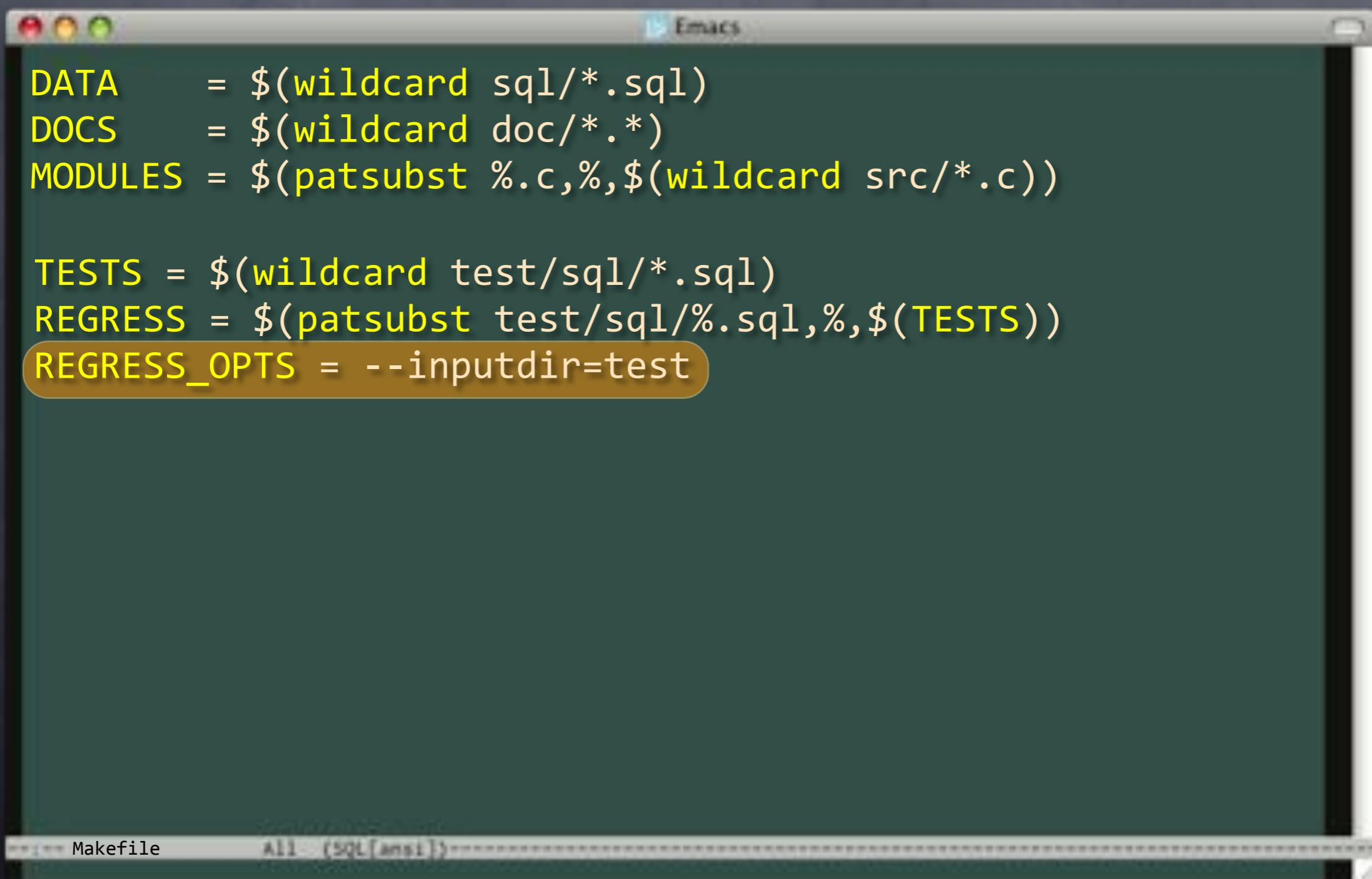
The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains a Makefile with the following content:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))

TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test
```

The line "REGRESS = \$(patsubst test/sql/%.sql,%,\$(TESTS))" is highlighted with a red rounded rectangle.

# Makefile



The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains a Makefile with the following content:

```
DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))

TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test
```

The line "REGRESS\_OPTS = --inputdir=test" is highlighted with a yellow oval.

# Makefile



```
DATA      = $(wildcard data/*)
DOCS      = $(wildcard docs/*)
MODULES   = $(patsubst %.c,$(wildcard src/*.c))
TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test
```

Makefile All - (SQL [ansi])

# Makefile

```
Emacs  
DATA      = $(wildcard sql/*.sql)  
DOCS      = $(wildcard doc/*.* )  
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))  
  
TESTS     = $(wildcard test/sql/*.sql)  
REGRESS   = $(patsubst test/sql/%.sql,%,$(TESTS))  
REGRESS_OPTS = --inputdir=test  
  
PG_CONFIG = pg_config  
PGXS := $(shell $(PG_CONFIG) --pgxs)  
include $(PGXS)
```

# Makefile

```
Emacs

DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))

TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test

PG_CONFIG = pg_config
PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Makefile

```
Emacs

DATA      = $(wildcard sql/*.sql)
DOCS      = $(wildcard doc/*.*)
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))

TESTS    = $(wildcard test/sql/*.sql)
REGRESS  = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS = --inputdir=test

PG_CONFIG = pg_config
PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)

Makefile  All- (SQL [ansi])-----
```

# Makefile

```
Emacs  
DATA      = $(wildcard sql/*.sql)  
DOCS      = $(wildcard doc/*.* )  
MODULES   = $(patsubst %.c,%,$(wildcard src/*.c))  
  
TESTS     = $(wildcard test/sql/*.sql)  
REGRESS   = $(patsubst test/sql/%.sql,%,$(TESTS))  
REGRESS_OPTS = --inputdir=test  
  
PG_CONFIG = pg_config  
PGXS := $(shell $(PG_CONFIG) --pgxs)  
include $(PGXS)
```

# CREATE EXTENSION

# CREATE EXTENSION

- New in 9.1

# CREATE EXTENSION

- New in 9.1
- Add extension to a DB with

# CREATE EXTENSION

- New in 9.1
- Add extension to a DB with
  - `CREATE EXTENSION` pair;

# CREATE EXTENSION

- New in 9.1
- Add extension to a DB with
  - CREATE EXTENSION pair;
- No need to run SQL script in psql

# Packaging Extensions Needs

# Packaging Extensions Needs

- Control file

# Packaging Extensions Needs

- Control file
- Migration from unpackaged

# Packaging Extensions Needs

- Control file
- Migration from unpackaged
- pair--unpackaged--0.1.0.sql

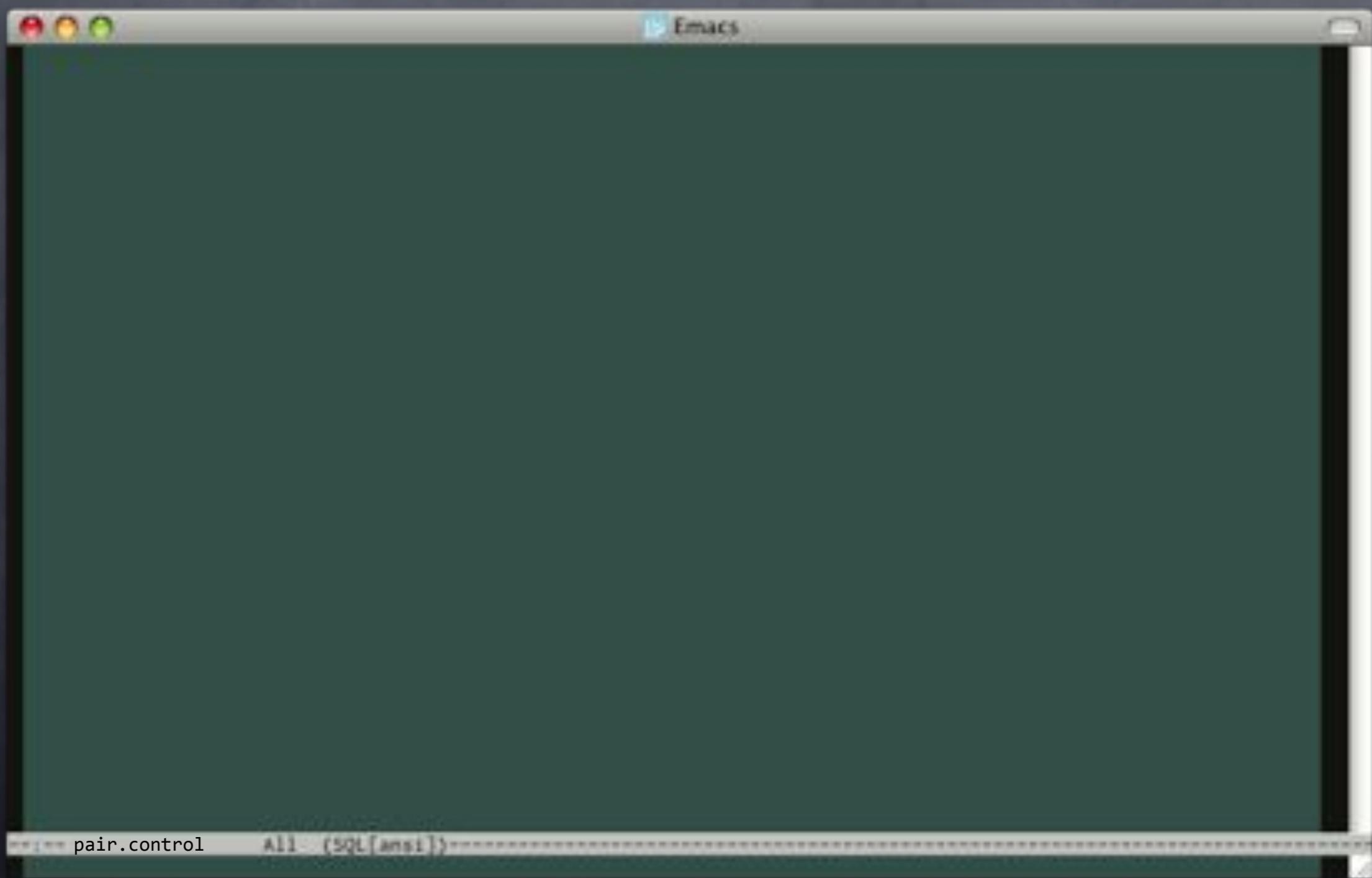
# Packaging Extensions Needs

- Control file
- Migration from unpackaged
  - pair--unpackaged--0.1.0.sql
- Properly-named SQL script

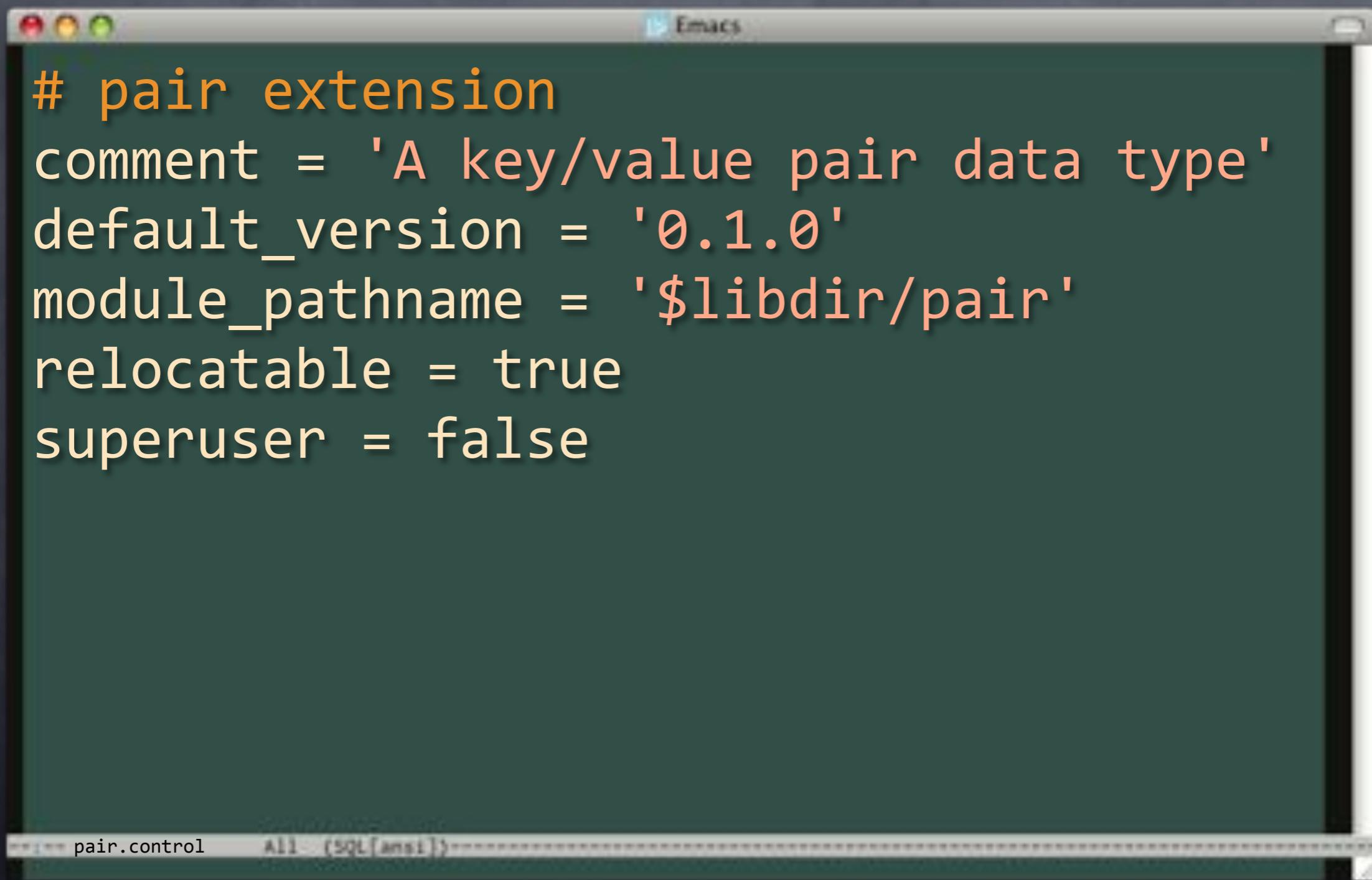
# Packaging Extensions Needs

- Control file
- Migration from unpackaged
  - pair--unpackaged--0.1.0.sql
- Properly-named SQL script
  - pair--0.1.0.sql

