# examples/output\_combiner.pql by Pequel

sample@youraddress.com

Output Combiner Example Script

# Table of Contents Output Combiner Example Script

SCRIPT NAME	1
DESCRIPTION	1
1. PROCESS DETAILS	1
1.1 SALES_TOTAL	1
Description	1
Derived Input Field Evaluation	1
1.2 LOCATION	1
Description	1
1.3 DESCRIPTION	1
Description	1
2. CONFIGURATION SETTINGS	2
2.1 pequeldoc	2
2.2 detail	2
2.3 noverbose	2
2.4 prefix	2
2.5 script_name	2
2.6 input_file	2
2.7 optimize	2
2.8 doc_title	2
2.9 doc_email	2
2.10 doc_version	2
3. TABLES	3
4. TABLE INFORMATION SUMMARY	4
4.1 Table List Sorted By Table Name	4
5. EXAMPLES/OUTPUT_COMBINER.PQL	5
options	5
input section	5
filter	5
output section	5
6. PEQUEL GENERATED PROGRAM	6
7. ABOUT PEQUEL	18
COPYRIGHT	18

#### **SCRIPT NAME**

examples/output\_combiner.pql

#### **DESCRIPTION**

#### 1. PROCESS DETAILS

Input records are read from copy\_output.pql. The input record contains **3** fields. Fields are delimited by the '|' character.

Output records are written to standard output. The output record contains **3** fields. Fields are delimited by the '|' character.

Input records are eliminated (filtered) unless DESCRIPTION !~ /State\s+Total/i.

#### 1.1 SALES TOTAL

**Output Field** 

#### Description

Set to input field **SALES\_TOTAL\_FMT** 

#### **Derived Input Field Evaluation**

=> &sprintf("%16s",&commify(&sprintf("%.2f",SALES\_TOTAL)))

#### 1.2 LOCATION

**Output Field** 

#### Description

Set to input field LOCATION

#### 1.3 DESCRIPTION

**Output Field** 

#### Description

Set to input field **DESCRIPTION** 

#### 2. CONFIGURATION SETTINGS

#### 2.1 pequeldoc

generate pod / pdf pequel script Reference Guide.: pdf

#### 2.2 detail

Include Pequel Generated Program chapter in Pequeldoc: 1

#### 2.3 noverbose

do not progress counter: 1

#### 2.4 prefix

directory pathname prefix.: examples

#### 2.5 script\_name

script filename: examples/output\_combiner.pql

#### 2.6 input\_file

input data filename: copy\_output.pql

#### 2.7 optimize

optimize generated code.: 1

#### 2.8 doc\_title

document title.: Output Combiner Example Script

#### 2.9 doc\_email

document email entry.: sample@youraddress.com

#### 2.10 doc\_version

document version for pequel script.: 2.3

# 3. TABLES

### 4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

# 5. EXAMPLES/OUTPUT\_COMBINER.PQL

#### options

```
pequeldoc(pdf)
detail(1)
noverbose(1)
prefix(examples)
script_name(examples/output_combiner.pql)
input_file(copy_output.pql)
optimize(1)
doc_title(Output Combiner Example Script)
doc_email(sample@youraddress.com)
doc_version(2.3)
```

#### input section

```
LOCATION
DESCRIPTION
SALES_TOTAL
SALES_TOTAL_FMT => &sprintf("%16s",&commify(&sprintf("%.2f",SALES_TOTAL)))
```

#### filter

DESCRIPTION !~ /State\s+Total/i

#### output section

string SALES\_TOTAL SALES\_TOTAL\_FMT string LOCATION LOCATION string DESCRIPTION DESCRIPTION

