Releasing Extensions on PGXN

David E. Wheeler
PostgreSQL Experts, Inc.

PGCon
May 20, 2011
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

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

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...

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...

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.

Monday, May 23, 2011
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
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:

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

Output:

```
node_type | strategy | actual_startup_time | actual_total_time
--- | --- | --- | ---
Index Scan | | 00:00:00.000017 | 00:00:00.000017
```
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:

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

Output:

```
node_type | strategy | actual_start_time | actual_total_time
--------- | -------- | ----------------- | ------------------
Index Scan |         | 00:00:00.000017   | 00:00:00.000017
```
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
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: trinity
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:
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
David E. Wheeler

Nickname: theory
Email: davido@justatheory.com
URL: http://justatheory.com/
Twitter: theory

Distributions
- explanation 0.2.0  Turn an explain plan into a table of nodes organized as a proximity tree 2011-02-21
- pair 0.1.2  A key/value pair data type 2011-04-20
- pgTAP 0.23.0  Unit testing for PostgreSQL 2011-02-02
- semver 0.2.1  A semantic version data type 2011-04-20
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

```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"
  }
}
```
```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"
  }
}
```
```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"
  }
}
```
```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"
    }
}
```
```
{
    "name": "pair",
    "abstract": "A key/value data 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"
    }
}
```
```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"
    }
}
```
```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"
    }
}
```
```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"
  }
}
```
```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

```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"
    }
}
```

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
%

Monday, May 23, 2011
Package it Up!

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

Easy, eh?
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.
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.
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?
URL: 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?
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
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
Thanks for requesting a PGXN account, tomlane. We'll get back to you once the hangover has worn off.
From: PGXN Admin
Subject: Welcome to PGXN!
Date: October 29, 2010 1:56:21 PM PDT
To: tomlane <dgj@postgresql.org>

What up, tomlane.

Your PGXN account request has been approved. Ready to get started? Great! Just click this link to set your password and get going:

https://manager.pgxn.org/account/reset/VskGiJ

Best,

PGXN Management
From: PGXN Admin
Subject: Welcome to PGXN!
Date: October 29, 2010 1:56:21 PM PDT
To: tomlane@dgi@postgresql.org

What up, tomlane.

Your PGXN account request has been approved. Ready to get started?
Great! Just click this link to set your password and get going:

https://manager.pgxn.org/account/reset/key03

Best,
PGXN Management
Reset Your PGXN Password

Please choose a password to use for your PGXN account.

Change Password

New Password:

Verify Password:

Change
Reset Your PGXN Password

Please choose a password to use for your PGXN account.

Change Password

New Password: omg WTF ROTFL lolz
Verify Password: omg WTF ROTFL lolz

Change
Password Changed

WOOT! Your password has been changed. So what are you waiting for? Go log in!
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.
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

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: Choose File no file selected

Release it!
Upload a Distribution

So you've developed a PGXN extension and want 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.

Uploading and release your extension:

- Choose your distribution archive (e.g., pair-0.1.0.zip)
- Click on "Release It!"

PGXN Manager v0.4.4. © 2010 David E. Wheeler. Distributed under the PostgreSQL License.
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: Choose File pair-0.1.0.zip

Release It!
pair-0.1.0
A key/value pair data type

Congratulations! This distribution has been released on PGXN.

Owner: tomlane
Status: stable
SHA1: 1a6ae0d4af6db420ed6c2e3fb338c15c9f5b8689
Extensions: * 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:
No resources, tags, or long 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'.

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:
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 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://justattheory.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:

```bash
make
make installcheck
make install
```

If you encounter an error such as:
Add README Extension
Add README Extension

% git mv README.pair README.md
%

Monday, May 23, 2011
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
- Textile
- Trac
- MediaWiki
- Text::Markup (fork me!)
- Write some Docs!
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/
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`.

Monday, May 23, 2011
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`. 
"provides": {
    "pair": {
        "file": "pair.sql",
        "version": "0.1.0"
    }
},
"meta-spec": {
    "version": "1.0.0",
    "url": "http://pgxn.org/meta/spec.txt"
}
"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"
}
}
"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"
}
}
"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"
}
Add Tags

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

{"tags": ["ordered pair", "pair", "key value"]}
Add Resources

```json
"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/"
  }
}
```
Update Version

```json
{
  "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

```json
{
    "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

Must be unique.