# Create the control file



# Create the control file

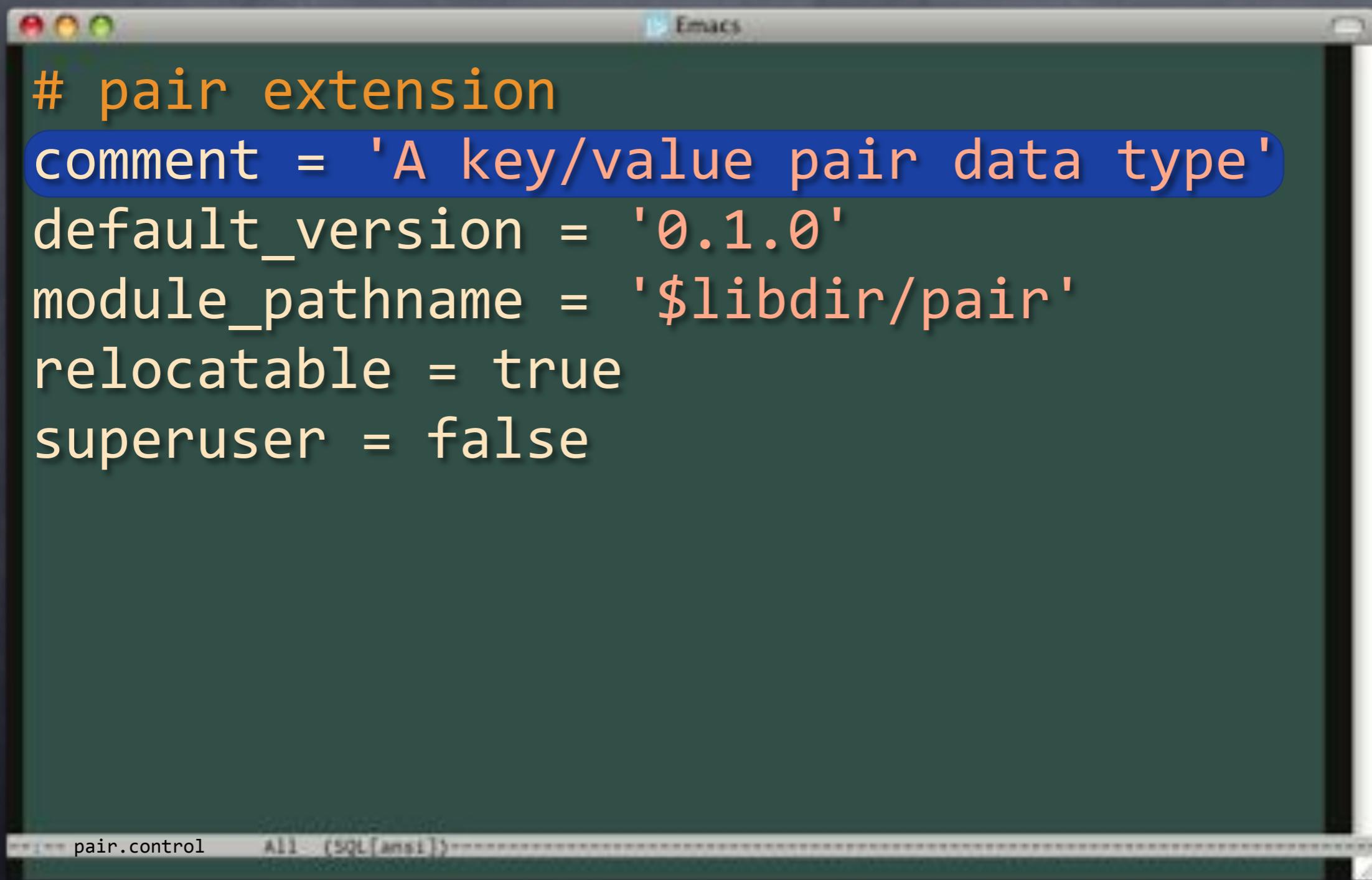


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '$libdir/pair'
relocatable = true
superuser = false
```

In the bottom left corner of the window, the file name "pair.control" is visible. The status bar at the bottom of the screen also displays "pair.control".

# Create the control file



The image shows a screenshot of an Emacs window with a dark green background and a white border. The title bar says "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '$libdir/pair'
relocatable = true
superuser = false
```

The line "comment = 'A key/value pair data type'" is highlighted with a blue rectangle. The status bar at the bottom shows "pair.control" and "All (SQL [ansi])".

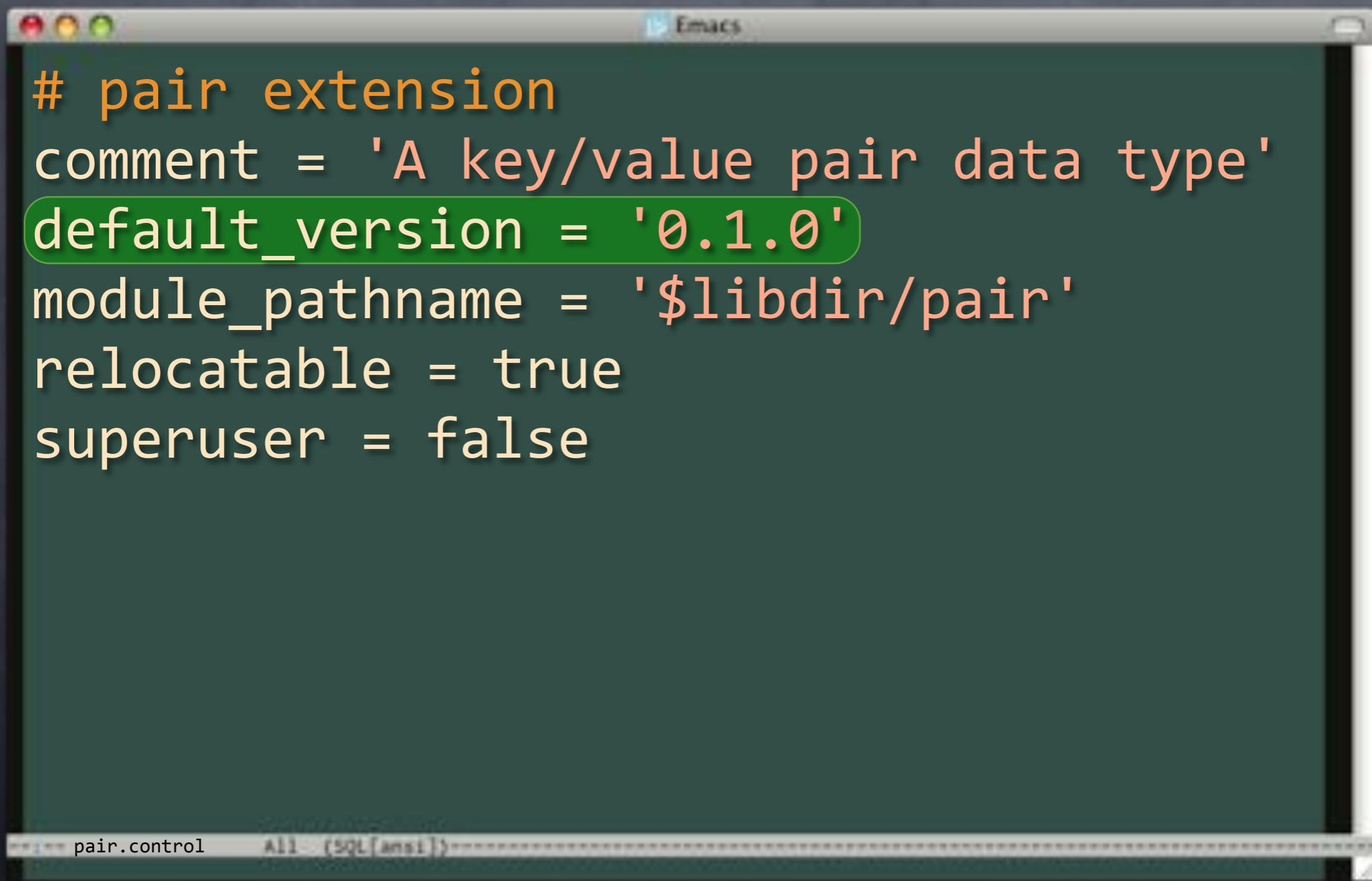
# Create the control file

Optional

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

pair.control All (SQL ANSI)

# Create the control file

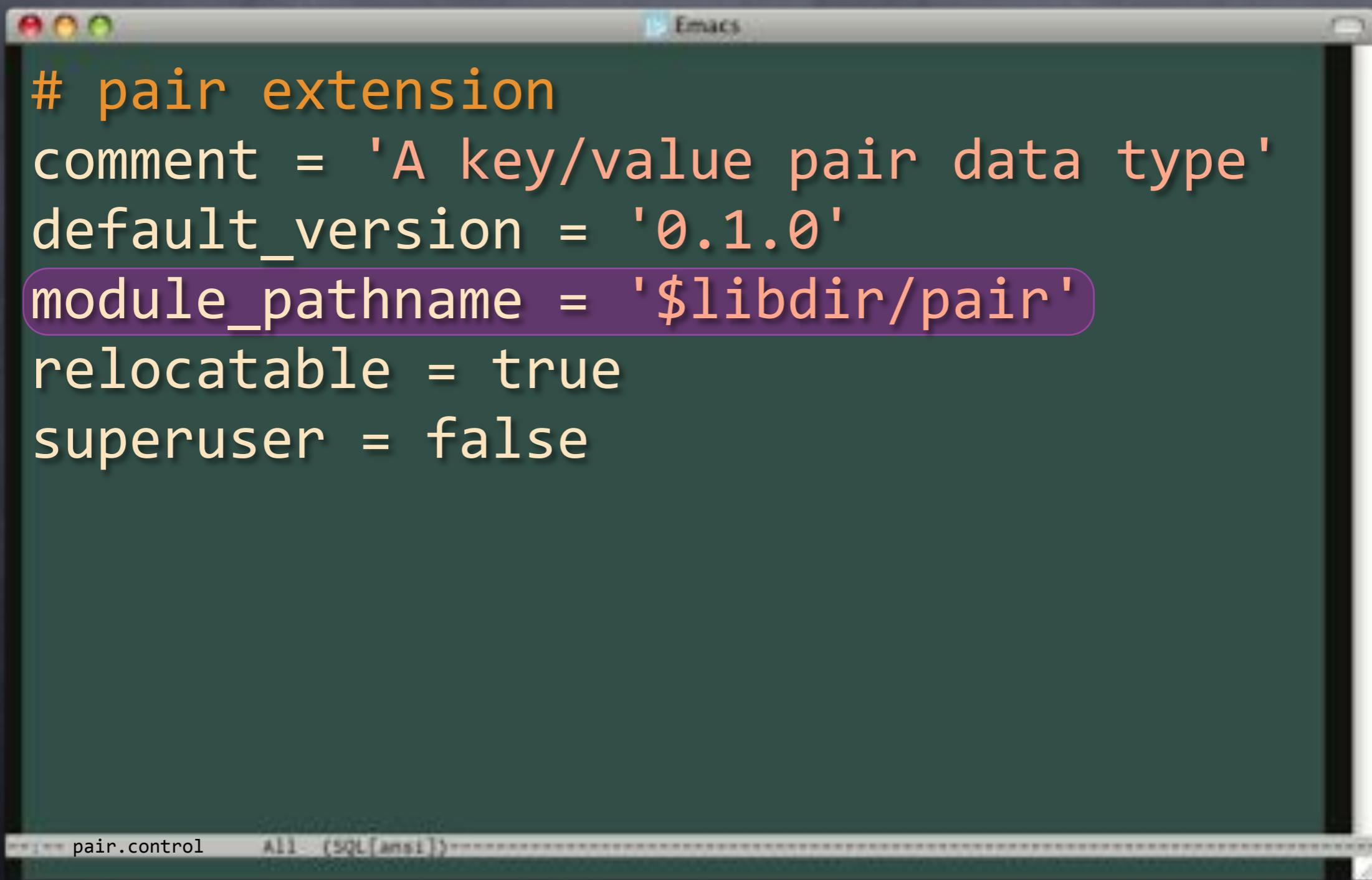


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

The line "default\_version = '0.1.0'" is highlighted with a green rounded rectangle. The status bar at the bottom left shows "pair.control" and "All (SQL [ansi])".