#### 6. PEQUEL GENERATED PROGRAM

```
# vim: syntax=perl ts=4 sw=4
#Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
          : http://sourceforge.net/projects/pequel/
#Script Name : examples/output_combiner.pql
#Created On : Tue Oct 25 09:25:37 2005
#For
#-----
#Options:
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
\#noverbose(1) do not progress counter
#prefix(examples) directory pathname prefix.
#script_name(examples/output_combiner.pql) script filename
#input_file(copy_output.pql) input data filename
\# optimize(1) optimize generated code.
#doc_title(Output Combiner Example Script) document title.
#doc_email(sample@youraddress.com) document email entry
#doc_version(2.3) document version for pequel script.
use strict;
use constant \_I\_LOCATION
                           => int
                                    0;
1;
                                    2;
use constant _I_SALES_TOTAL_FMT => int
                                    3;
1;
use constant _O_LOCATION
                                    2;
use constant _O_DESCRIPTION
                           => int
                                    3;
local $\="\n";
local $,="|";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 3;
my @I_VAL;
my @O VAL;
foreach my $f (1..3) { $0_VAL[$f] = undef; }
if (open(INPUT_COPY_OUTPUT, '-|') == 0) # Fork -- read from child
   &p_input_copy_output::input_copy_output;
   exit(0);
}
while (<INPUT COPY OUTPUT>)
   chomp;
   @I_VAL = split("[|]", $_);
   next unless ($I_VAL[_I_DESCRIPTION] !~ /State\s+Total/i);
   $I_VAL[_I_SALES_TOTAL_FMT] = sprintf("%16s",&{sub
   my $idec = index(sprintf("%.2f",$I_VAL[_I_SALES_TOTAL]), '.');
   my $dec = $idec > 0 ? substr(sprintf("%.2f", $I_VAL[_I_SALES_TOTAL]), $idec) : '';
   my $txt = reverse($idec > 0 ? substr(sprintf("%.2f",$I_VAL[_I_SALES_TOTAL]), 0, $idec) : sprintf("%.2f",$I
VAL[ I SALES TOTAL]));
   t = -\infty / (\d\d\d)(?=\d)(?!\d^*\.)/$1,/g;
   return (scalar reverse $txt) . $dec;
   $0_VAL[_O_SALES_TOTAL] = $I_VAL[_I_SALES_TOTAL_FMT];
   $0_VAL[_0_LOCATION] = $I_VAL[_I_LOCATION];
   $0_VAL[_O_DESCRIPTION] = $I_VAL[_I_DESCRIPTION];
   print STDOUT
      $0_VAL[_O_SALES_TOTAL],
      SO VAL[ O LOCATION].
      $0_VAL[_O_DESCRIPTION]
#-----
   package p input chain pequel pt1;
   sub input chain pequel pt1
    vim: syntax=perl ts=4 sw=4
   Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
             : http://sourceforge.net/projects/pequel/
   Script Name : examples/chain_pequel_pt1.pql
    Created On : Tue Oct 25 09:25:27 2005
        Options:
```

```
input file(sample.data) input data filename
       optimize(1) optimize generated code.
       hash(1) Generate in memory. Input data can be unsorted.
#
       doc_title(Pequel Chaining Part-1 Example Script) document title.
       doc email(sample@youraddress.com) document email entry.
       doc version(2.3) document version for pequel script.
use strict;
      use constant _I_PRODUCT_CODE
                                 => int
      use constant _I_COST_PRICE
                                 => int
                                          1;
      use constant _I_DESCRIPTION use constant _I_SALES_CODE
                                 => int
                                          2:
                                 => int
                                          3;
      use constant _I_SALES_PRICE
                                 => int
                                           4;
                                 => int
      use constant _I_SALES_QTY
                                           5;
      use constant _I_SALES_DATE
                                 => int
                                           6;
      use constant _I_LOCATION
                                 => int
      use constant _I_SALES_TOTAL => int
      use constant _O_LOCATION
                                 => int
                                          1;
      use constant _O_PRODUCT_CODE
use constant _O_SALES_TOTAL
                                 => int
                                           2;
                                 => int
      local $\="\n";
      local $,="|";
