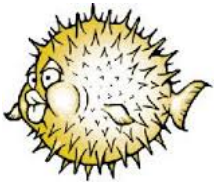
SqlPorts, an adventure in sqlite

Marc Espie <espie@openbsd.org>, <espie@lse.epita.fr>



March 12, 2019

Some background

- Roughly 10000 ports in OpenBSD
- a lot of (mostly identical stuff)
- How to achieve infrastructure changes

Typical port

```
COMMENT =       command-line utility for POSIX tar(1) archive files

DISTNAME =      minitar-0.8
CATEGORIES =    archivers

HOMEPAGE =      http://www.github.com/atoulme/minitar
MAINTAINER =    Sebastian Reitenbach <sebastia@openbsd.org>

# GPLv2/Ruby license
PERMIT_PACKAGE_CDROM =          Yes

MODULES =               lang/ruby

CONFIGURE_STYLE =       ruby gem

.include <bsd.port.mk>
```

Reflection

```
$ make show=MODULES
lang/ruby
hub$ make show=PERMIT_DISTFILES_FTP
Yes
hub$ make show=CFLAGS
-O2 -pipe
```

Better Reflection

```
archivers/ruby-minitar,ruby25.BUILD_DEPENDS=lang/ruby/2.5
archivers/ruby-minitar,ruby25.IS_INTERACTIVE=No
archivers/ruby-minitar,ruby25.SUBPACKAGE=-
archivers/ruby-minitar,ruby25.FLAVOR=ruby25
archivers/ruby-minitar,ruby25.BUILD_PACKAGES= -
archivers/ruby-minitar,ruby25.MULTI_PACKAGES=-
archivers/ruby-minitar,ruby25.DISTFILES=minitar-0.8.gem
archivers/ruby-minitar,ruby25.MASTER_SITES=https://rubygems.org/downloads/
archivers/ruby-minitar,ruby25.CHECKSUM_FILE=/usr/ports/archivers/ruby-minitar/distinfo
archivers/ruby-minitar,ruby25.FETCH_MANUALLY=No
archivers/ruby-minitar,ruby25.PERMIT_DISTFILES_FTP=Yes
archivers/ruby-minitar,ruby25.NO_TEST=Yes
archivers/ruby-minitar,ruby25.TEST_IS_INTERACTIVE=No
archivers/ruby-minitar,ruby25.HOMEPAGE=http://www.github.com/atoulme/minitar
archivers/ruby-minitar,ruby25.DISTNAME=minitar-0.8
archivers/ruby-minitar,ruby25.USE_GMAKE=No
archivers/ruby-minitar,ruby25.USE_GROFF=No
archivers/ruby-minitar,ruby25.MODULES=lang/ruby
archivers/ruby-minitar,ruby25.FLAVORS=ruby24 ruby25 ruby26 jruby
archivers/ruby-minitar,ruby25.NO_BUILD=No
archivers/ruby-minitar,ruby25.CONFIGURE_STYLE=ruby gem
archivers/ruby-minitar,ruby25.USE_LIBTOOL=Yes
archivers/ruby-minitar,ruby25.SEPARATE_BUILD=No
archivers/ruby-minitar,ruby25.TARGETS= do-build do-install
archivers/ruby-minitar,ruby25.MAINTAINER=Sebastian Reitenbach <sebastia@openbsd.org>
archivers/ruby-minitar,ruby25.MAKEFILE_LIST=/usr/share/mk/sys.mk Makefile /usr/share/mk/bsd.port.mk /usr/share/mk/bsd.own.mk /etc/mk.c
archivers/ruby-minitar,ruby25.USE_LLD=Yes
archivers/ruby-minitar,ruby25.USE_WXNEEDED=No
archivers/ruby-minitar,ruby25.COMPILER=base-clang base-gcc gcc3
archivers/ruby-minitar,ruby25.COMPILER_LANGS=c c++
archivers/ruby-minitar,ruby25.COMPILER_LINKS= clang /usr/bin/clang  clang++ /usr/bin/clang++  cc /usr/bin/cc c++ /usr/bin/c++
```