# Create the control file

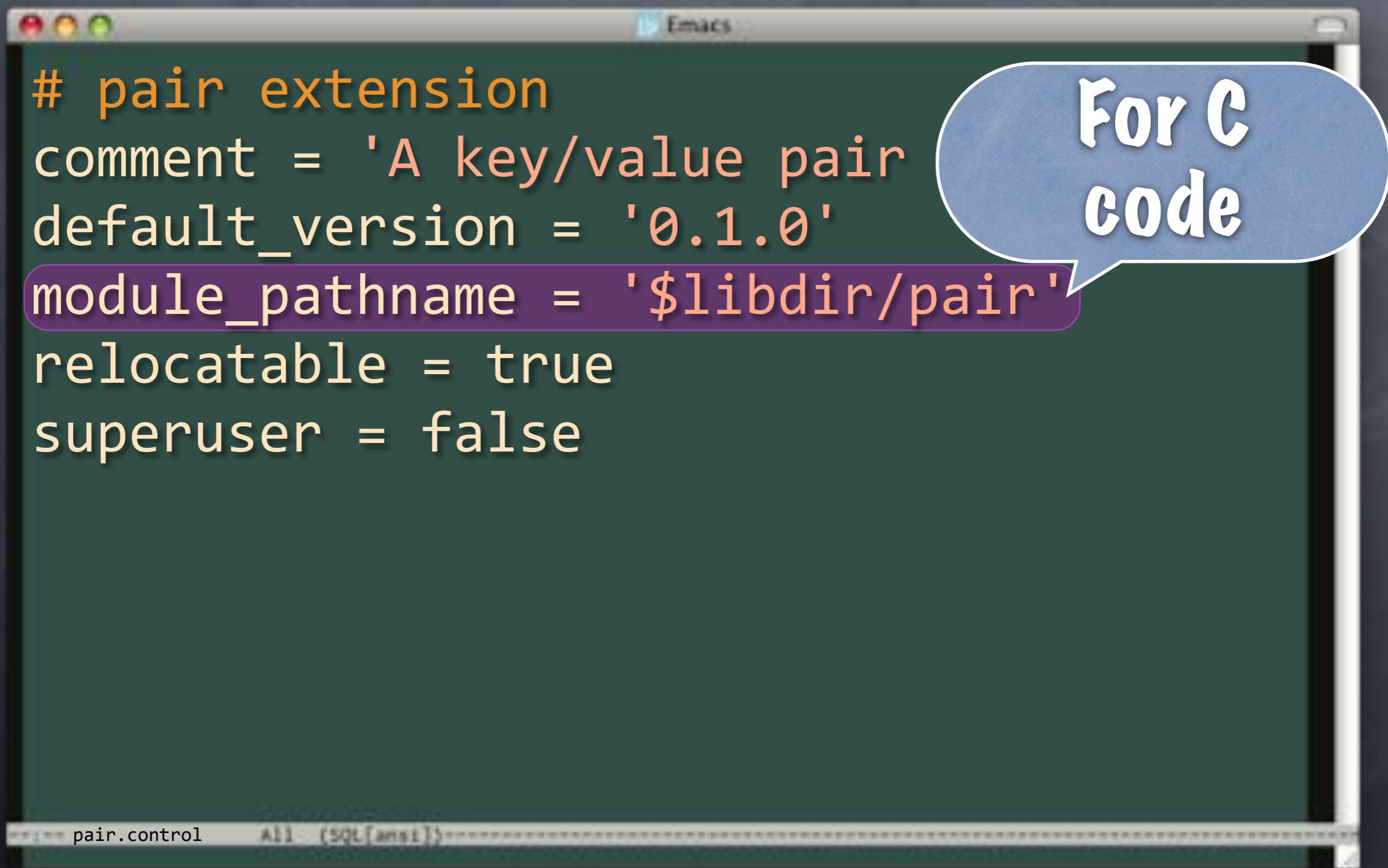


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

The line "module.pathname = '\$libdir/pair'" is highlighted with a purple oval. The status bar at the bottom left shows "pair.control" and "All (SQL [ansi])".

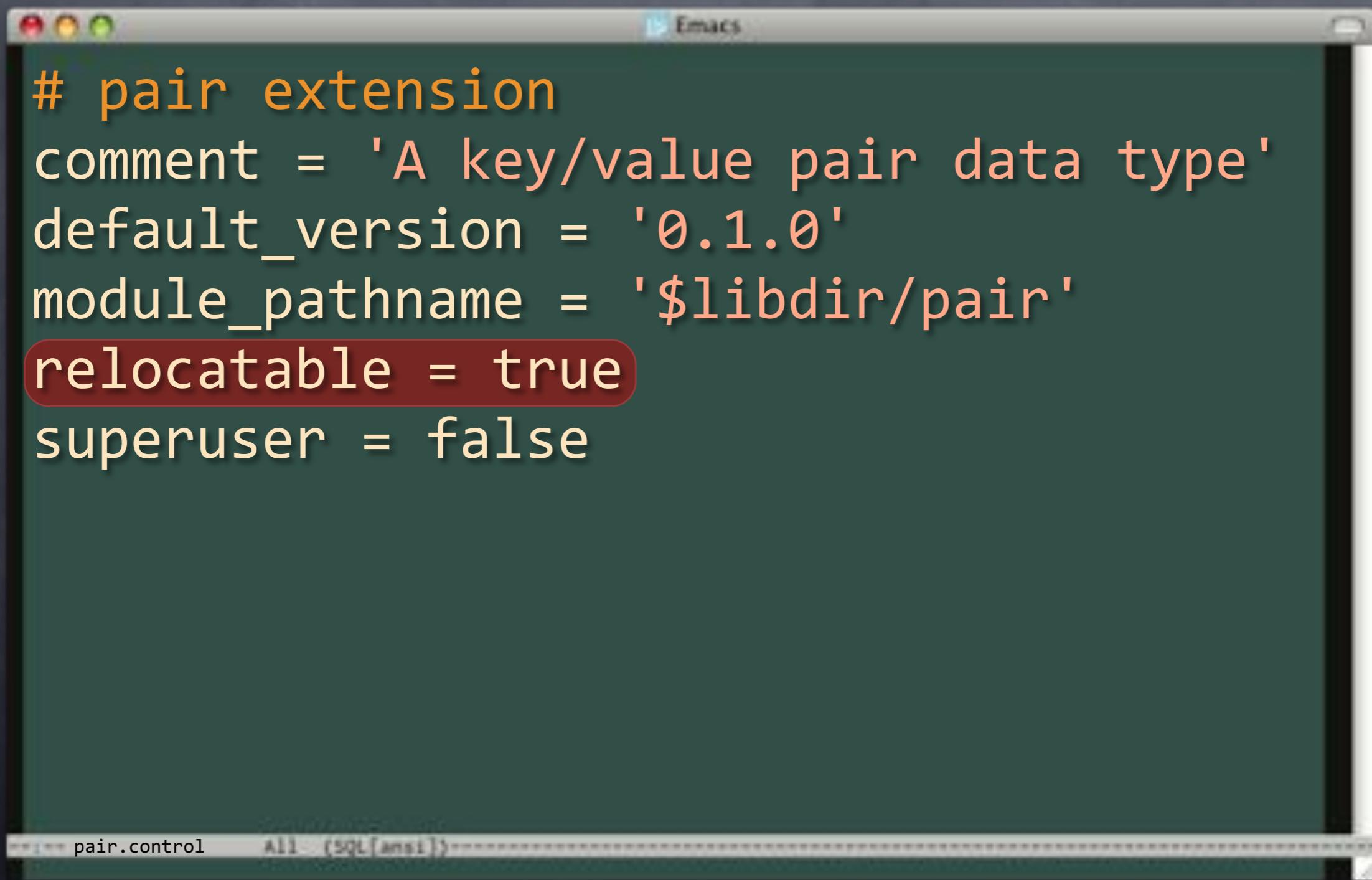
# Create the control file



```
# pair extension
comment = 'A key/value pair
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

For C code

# Create the control file

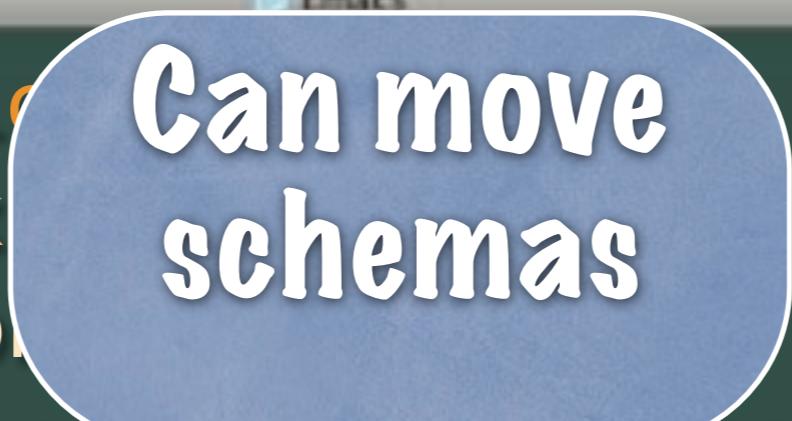


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

The word "relocatable" is highlighted with a red rounded rectangle. The status bar at the bottom shows "pair.control" and "All (SQL [ansi])".

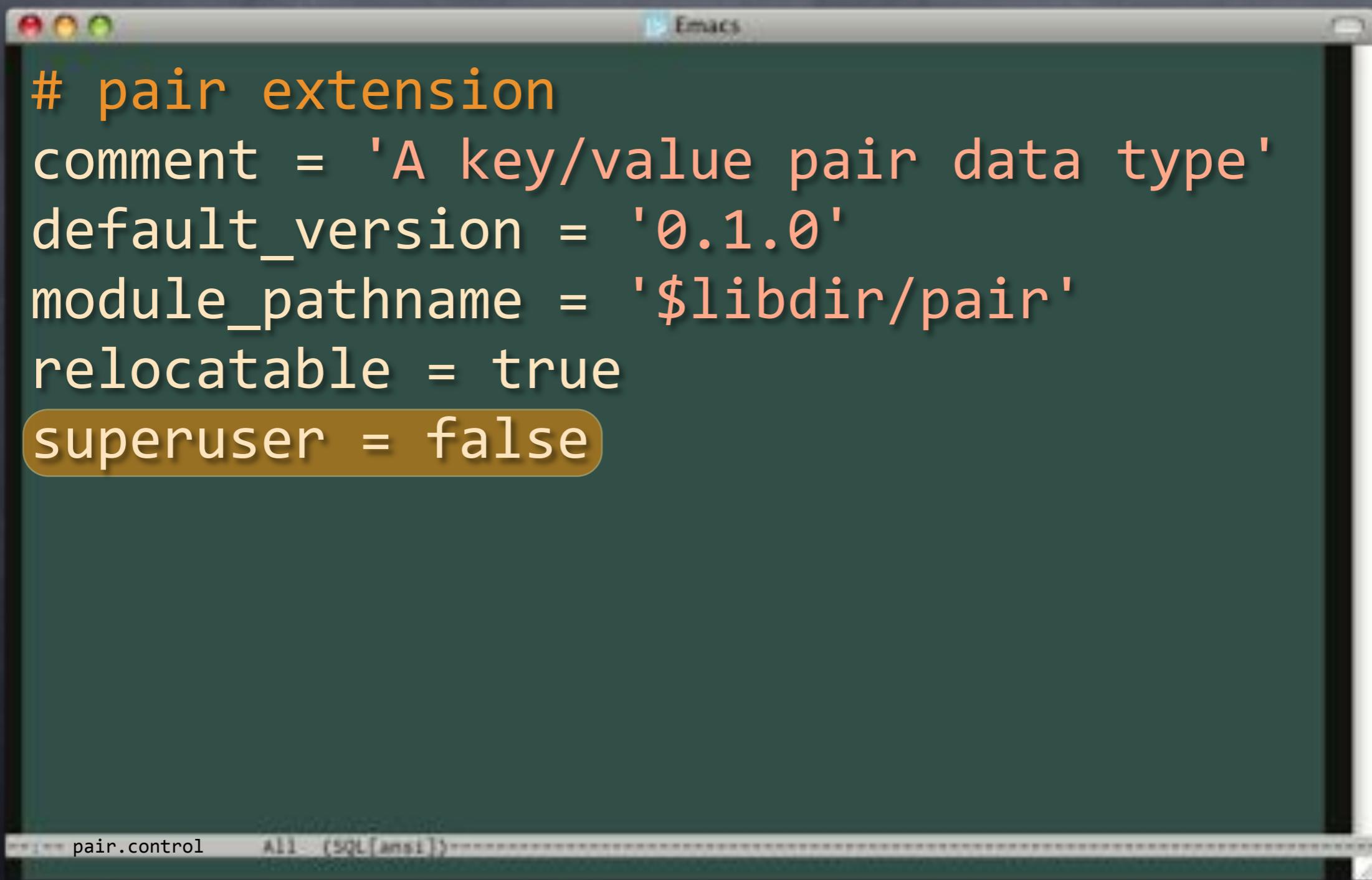
# Create the control file