      use constant VERBOSE => int 10000;
      use constant LAST ICELL => int 8;
      my @I_VAL;
      my %O_VAL;
      my $key;
      open(DATA, q{examples/sample.data})|| die "Cannot open examples/sample.data: $!";
      open(STDOUT, '|-', q{sort -t'|' -y -k 1,1 2>/dev/null |});
      while (<DATA>)
      {
          chomp;
          @I_VAL = split("[|]", $_);
          $key = ( $I_VAL[_I_LOCATION] ) . '|' . ( $I_VAL[_I_PRODUCT_CODE] );
          $0_VAL{$key}{_O_LOCATION} = $I_VAL[_I_LOCATION];
          $0_VAL{$key}{_0_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
          $I_VAL[_I_SALES_TOTAL] = $I_VAL[_I_SALES_QTY] * $I_VAL[_I_SALES_PRICE];
          $O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
      foreach $key (sort keys %O_VAL)
          print STDOUT
             $O_VAL{$key}{_O_LOCATION},
             $O_VAL{$key}{_O_PRODUCT_CODE},
             $0_VAL{$key}{_O_SALES_TOTAL}
      close(STDOUT);
   +-----
   }
}
{
   package p_input_copy_output;
   sub input_copy_output
     vim: syntax=perl ts=4 sw=4
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
              : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy_output.pql
    Created On : Tue Oct 25 09:25:27 2005
#-----
       input_file(chain_pequel_pt1.pql) input data filename
       optimize(1) optimize generated code.
       doc_title(Copy Output Record Example Script) document title.
       doc_email(sample@youraddress.com) document email entry.
       doc_version(2.3) document version for pequel script.
use strict;
      use Fcntl ':flock';
      use constant _I_LOCATION
      use constant _I_PRODUCT_CODE => int
      use constant _I_SALES_TOTAL
                                  => int
      use constant _I_LOCATION_DESC => int
                                           3;
      use constant _I_DESCRIPTION
                                  => int
                                            4;
      use constant _O_LOCATION
      => int
                                           2;
      use constant _T_LOC_DESCRIPT_FLD_1 => int
```

```
use constant _I_LOC_DESCRIPT_LOCATION_FLD_KEY => int
                                                                                                                                 5;
               use constant _I_LOC_DESCRIPT_LOCATION_FLD_1 => int
                local $\="\n";
                local $,="|";
               use constant VERBOSE => int 10000;
               use constant LAST ICELL => int 4;
               mv @I VAL;
               my @O_VAL;
               my $key I LOCATION;
               my $previous_key__I_LOCATION = undef;
               foreach my $f (1..3) { $O_VAL[$f] = undef; }
my $_TABLE_LOC_DESCRIPT = &InitLookupLOC_DESCRIPT; # ref to $$LOC_DESCRIPT hash
                if (open(INPUT_CHAIN_PEQUEL_PT1, '-|') == 0) # Fork -- read from child
                        &p input chain pequel pt1::input chain pequel pt1;
                        exit(0);
                }
                open(STDOUT, '|-', q{sort -t'|' -y -k 3nr,3nr 2>/dev/null |});
                if (open(DIVERT_INPUT_COPY_OUTPUT_WA, '|-') == 0) # Fork -- write to child
               {
                        &p_divert_input_copy_output_wa::divert_input_copy_output_wa;
                        exit(0);
                if (open(DIVERT_INPUT_COPY_OUTPUT_SA, '|-') == 0) # Fork -- write to child
                {
                        &p divert input copy output sa::divert input copy output sa;
                        exit(0);
                }
                if (open(DIVERT_INPUT_COPY_OUTPUT_NSW, '|-') == 0) # Fork -- write to child
                {
                        &p_divert_input_copy_output_nsw::divert_input_copy_output_nsw;
                        exit(0);
                }
                if (open(DIVERT_INPUT_COPY_OUTPUT_VIC, '|-') == 0) # Fork -- write to child
                {
                        &p_divert_input_copy_output_vic::divert_input_copy_output_vic;
                        exit(0);
                }
                if (open(DIVERT\_INPUT\_COPY\_OUTPUT\_NT, '|-') == 0) # Fork -- write to child
                {