Schematically

```
$ for every port; do
        SUBDIR=$port make dump-vars
done | mksqlite sqlports
$ sqlite3 sqlports
sqlite> select count(fullpkgpath) from ports where wantlib like '%QtCore%';
216
sqlite> ...
```

Scale issues

- Many variables
- OO approach (roughly 90 vars, 30 classes)

```
our $vars = {
    AUTOCONF_VERSION => 'AutoVersionVar',
    AUTOMAKE_VERSION => 'AutoVersionVar',
    BROKEN => 'BrokenVar',
    BUILD_DEPENDS => 'BuildDependsVar',
    CATEGORIES => 'CategoriesVar',
    COMES_WITH => 'DefinedVar',
    COMMENT => 'AnyVar',
    COMPILER_LINKS => 'CompilerLinksVar',
    CONFIGURE_ARGS => 'ConfigureArgsVar',
    CONFIGURE_STYLE => 'ConfigureVar',
    DESCR => 'DescrVar',
    DISTFILES => 'SupdistfilesVar',
    DPB_PROPERTIES => 'DPBPropertiesVar',
    PATCHFILES => 'PatchfilesVar',
    DISTNAME => 'AnyVar',
    DIST_SUBDIR => 'DefinedVar',
    EPOCH => 'AnyVar',
    FLAVORS => 'FlavorsVar',
    FULLPKGNAME => 'FullpkgnameVar',
    GH_ACCOUNT => 'DefinedVar',
    GH_COMMIT => 'DefinedVar',
    GH_PROJECT => 'DefinedVar',
    GH_TAGNAME => 'DefinedVar',
    HOMEPAGE => 'AnyVar',
    IGNORE => 'DefinedVar',
    IS_INTERACTIVE => 'AnyVar',
```

Types of data

- Boolean/limited choice variables (USE_GMAKE)
- Lists (DISTFILES, WANTLIB...)
- ordered lists (CONFIGURE_ARGS)
- ...

In the beginning

```
CREATE TABLE Ports (FULLPKGPATH TEXT NOT NULL PRIMARY KEY,
SHARED_LIBS TEXT,FULLPKGNAME TEXT,IS_INTERACTIVE TEXT,MASTER_SITES5 TEXT,
MASTER_SITES9 TEXT,PERMIT_PACKAGE_FTP TEXT,REGRESS_DEPENDS TEXT,
CONFIGURE_STYLE TEXT,CATEGORIES TEXT,MASTER_SITES6 TEXT,
SHARED_ONLY TEXT, MAINTAINER TEXT,MASTER_SITES2 TEXT,
MODULES TEXT,SUPDISTFILES TEXT,BROKEN TEXT,PSEUDO_FLAVORS TEXT,
DIST_SUBDIR TEXT,COMMENT TEXT,PERMIT_DISTFILES_FTP TEXT,DESCR TEXT,
MASTER_SITES0 TEXT,USE_MOTIF TEXT,WANTLIB TEXT,MULTI_PACKAGES TEXT,
SEPARATE_BUILD TEXT,BUILD_DEPENDS TEXT,AUTOCONF_VERSION TEXT,
USE_LIBTOOL TEXT,LIB_DEPENDS TEXT,MASTER_SITES8 TEXT,PKGNAME TEXT,
ONLY_FOR_ARCHS TEXT,MASTER_SITES TEXT,PACKAGES TEXT,NO_REGRESS TEXT,
PACKAGING TEXT,RUN_DEPENDS TEXT,AUTOMAKE_VERSION TEXT,DISTFILES TEXT,
HOMEPAGE TEXT,SUBPACKAGE TEXT,MASTER_SITES3 TEXT,MASTER_SITES7 TEXT,
CONFIGURE_ARGS TEXT,PKG_ARCH TEXT,FLAVORS TEXT,DISTNAME TEXT,NO_BUILD TEXT,
USE_GMAKE TEXT,PERMIT_DISTFILES_CDROM TEXT,REGRESS_IS_INTERACTIVE TEXT,
PERMIT_PACKAGE_CDROM TEXT,MASTER_SITES4 TEXT,MASTER_SITES1 TEXT);
```