```
# pair extension
comment = 'A k
default_version = 3
module.pathname = 'pair'
relocatable = true
superuser = false
```

The screenshot shows an Emacs window displaying a configuration file named "pair.control". The file contains several lines of YAML-like syntax. A blue speech bubble with the text "Can move schemas" is overlaid on the screen, pointing towards the "relocatable = true" line. The Emacs window has a dark background and a light green border. The status bar at the bottom shows the file name "pair.control" and the mode "All - (SQL [ansi])".

# Create the control file

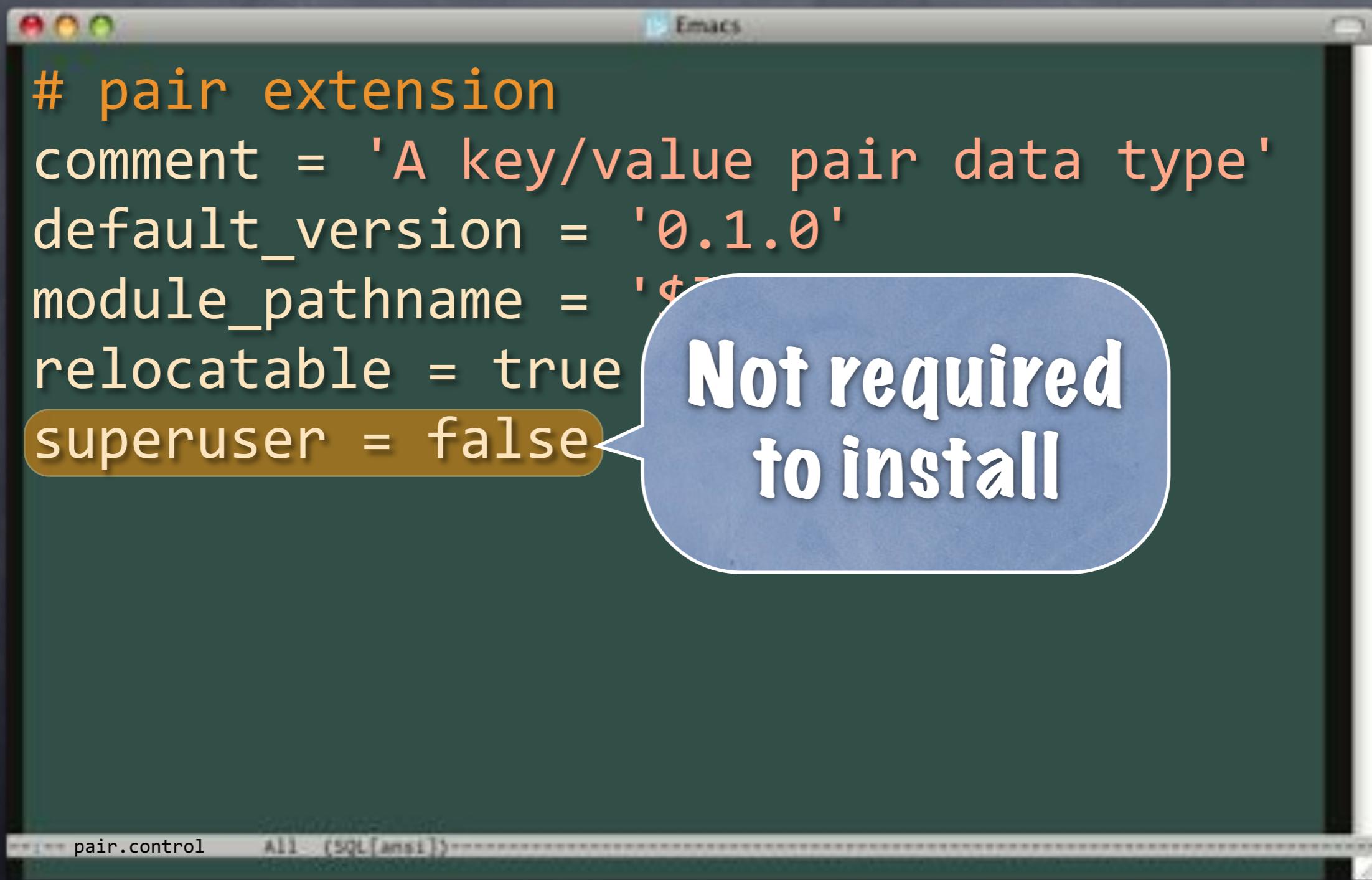


The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

The word "superuser" is highlighted with a yellow-to-orange gradient rectangle. The status bar at the bottom left shows "pair.control" and "All (SQL [ansi])".

# Create the control file



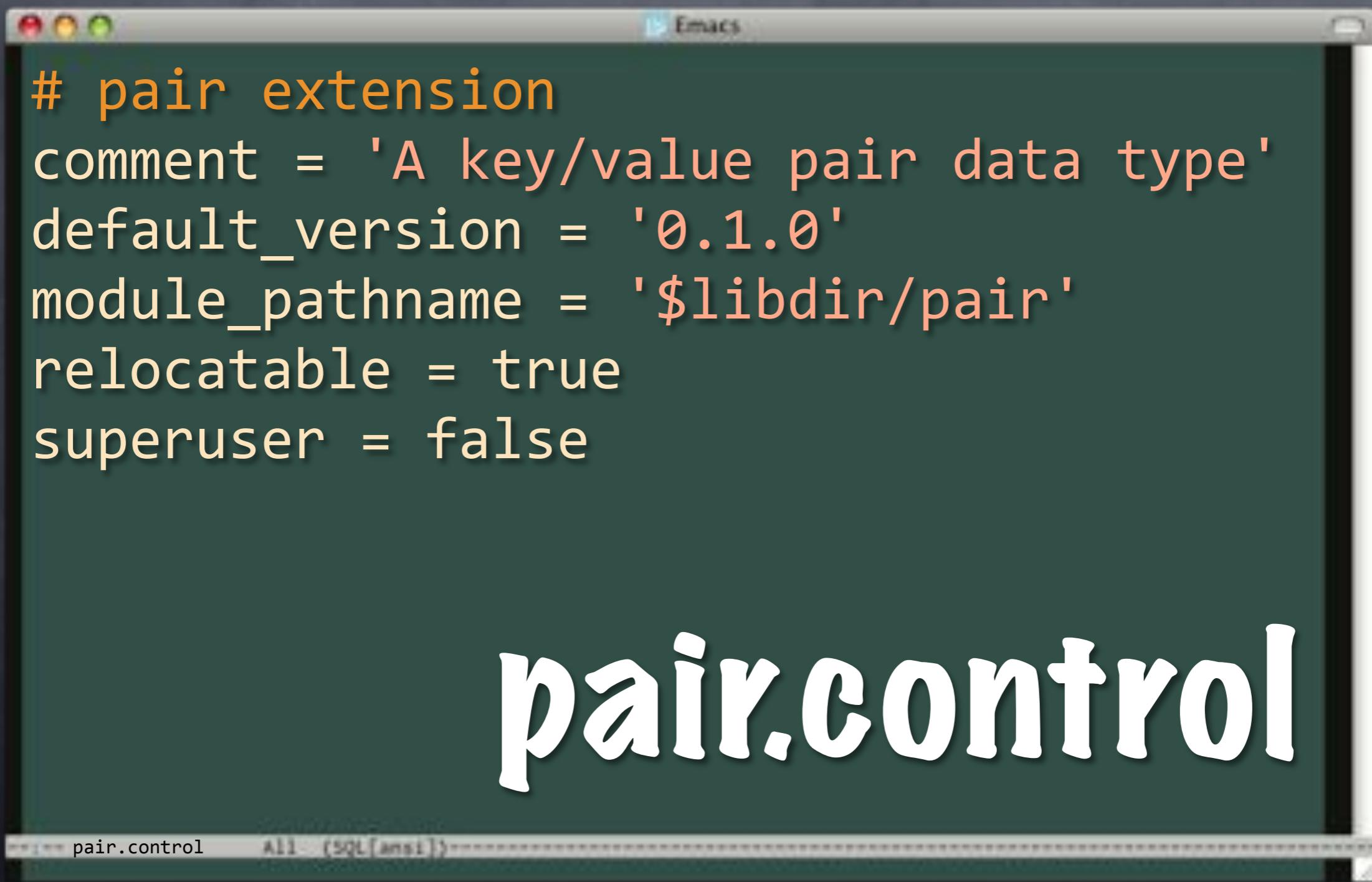
The screenshot shows an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following code:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '$'
relocatable = true
superuser = false
```

A blue callout bubble points from the word "superuser" to the text "Not required to install".