```json
{
"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",
"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",
  "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"
    }
  }
}
```
PGXN Meta Spec

http://pgxn.org/spec/
Package it up

%
Package it up

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

Monday, May 23, 2011
Release it
pair: A key/value pair data type / PostgreSQL Extension Network

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.
Lots of great metadata!
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.
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 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 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".

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 blog post. Give it a read.
Niiiiice.

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”.

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 blog post. Give it a read...
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

% mkdir test
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

```bash
% mkdir test
% git mv sql test/
% git mv expected test/
% mkdir sql
% git mv *.sql sql/
```
Update Path

```
{
    "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 Path

```json
{
  "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"
    }
  }
}
```
```
{
    "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
DATA      = $(wildcard sql/\*\.sql)
DOCS      = $(wildcard doc/\*.*
MODULES   = $(patsubst %.c,\%,$(wildcard src/\*\.c))
Makefile

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

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

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
DATA = $(wildcard inputdir/*.sql)
DOCS = $(wildcard inputdir/*.sql)
MODULES = $(patsubst test/sql/%.sql,src/$(shell sed -n 1p inputdir/*.sql))

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

Makefile

inputdir
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)
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)
included $(PGXS)
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)
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
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '${libdir}/pair'
relocatable = true
superuser = false
Create the control file

```python
# pair extension
comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '${libdir/pair'}
relocatable = true
superuser = false
```
Create the control file

```python
# pair extension

comment = 'A key/value pair data type'
default_version = '0.1.0'
module_pathname = '${libdir}/pair'
relocatable = true
superuser = false
```
Create the control file

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

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

```plaintext
# pair extension
comment = 'A key/value pair
default_version = '0.1.0'
module_pathname = '$libdir/pair'
relocatable = true
superuser = false
```
Create the control file

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

Can move schemas

```c
# pair extension
comment = 'A key data type'
default_version = $install/pair'
module_pathname = $install/pair'
relocatable = true
superuser = false
```
Create the control file

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

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

Not required to install
Create the control file

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

pair.control
Migration from Unpackaged
Migration from Unpackaged

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);
Migration from Unpackaged

```
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);
```

sql/pair--unpackaged--0.1.0.sql
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),\n(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\.*" && 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

```
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 = $(patsubst \%\*\%,$(wildcard src/\*\*\c))

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

```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\.." && echo no || echo yes)
```

Extract from control file
Update the Makefile

EXTENSION = pair
EXTVERSION = $(shell grep default_version \n    $(EXTENSION).control | \ sed -e \n    "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
MODULRES = $(patsubst %.c,%,$(wildcard src/*.c))

PG_CONFIG = pg_config
PG91 = $(shell $(PG_CONFIG) --version \n    | grep -qE " 8\.| 9\.." && echo no || echo yes)
Update the Makefile

EXTENSION = pair
EXTVERSION = $(shell grep default_version \n  $(EXTENSION).control | \ sed -e \n  "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 \n  | grep -qE " 8\.| 9\./" && echo no || echo yes)
Update the Makefile

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

```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

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

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

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

```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

- 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

% 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
%

Monday, May 23, 2011
Thank you
Dickson S. Guedes
% curl -O http://api.pgxnx.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.pgxnx.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
%
% curl -O http://api.pgxnx.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=#
% 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=#
% 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=#
% 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
%
% 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
%

Monday, May 23, 2011
% 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 %
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.