Several issues

- Real database with actual keys
- ... unusable with sqlite3
- ... so requires lots of views
- ... but views are unreadable for humans

Better schema

```
CREATE VIEW _Ports AS SELECT T0039.FULLPKGPATH AS FULLPKGPATH,
T0040.VALUE AS AUTOCONF_VERSION, T0041.VALUE AS AUTOMAKE_VERSION,
Ports.COMES_WITH AS COMES_WITH, Ports.COMMENT AS COMMENT, Ports.COMPILER
AS COMPILER, Ports.COMPILER_LANGS AS COMPILER_LANGS, Ports.DISTFILES
AS DISTFILES, Ports.DISTNAME AS DISTNAME, Ports.DIST_SUBDIR AS
DIST_SUBDIR, Ports.EPOCH AS EPOCH, Ports.FIX_EXTRACT_PERMISSIONS
AS FIX_EXTRACT_PERMISSIONS, Ports.FULLPKGNAME AS FULLPKGNAME,
Ports.GH_ACCOUNT AS GH_ACCOUNT, Ports.GH_COMMIT AS GH_COMMIT,
Ports.GH_PROJECT AS GH_PROJECT, Ports.GH_TAGNAME AS GH_TAGNAME,
Ports.HOMEPAGE AS HOMEPAGE, Ports.IGNORE AS IGNORE, Ports.IS_INTERACTIVE
AS IS_INTERACTIVE, T0042.VALUE AS MAINTAINER, Ports.NO_BUILD AS
NO_BUILD, Ports.NO_TEST AS NO_TEST, Ports.PATCHFILES AS PATCHFILES,
T0043.VALUE AS PERMIT_DISTFILES_FTP, T0044.VALUE AS PERMIT_PACKAGE_CDROM,
T0045.VALUE AS PERMIT_PACKAGE_FTP, Ports.PKGNAME AS PKGNAME,
Ports.PKGSPEC AS PKGSPEC, T0046.VALUE AS PKG_ARCH, Ports.PORTROACH
AS PORTROACH, Ports.PORTROACH_COMMENT AS PORTROACH_COMMENT, T0047.VALUE
AS PREFIX, Ports.PSEUDO_FLAVOR AS PSEUDO_FLAVOR, Ports.REVISION AS
REVISION, T0048.VALUE AS SEPARATE_BUILD, Ports.STATIC_PLIST AS
STATIC_PLIST, Ports.SUBPACKAGE AS SUBPACKAGE, Ports.SUPDISTFILES
AS SUPDISTFILES, Ports.TEST_IS_INTERACTIVE AS TEST_IS_INTERACTIVE,
Ports.UPDATE_PLIST_ARGS AS UPDATE_PLIST_ARGS, Ports.USE_GMAKE AS
USE_GMAKE, Ports.USE_GROFF AS USE_GROFF, Ports.USE_LIBTOOL AS
USE_LIBTOOL, Ports.USE_WXNEEDED AS USE_WXNEEDED FROM Ports JOIN
Paths T0039 ON T0039.ID=Ports.FULLPKGPATH LEFT JOIN AutoVersion
T0040 ON T0040.KEYREF=Ports.AUTOCONF_VERSION LEFT JOIN AutoVersion
T0041 ON T0041.KEYREF=Ports.AUTOMAKE_VERSION JOIN Email T0042 ON
T0042.KEYREF=Ports.MAINTAINER JOIN Keywords2 T0043 ON
T0043.KEYREF=Ports.PERMIT_DISTFILES_FTP JOIN Keywords2 T0044 ON
T0044.KEYREF=Ports.PERMIT_PACKAGE_CDROM JOIN Keywords2 T0045 ON
T0045.KEYREF=Ports.PERMIT_PACKAGE_FTP JOIN Arch T0046 ON
T0046.KEYREF=Ports.PKG_ARCH JOIN Prefix T0047 ON T0047.KEYREF=Ports.PREFIX
```

ORM ?