pair.control

# Create the control file



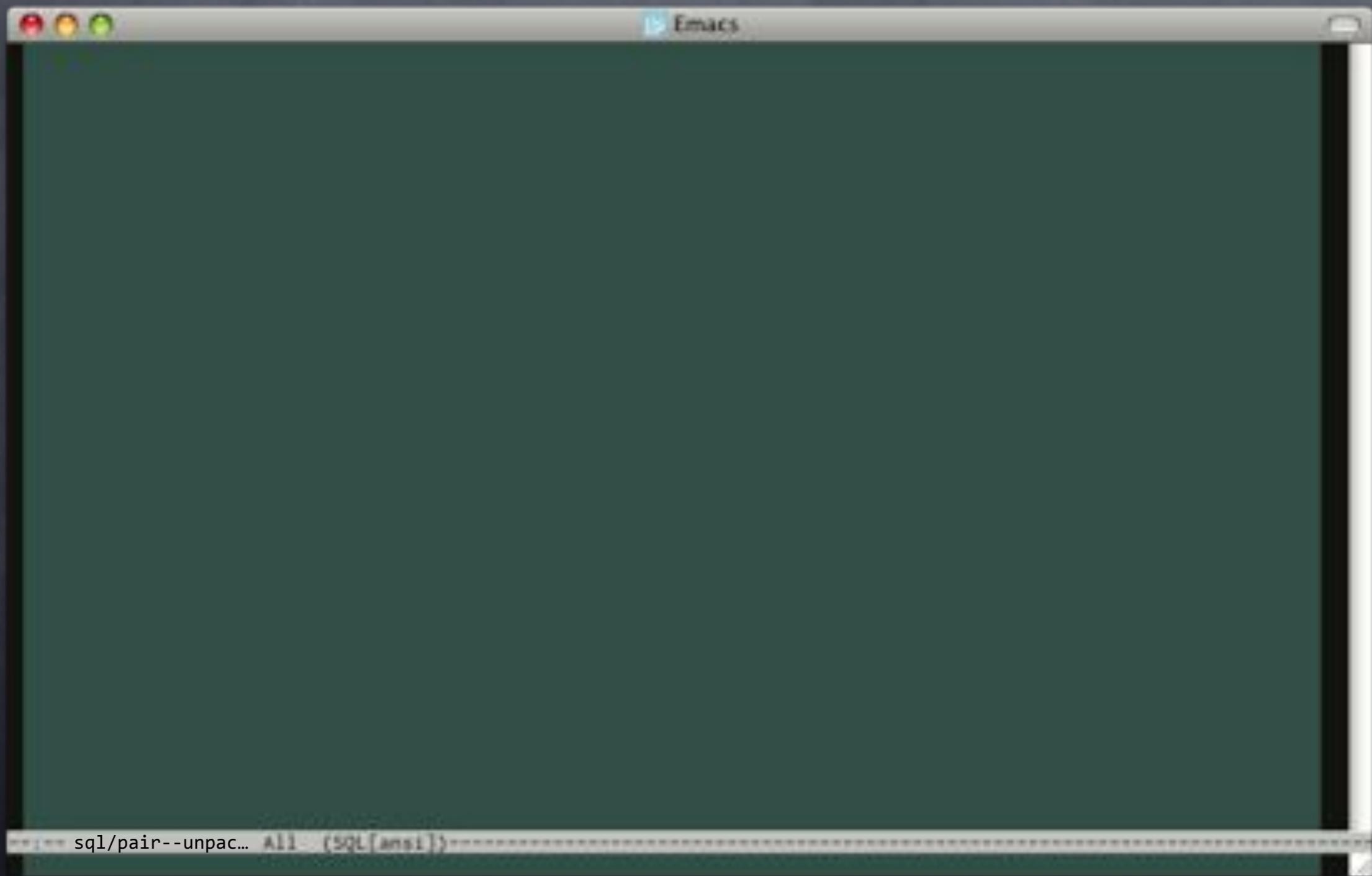
The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module.pathname = '$libdir/pair'
relocatable = true
superuser = false
```

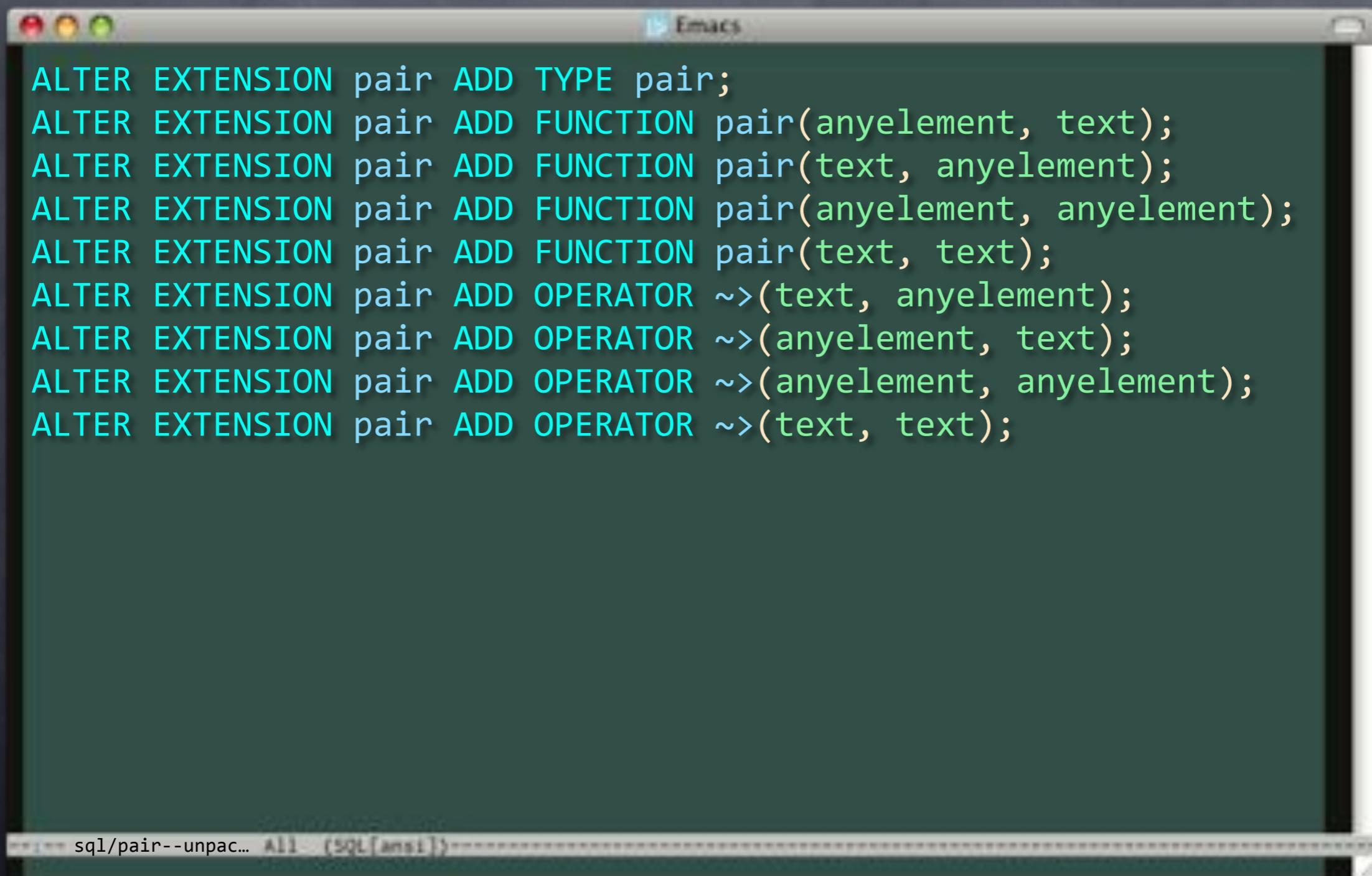
Below the code, the text "pair.control" is displayed in large white letters. At the bottom of the window, the status bar shows "pair.control" and "All - (SQL [ansi])".

pair.control

# Migration from Unpackaged



# Migration from Unpackaged

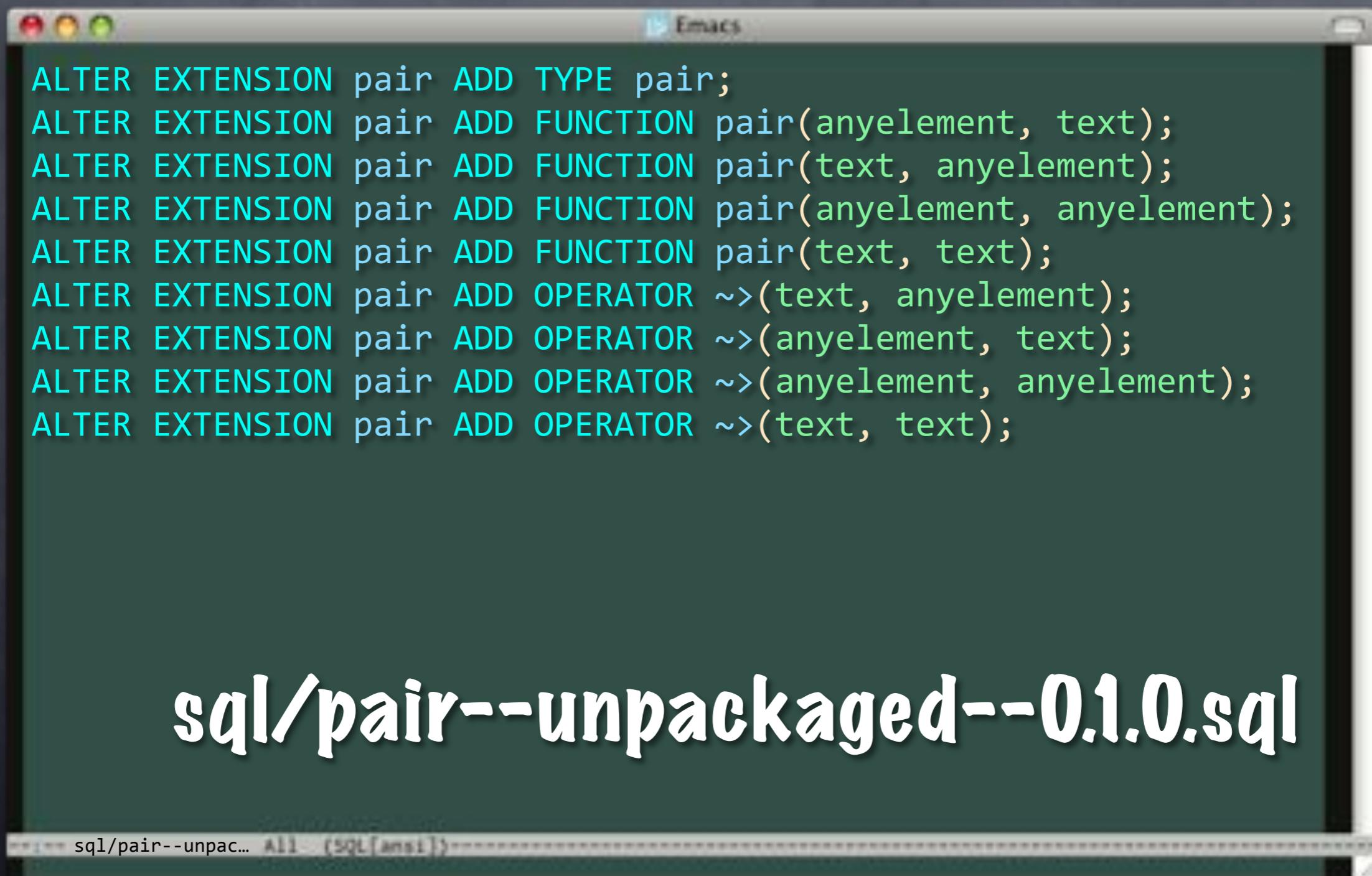


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following SQL code:

```
ALTER EXTENSION pair ADD TYPE pair;
ALTER EXTENSION pair ADD FUNCTION pair(anyelement, text);
ALTER EXTENSION pair ADD FUNCTION pair(text, anyelement);
ALTER EXTENSION pair ADD FUNCTION pair(anyelement, anyelement);
ALTER EXTENSION pair ADD FUNCTION pair(text, text);
ALTER EXTENSION pair ADD OPERATOR ~>(text, anyelement);
ALTER EXTENSION pair ADD OPERATOR ~>(anyelement, text);
ALTER EXTENSION pair ADD OPERATOR ~>(anyelement, anyelement);
ALTER EXTENSION pair ADD OPERATOR ~>(text, text);
```

The status bar at the bottom shows "sql/pair--unpac... All- (SQL [ansi])".

# Migration from Unpackaged

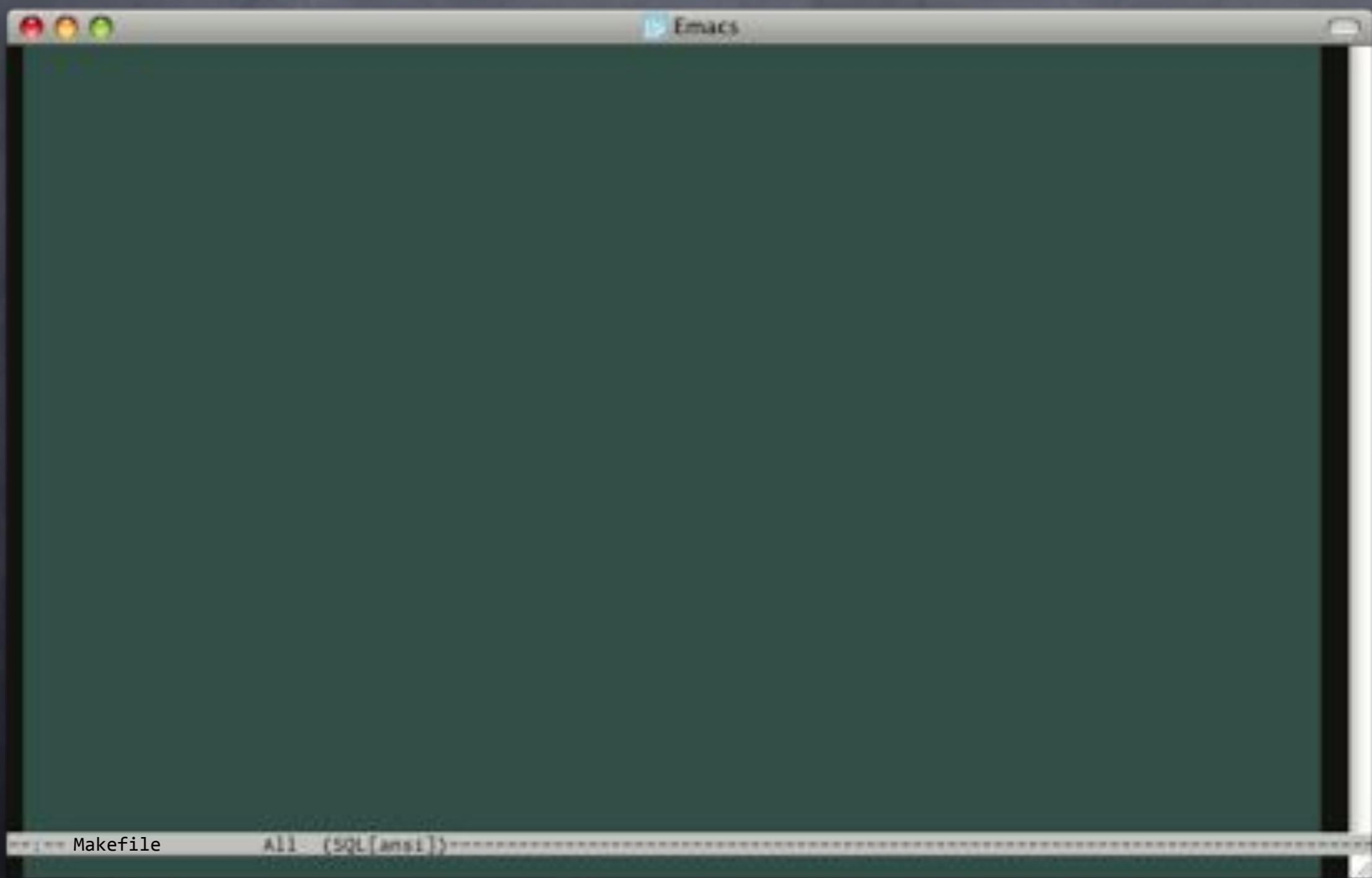


The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains the following SQL code:

```
ALTER EXTENSION pair ADD TYPE pair;
ALTER EXTENSION pair ADD FUNCTION pair(anyelement, text);
ALTER EXTENSION pair ADD FUNCTION pair(text, anyelement);
ALTER EXTENSION pair ADD FUNCTION pair(anyelement, anyelement);
ALTER EXTENSION pair ADD FUNCTION pair(text, text);
ALTER EXTENSION pair ADD OPERATOR ~>(text, anyelement);
ALTER EXTENSION pair ADD OPERATOR ~>(anyelement, text);
ALTER EXTENSION pair ADD OPERATOR ~>(anyelement, anyelement);
ALTER EXTENSION pair ADD OPERATOR ~>(text, text);
```

Below the code, the file name "sql/pair--unpackaged--0.1.0.sql" is displayed in large white font. At the bottom of the window, there is a status bar with the text "sql/pair--unpac... All- (SQL [ansi])".

# Update the Makefile

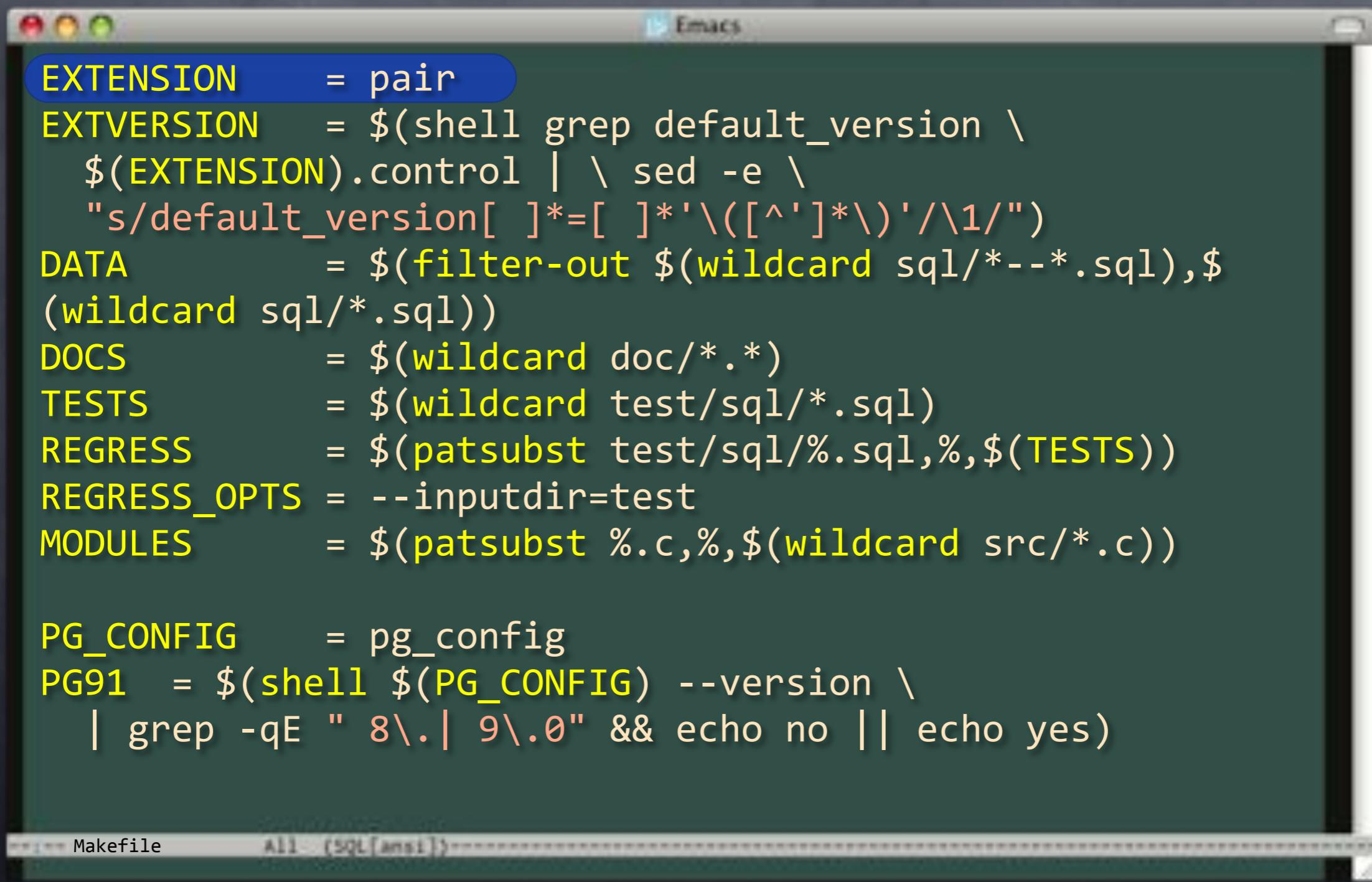


# Update the Makefile

```
EXTENSION      = pair
EXTVERSION     = $(shell grep default_version \
$(EXTENSION).control | \ sed -e \
"s/default_version[ ]*[ ]*[ ]*\([^\"]*\)\)'\/\1/"')
DATA           = $(filter-out $(wildcard sql/*--*.sql),$(
wildcard sql/*.sql))
DOCS           = $(wildcard doc/*.*)
TESTS          = $(wildcard test/sql/*.sql)
REGRESS         = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS   = --inputdir=test
MODULES         = $(patsubst %.c,%,$(wildcard src/*.c))

PG_CONFIG      = pg_config
PG91    = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

# Update the Makefile



The image shows a screenshot of an Emacs window with a dark green background. The title bar says "Emacs". The buffer contains a Makefile with syntax highlighting. A blue oval highlights the word "pair" in the first line. The Makefile defines several variables:

```
EXTENSION      = pair
EXTVERSION     = $(shell grep default_version \
$(EXTENSION).control | \ sed -e \
"s/default_version[ ]*[ ]*[ ]*\([^\"]*\)\)'\\"/\1/"')
DATA           = $(filter-out $(wildcard sql/*---*.sql),$(
wildcard sql/*.sql))
DOCS           = $(wildcard doc/*.*)
TESTS          = $(wildcard test/sql/*.sql)
REGRESS        = $(patsubst test/sql/%.sql%,,$(TESTS))
REGRESS_OPTS   = --inputdir=test
MODULES         = $(patsubst %.c%,,$(wildcard src/*.c))

PG_CONFIG      = pg_config
PG91    = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

The status bar at the bottom shows "Makefile" and "All (SQL [ansi])".

# Update the Makefile

```
EXTENSION      = pair
EXTVERSION     = $(shell grep default_version \
$(EXTENSION).control | \ sed -e \
"s/default_version[ ]*[ ]*'\([^\n]*\)\')' /\1/"")
DATA           = $(filter-out $(wildcard sql/*--*.sql),$(
wildcard sql/*.sql))
DOCS           = $(wildcard doc/*.*)
TESTS          = $(wildcard test/sql/*.sql)
REGRESS         = $(patsubst test/sql/%.sql%,,$(TESTS))
REGRESS_OPTS   = --inputdir=test
MODULES         = $(patsubst %.c%,,$(wildcard src/*.c))

PG_CONFIG       = pg_config
PG91    = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

# Update the Makefile



```
Emacs  
Makefile All- (SQL [ansi])  
  
EXTENSION      = pair  
EXTVERSION     = $(shell grep default_version \  
$(EXTENSION).control | \ sed -e \  
"s/default_version[ ]*[ ]*\([^\"]*\)\)/\1/")  
DATA           = $(filter-out $(wildcard sql/*--*.sql), $  
(wildcard sql/*.sql))  
DOCS           = $(wildcard doc/*.*)  
TESTS          = $(wildcard test/sql/*.sql)  
REGRESS        = $(patsubst test/sql/%.sql,%,$(TESTS))  
REGRESS_OPTS   = --inputdir=test  
MODULES        = $(patsubst %.c,%,$(wildcard src/*.c))  
  
PG_CONFIG      = pg_config  
PG91    = $(shell $(PG_CONFIG) --version \  
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

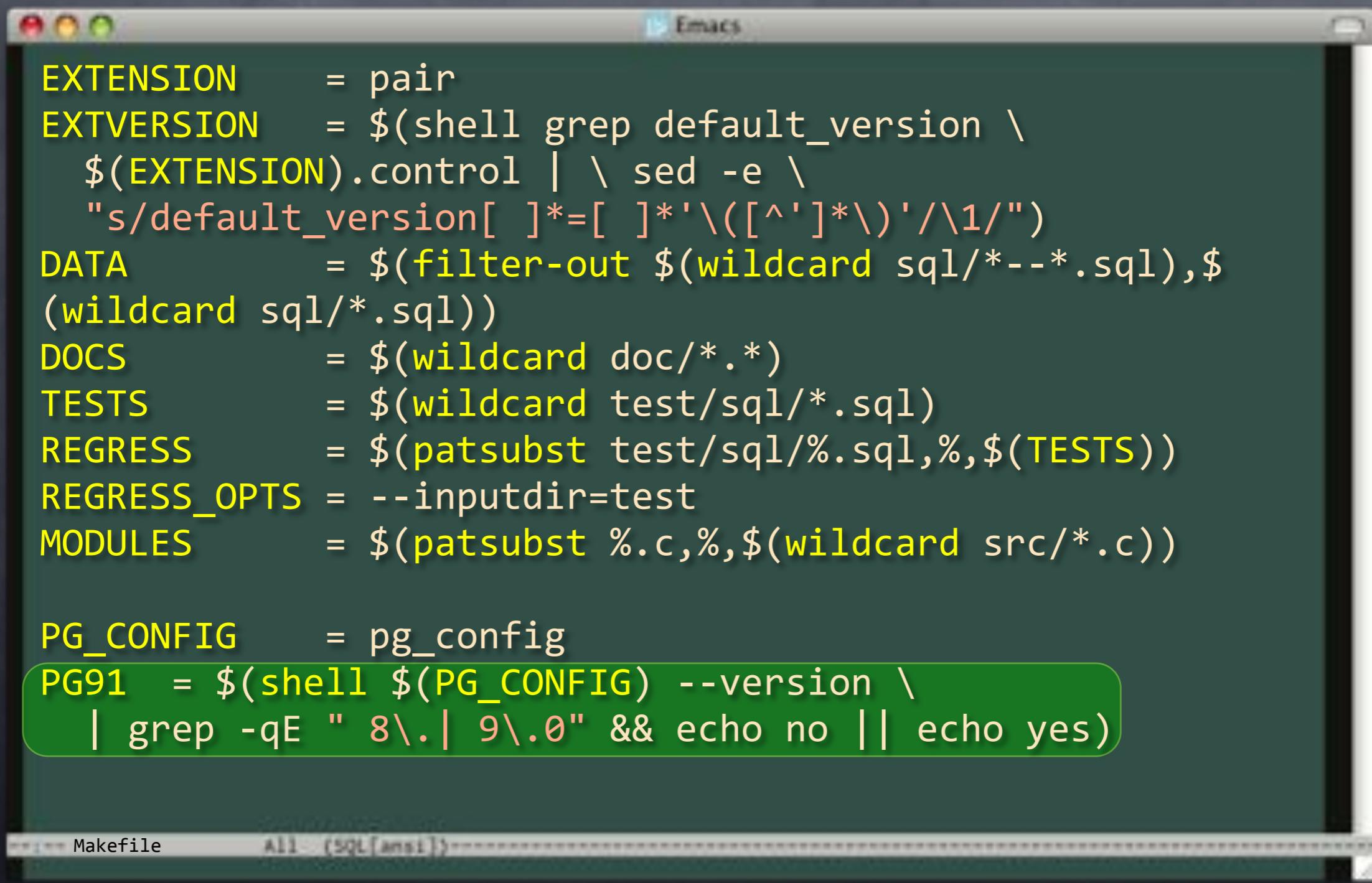
Extract from  
control file

# Update the Makefile

```
EXTENSION      = pair
EXTVERSION     = $(shell grep default_version \
$(EXTENSION).control | \ sed -e \
"s/default_version[ ]*[ ]*=[ ]*'\([^\']*\)\')'/\1/"')
DATA           = $(filter-out $(wildcard sql/*--*.sql),$(
wildcard sql/*.sql))
DOCS           = $(wildcard doc/*.*)
TESTS          = $(wildcard test/sql/*.sql)
REGRESS         = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS   = --inputdir=test
MODULES         = $(patsubst %.c,%,$(wildcard src/*.c))

PG_CONFIG      = pg_config
PG91    = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