Nope! For instance, p5-DBIx-Class depends on

- p5-Clone-Choose-0.010
- p5-Hash-Merge-0.300
- p5-MRO-Compat-0.13
- p5-Module-Find-0.13
- p5-Sub-Exporter-0.987
- p5-Sub-Exporter-Progressive-0.001013
- p5-Devel-GlobalDestruction-0.14
- p5-Class-Method-Modifiers-2.12
- p5-Role-Tiny-2.000005
- p5-Sub-Quote-2.005000
- p5-strictures-2.000005
- p5-Moo-2.003004
- p5-Path-Class-0.37
- p5-Scope-Guard-0.21
- p5-Sub-Name-0.21
- p5-Variable-Magic-0.62
- p5-B-Hooks-EndOfScope-0.24
- p5-namespace-clean-0.27
- p5-Config-Any-0.32
- p5-Context-Preserve-0.03
- p5-Data-Dumper-Concise-2.022
- p5-SQL-Abstract-1.86
- p5-Class-XSAccessor-1.19
- p5-Class-Accessor-Grouped-0.10014
- p5-Algorithm-C3-0.10
- p5-Sub-Uplevel-0.2800v0
- p5-Test-Exception-0.43
- p5-Class-C3-0.34
- p5-Class-Inspector-1.31
- p5-Class-C3-Componentised-1.001002

Not appropriate

- Not for human consumption

```
SELECT me.trackid, me.cd, me.title FROM track me  JOIN cd cd ON
cd.cdid = me.cd  JOIN artist artist ON artist.artistid = cd.artist
WHERE ( artist.name = ? )
```

- Can't create views, only requests

Pseudo ORM for humans

- Geared towards creating views
- Readable views
- Piecewise (OO for variables)

Example 1

```
CREATE VIEW Targets AS
    SELECT
        Id AS PathId,
        _Paths.FullPkgPath AS FullPkgPath,
        _TargetKeys.Value AS Value,
        N
    FROM _Targets
        JOIN _Paths
            ON Canonical=_Targets.FullPkgPath
        JOIN _TargetKeys
            ON KeyRef=_Targets.Value
```