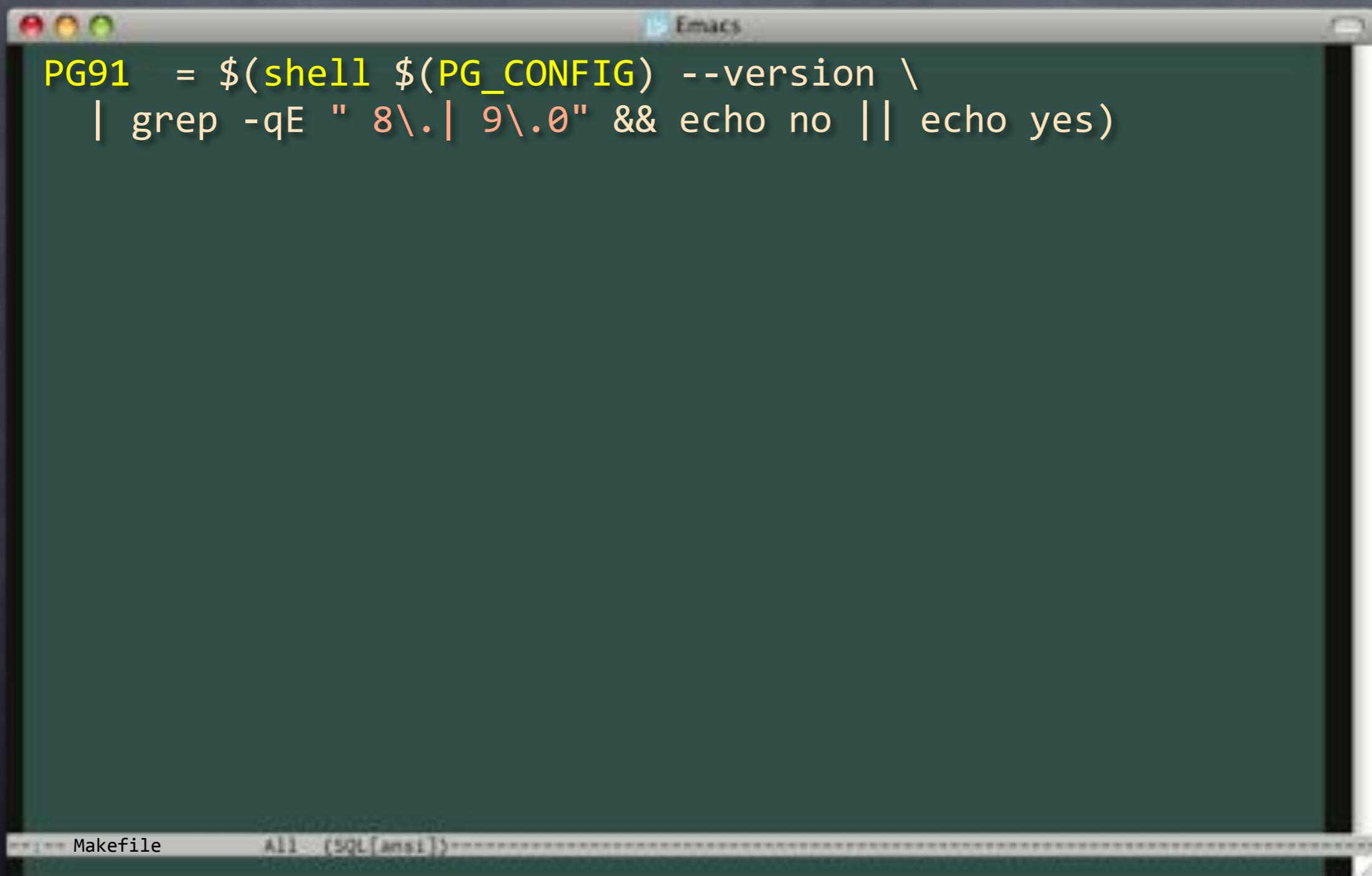
# Update the Makefile



```
EXTENSION      = pair
EXTVERSION     = $(shell grep default_version \
$(EXTENSION).control | \ sed -e \
"s/default_version[ ]*[ ]*=[ ]*'\([^\']*'\)'\1/"')
DATA           = $(filter-out $(wildcard sql/*--*.sql),$(
wildcard sql/*.sql))
DOCS           = $(wildcard doc/*.*)
TESTS          = $(wildcard test/sql/*.sql)
REGRESS         = $(patsubst test/sql/%.sql,%,$(TESTS))
REGRESS_OPTS   = --inputdir=test
MODULES         = $(patsubst %.c,%,$(wildcard src/*.c))

PG_CONFIG      = pg_config
PG91    = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

# Update the Makefile



The image shows a screenshot of an Emacs window with a dark green background. The title bar reads "Emacs". The buffer contains the following text:

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\.0" && echo no || echo yes)
```

The status bar at the bottom left shows "Makefile" and "All- (SQL [ansi])".

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/${EXTENSION}--${EXTVERSION}.sql

sql/${EXTENSION}--${EXTVERSION}.sql: sql/${EXTENSION}.sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/${EXTENSION}--$(
EXTVERSION).sql
EXTRA_CLEAN = sql/${EXTENSION}--${EXTVERSION}.sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

# Update the Makefile

```
PG91 = $(shell $(PG_CONFIG) --version \
| grep -qE " 8\.\| 9\.\." && echo no || echo yes)

ifeq ($(PG91),yes)
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: sql/$(EXTENSION).sql
    cp $< $@

DATA = $(wildcard sql/*--*.sql) sql/$(EXTENSION)--$(
(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
endif

PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
```

Or Forget It

# Or Forget It

- ➊ Copy Makefile

# Or Forget It

- Copy Makefile
- Edit first line

# Or Forget It

- Copy Makefile
- Edit first line
  - EXTENSION=pair

# Or Forget It

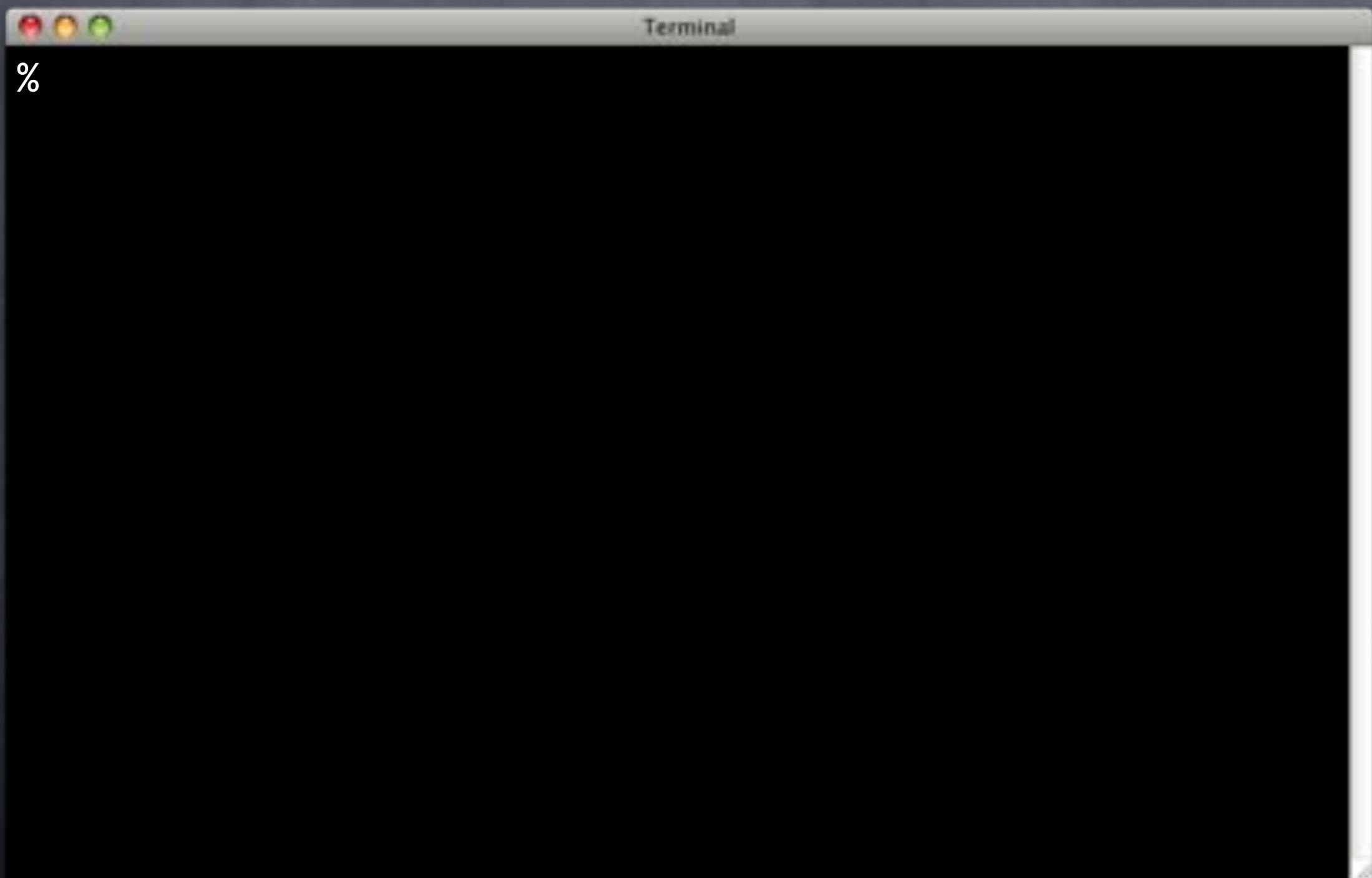
- Copy Makefile
- Edit first line
  - EXTENSION=pair
- Ignore the rest

# Or Forget It

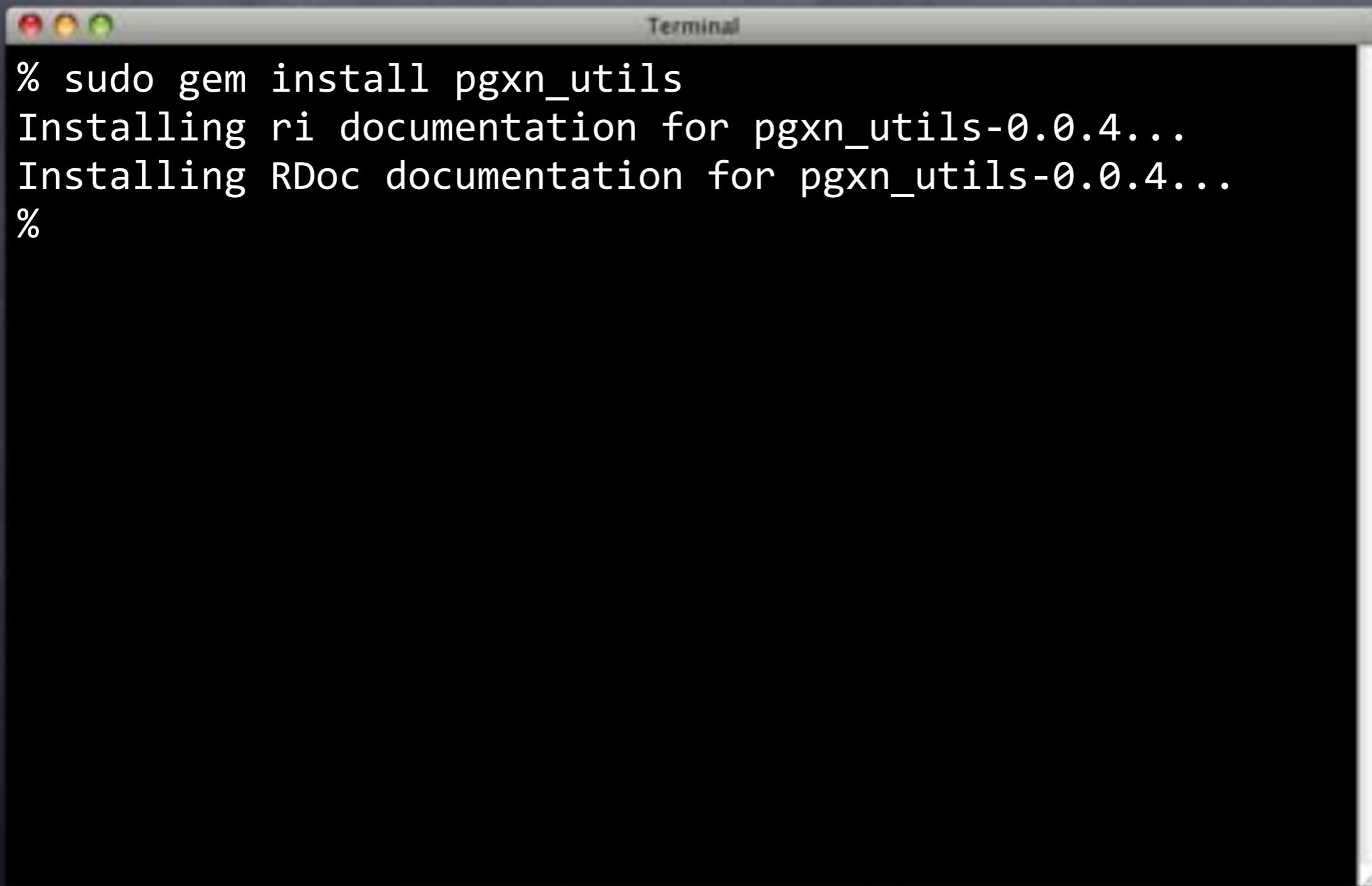
- Copy Makefile
- Edit first line
  - EXTENSION=pair
- Ignore the rest

Better still...

# Skeleton in the Closet

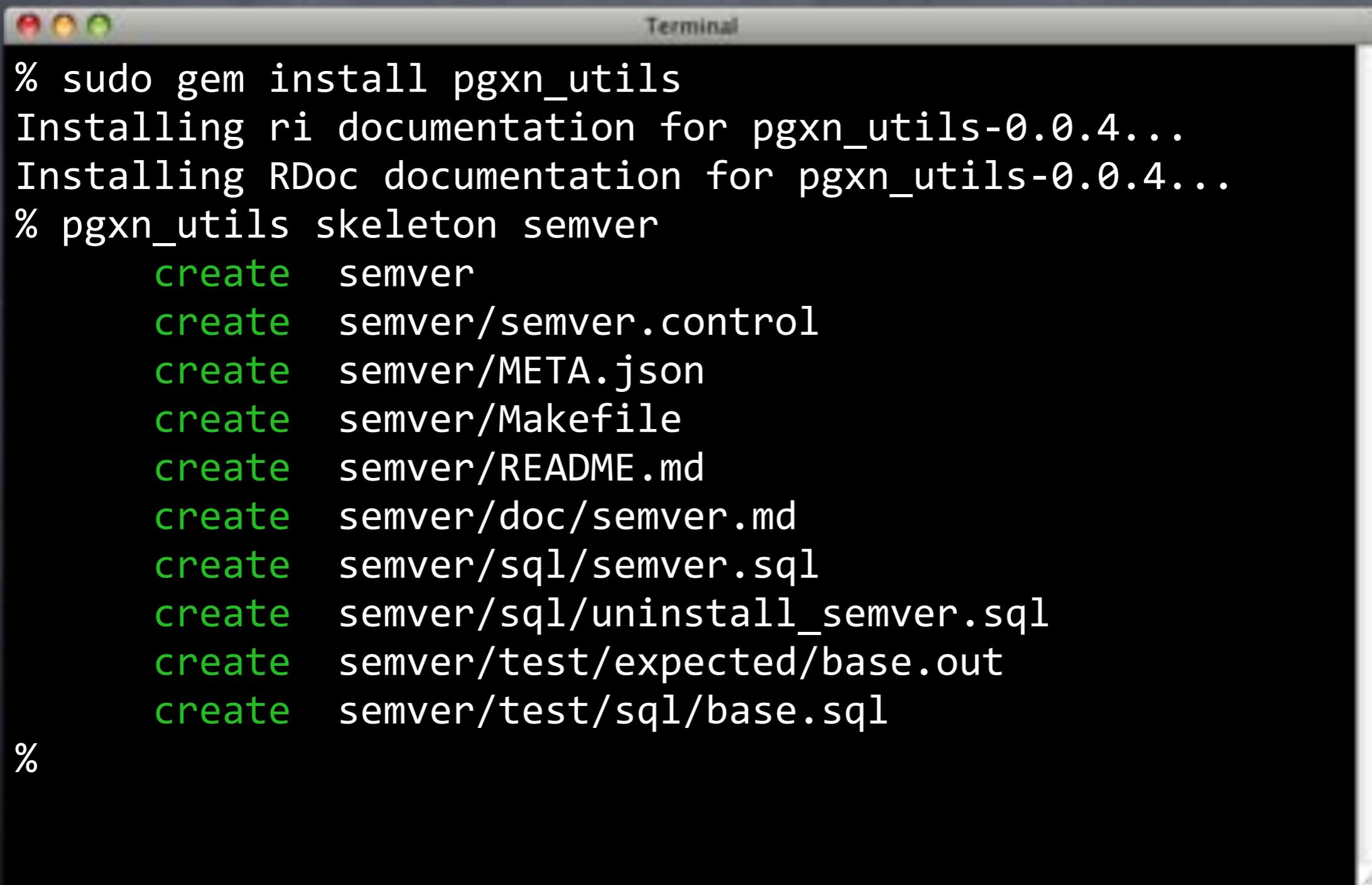


# Skeleton in the Closet



```
% sudo gem install pgxn_utils
Installing ri documentation for pgxn_utils-0.0.4...
Installing RDoc documentation for pgxn_utils-0.0.4...
%
```

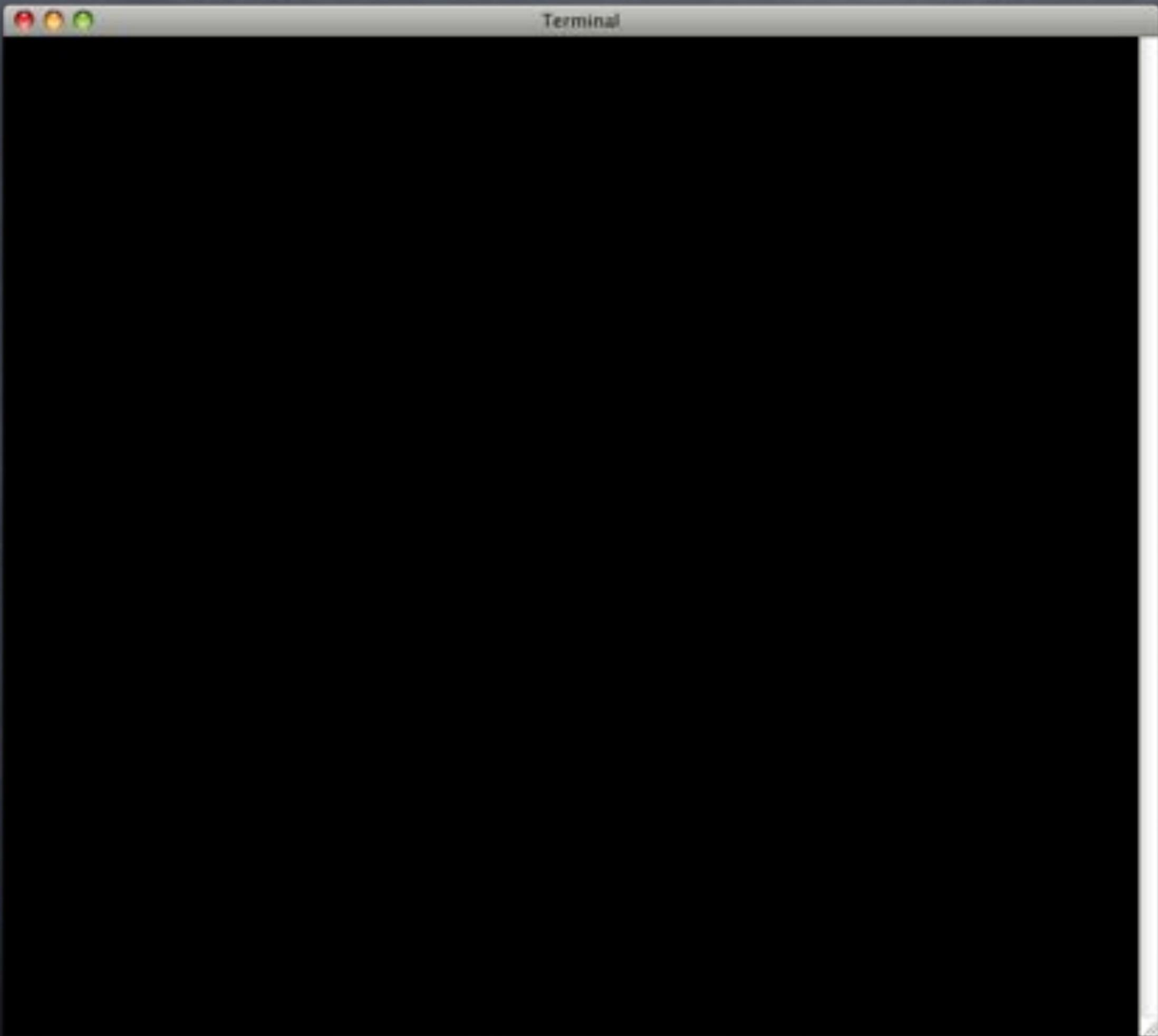
# Skeleton in the Closet

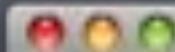


A screenshot of a Mac OS X Terminal window titled "Terminal". The window contains the following text:

```
% sudo gem install pgxn_utils
Installing ri documentation for pgxn_utils-0.0.4...
Installing RDoc documentation for pgxn_utils-0.0.4...
% pgxn_utils skeleton semver
  create  semver
  create  semver/semver.control
  create  semver/META.json
  create  semver/Makefile
  create  semver/README.md
  create  semver/doc/semver.md
  create  semver/sql/semver.sql
  create  semver/sql/uninstall_semver.sql
  create  semver/test/expected/base.out
  create  semver/test/sql/base.sql
%
```

Thank you  
Dickson S. Guedes





```
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
%
```



```
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
%
```



```
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
% make  
cp sql/pair.sql sql/pair--0.1.0.sql  
% sudo make install  
# ...elided  
%
```

```
Terminal  
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
% make  
cp sql/pair.sql sql/pair--0.1.0.sql  
% sudo make install  
# ...elided  
% psql try  
psql (9.1devel)  
Type "help" for help.
```

```
try=# create extension pair;  
CREATE EXTENSION  
try=#
```

```
Terminal  
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
% make  
cp sql/pair.sql sql/pair--0.1.0.sql  
% sudo make install  
# ...elided  
% psql try  
psql (9.1devel)  
Type "help" for help.
```

```
try=# create extension pair;  
CREATE EXTENSION  
try=#
```

```
Terminal  
% curl -O http://api.pgxn.org/dist/pair/0.1.2/pair-0.1.2.zip  
% unzip pair-0.1.2.zip  
Archive: pair-0.1.2.zip  
% cd pair-0.1.2  
% make  
cp sql/pair.sql sql/pair--0.1.0.sql  
% sudo make install  
# ...elided  
% psql try  
psql (9.1devel)  
Type "help" for help.  
  
try=# create extension pair;  
CREATE EXTENSION  
try=# \dT  
      List of data types  
 Schema | Name | Description  
-----+-----+-----  
 public | pair |  
(1 row)  
  
try=#[
```

Wow!



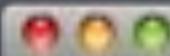
Terminal

%



```
% sudo easy_install pgxnclient
Installing pgxncli.py script to /usr/local/bin
Installing pgxn script to /usr/local/bin
Processing dependencies for pgxnclient
Finished processing dependencies for pgxnclient
%
```

```
Terminal  
% sudo easy_install pgxnclient  
Installing pgxncli.py script to /usr/local/bin  
Installing pgxn script to /usr/local/bin  
Processing dependencies for pgxnclient  
Finished processing dependencies for pgxnclient  
% pgxn install pair  
INFO: best version: pair 0.1.3  
INFO: saving /tmp/tmpB3eZEr/pair-0.1.3.zip  
INFO: unpacking: /tmp/tmpB3eZEr/pair-0.1.3.zip  
INFO: building extension  
INFO: installing extension  
%
```



```
% sudo easy_install pgxnclient
Installing pgxncli.py script to /usr/local/bin
Installing pgxn script to /usr/local/bin
Processing dependencies for pgxnclient
Finished processing dependencies for pgxnclient
% pgxn install pair
INFO: best version: pair 0.1.3
INFO: saving /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: unpacking: /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: building extension
INFO: installing extension
%
```



```
% sudo easy_install pgxnclient
Installing pgxncli.py script to /usr/local/bin
Installing pgxn script to /usr/local/bin
Processing dependencies for pgxnclient
Finished processing dependencies for pgxnclient
% pgxn install pair
INFO: best version: pair 0.1.3
INFO: saving /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: unpacking: /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: building extension
INFO: installing extension
% pgxn load pair -d try
INFO: best version: pair 0.1.3
CREATE EXTENSION
%
```

```
% sudo easy_install pgxnclient
Installing pgxncli.py script to /usr/local/bin
Installing pgxn script to /usr/local/bin
Processing dependencies for pgxnclient
Finished processing dependencies for pgxnclient
% pgxn install pair
INFO: best version: pair 0.1.3
INFO: saving /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: unpacking: /tmp/tmpB3eZEr/pair-0.1.3.zip
INFO: building extension
INFO: installing extension
% pgxn load pair -d try
INFO: best version: pair 0.1.3
CREATE EXTENSION
%
```

Yes!

# Client Plans

# Client Plans

- Git-like dispatch

# Client Plans

- Git-like dispatch
- `pgxn foo => pgxn-foo`

# Client Plans

- Git-like dispatch
  - pgxn foo => pgxn-foo
- pgxn\_utils adapted

# Client Plans

- Git-like dispatch
  - `pgxn foo => pgxn-foo`
- `pgxn_utils` adapted
- `pgxn skeleton semver`

# Client Plans

- Git-like dispatch
  - pgxn foo => pgxn-foo
- pgxn\_utils adapted
  - pgxn skeleton semver
- META.json validator under development

# Client Plans

- Git-like dispatch
  - pgxn foo => pgxn-foo
- pgxn\_utils adapted
  - pgxn skeleton semver
- META.json validator under development
  - pgxn validate-meta

# Client Plans

- Git-like dispatch
  - pgxn foo => pgxn-foo
- pgxn\_utils adapted
  - pgxn skeleton semver
- META.json validator under development
  - pgxn validate-meta
- New dispatcher next week

# Client Plans

- Git-like dispatch
  - pgxn foo => pgxn-foo
- pgxn\_utils adapted
  - pgxn skeleton semver
- META.json validator under development
  - pgxn validate-meta
- New dispatcher next week
- Write some utilities!

Thank you  
Daniele Varrazzo

# Resources

# Resources

- Site: <http://pgxn.org/>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>
- Release: <http://manager.pgxn.org/>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>
- Release: <http://manager.pgxn.org/>
- API: <http://github.com/pgxn/pgxn-api/wiki/>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>
- Release: <http://manager.pgxn.org/>
- API: <http://github.com/pgxn/pgxn-api/wiki/>
- Group: <http://groups.google.com/group/pgxn-users>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>
- Release: <http://manager.pgxn.org/>
- API: <http://github.com/pgxn/pgxn-api/wiki/>
- Group: <http://groups.google.com/group/pgxn-users>
- Donate: <http://fundraising.pgxn.org/>

# Resources

- Site: <http://pgxn.org/>
- Twitter: <http://twitter.com/pgxn>
- Blog: <http://blog.pgxn.org/>
- Release: <http://manager.pgxn.org/>
- API: <http://github.com/pgxn/pgxn-api/wiki/>
- Group: <http://groups.google.com/group/pgxn-users>
- Donate: <http://fundraising.pgxn.org/>



# Thank you.

## Releasing Extensions on PGXN

David E. Wheeler  
PostgreSQL Experts, Inc.



Text: Attribution-Noncommercial-Share Alike 3.0 United States:  
<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>  
Images licensed independently and © Their respective owners.