Example 2

```
CREATE VIEW Ports AS
    SELECT
        Id AS PathId,
        _Paths.FullPkgPath AS FullPkgPath,
        _AutoVersion.Value AS AUTOCONF_VERSION,
        T0001.Value AS AUTOMAKE_VERSION,
        Depends_ordered.Value AS BUILD_DEPENDS,
        Categories_ordered.Value AS CATEGORIES,
        COMES_WITH,
        COMMENT,
        COMPILER,
        COMPILER_LANGS,
        CompilerLinks_ordered.Value AS COMPILER_LINKS,
        ConfigureArgs_ordered.Value AS CONFIGURE_ARGS,
        Configure_ordered.Value AS CONFIGURE_STYLE,
        _Descr.Filename AS DESCR,
        ...
    FROM _Ports
        JOIN _Paths
            ON Canonical=_Ports.FullPkgPath
        LEFT JOIN _AutoVersion
            ON _AutoVersion.KeyRef=AUTOCONF_VERSION
        LEFT JOIN _AutoVersion T0001
            ON T0001.KeyRef=AUTOMAKE_VERSION
        LEFT JOIN Depends_ordered
            ON Depends_ordered.FullPkgpath=_Ports.FullPkgpath AND Depends_ordered.Type=2
        JOIN Categories_ordered
            ON Categories_ordered.FullPkgpath=_Ports.FullPkgpath
        LEFT JOIN CompilerLinks_ordered
            ON CompilerLinks_ordered.FullPkgpath=_Ports.FullPkgpath
        LEFT JOIN ConfigureArgs_ordered
```

```
my $t = "_Paths";
my $v = "Paths";
Sql::Create::Table->new($t)->add(
    Sql::Column::Key->new("Id")->noautoincrement,
    Sql::Column::Text->new("FullPkgPath")->notnull->unique,
    Sql::Column::Integer->new("PkgPath")->references($t),
    Sql::Column::Integer->new("Canonical")->references($t));
Sql::Create::View->new($v, origin => $t)->add(
    Sql::Column::View->new("PathId", origin => "Id"),
    Sql::Column::View->new("FullPkgPath"),
    Sql::Column::View->new("PkgPath", origin => "FullPkgPath")
        ->join(Sql::Join->new($t)->add(
            Sql::Equal->new("Id", "PkgPath"))),
    Sql::Column::View->new("Canonical", origin => "FullPkgPath")
        ->join(Sql::Join->new($t)->add(
            Sql::Equal->new("Id", "Canonical")))
    );
```

```
CREATE TABLE _Paths (Id INTEGER PRIMARY KEY,
  FullPkgPath TEXT NOT NULL UNIQUE,
  PkgPath INTEGER NOT NULL REFERENCES _Paths(Id),
  Canonical INTEGER NOT NULL REFERENCES _Paths(Id));
CREATE VIEW Paths AS
    SELECT
        _Paths.Id AS PathId,
        _Paths.FullPkgPath AS FullPkgPath,
        T0001.FullPkgPath AS PkgPath,
        T0002.FullPkgPath AS Canonical
    FROM _Paths
        JOIN _Paths T0001
            ON T0001.Id=_Paths.PkgPath
        JOIN _Paths T0002
            ON T0002.Id=_Paths.Canonical
```

```
sub pathref
{
        my $j = Sql::Join->new('_Paths')->add(
            Sql::Equal->new('Canonical', 'FullPkgPath'));
        return (Sql::Column::View->new("PathId", origin => "Id")->join($j),
            Sql::Column::View->new('FullPkgPath')->join($j));
}
```

```
CREATE VIEW Multi AS
    SELECT
        _Paths.Id AS PathId,
        _Paths.FullPkgPath AS FullPkgPath,
        Value,
        T0001.FullPkgPath AS SubPkgPath
    FROM _Multi
        JOIN _Paths
            ON _Paths.Canonical=_Multi.FullPkgPath
        JOIN _Paths T0001
            ON T0001.Id=SubPkgPath
```

Crazy SQL

```
with recursive d (fullpkgpath, dependspath) as
        (select root.fullpkgpath, root.dependspath
                from _canonical_depends root
                        join _paths
                            on root.dependspath=_paths.canonical
                        join _paths p2
                            on p2.fullpkgpath="$1" and p2.id=_paths.pkgpath
            union
            select child.fullpkgpath, child.dependspath
            from d parent, _canonical_depends child
                where parent.fullpkgpath=child.dependspath)
select distinct(_paths.fullpkgpath) from d
    join _paths
        on _paths.id=d.fullpkgpath
order by _paths.fullpkgpath;
```

Problems

- Testing: I had to write a quick framework
- ... to diff schemas
- and some queries

Normalization

Making sure I get the exact same set of ports

Slower queries

Don't use the user queries in a tight loop

```
CREATE VIEW NotForArch_ordered AS
    WITH o AS
        (SELECT
            FullPkgPath,
            _Arch.Value AS Value
         FROM _NotForArch
            JOIN _Arch
                ON KeyRef=_NotForArch.Value
            ORDER BY N)
    SELECT
        FullPkgPath,
        group_concat(Value, ' ') AS Value
    FROM o
        GROUP BY FullPkgPath
```

- Paradox: having proper keys means that things do move
- So adjuncting a db with keys that move requires translation

No questions, right ?