                        &p_divert_input_copy_output_nt::divert_input_copy_output_nt;
                        exit(0);
                while (<INPUT_CHAIN_PEQUEL_PT1>)
                {
                        chomp;
                        @I_VAL = split("[|]", $_);
                       \verb|next unless ($I_VAL[_I_LOCATION]| eq 'WA' || $I_VAL[_I_LOCATION]| eq 'SA' || $I_VAL[_I_LOCATION]| eq 'WA' || $I_VAL[_I_LOC
'NSW' || $I_VAL[_I_LOCATION] eq 'VIC' || $I_VAL[_I_LOCATION] eq 'NT');
                        if ($I_VAL[_I_LOCATION] eq 'WA')
                        {
                                print DIVERT_INPUT_COPY_OUTPUT_WA
                                       @I_VAL[0..LAST_ICELL];
                                next;
                        }
                        if ($I_VAL[_I_LOCATION] eq 'SA')
                               print DIVERT_INPUT_COPY_OUTPUT_SA
                                       @I_VAL[0..LAST_ICELL];
                                next;
                        }
                        if ($I_VAL[_I_LOCATION] eq 'NSW')
                                print DIVERT_INPUT_COPY_OUTPUT_NSW
                                       @I_VAL[0..LAST_ICELL];
                                next;
                        if ($I_VAL[_I_LOCATION] eq 'VIC')
                                print DIVERT_INPUT_COPY_OUTPUT_VIC
                                       @I_VAL[0..LAST_ICELL];
                                next;
                        }
                        if ($I_VAL[_I_LOCATION] eq 'NT')
```

```
print DIVERT_INPUT_COPY_OUTPUT NT
                 @I VAL[0..LAST ICELL];
              next;
           }
                I LOCATION = $I VAL[ I LOCATION];
           $kev
           if (!defined($previous_key__I_LOCATION))
              $previous key I LOCATION = $key I LOCATION;
           }
           elsif ($previous_key__I_LOCATION ne $key__I_LOCATION)
              flock(STDOUT, LOCK_EX);
              print STDOUT
                  $0_VAL[_O_LOCATION],
                  $0_VAL[_O_DESCRIPTION],
                  $0_VAL[_O_SALES_TOTAL]
              if
                  $0_VAL[_O_SALES_TOTAL] > 0
              flock(STDOUT, LOCK_UN);
              $previous_key__I_LOCATION = $key__I_LOCATION;
              @O_VAL = undef;
           }
           $!_VAL[_I_LOCATION_DESC] = $$_TABLE_LOC_DESCRIPT{qq{$I_VAL[_I_LOCATION]}};
           $0_VAL[_0_LOCATION] = $I_VAL[_I_LOCATION_DESC];
           $I_VAL[_I_DESCRIPTION] = 'State Total';
           $0_VAL[_O_DESCRIPTION] = $I_VAL[_I_DESCRIPTION];
           $0_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
       flock(STDOUT, LOCK_EX);
       print STDOUT
          $0_VAL[_O_LOCATION],
           $0_VAL[_O_DESCRIPTION],
           $0_VAL[_O_SALES_TOTAL]
       (
           $0_VAL[_O_SALES_TOTAL] > 0
       flock(STDOUT, LOCK_UN);
       close(DIVERT_INPUT_COPY_OUTPUT_NT);
       close(DIVERT_INPUT_COPY_OUTPUT_VIC);
       close(DIVERT_INPUT_COPY_OUTPUT_NSW);
       close(DIVERT_INPUT_COPY_OUTPUT_SA);
       close(DIVERT_INPUT_COPY_OUTPUT_WA);
       close(STDOUT);
    +++++ Table LOC_DESCRIPT --> Type :Pequel::Type::Table::Local +++++
       sub InitLookupLOC_DESCRIPT
           my %_TABLE_LOC_DESCRIPT;
           %_TABLE_LOC_DESCRIPT =
               'NSW' => 'New South Wales',
              'NT' => 'Northern Territory',
              'QLD' => 'Queensland',
              'SA' => 'South Australia',
'VIC' => 'Victoria',
              'WA' => 'Western Australia'
           );
           return \%_TABLE_LOC_DESCRIPT;
       }
   }
}
{
   package p_divert_input_copy_output_sa;
   sub divert_input_copy_output_sa
#
     vim: syntax=perl ts=4 sw=4
#-----
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
               : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy_output_SA.pql
    Created On : Tue Oct 25 09:25:31 2005
Options:
        optimize(1) optimize generated code.
```

```
doc_title(Copy Output Record Example Script) document title.
        doc email(sample@youraddress.com) document email entry.
        doc version(2.3) document version for peguel script.
  #-
       use strict;
       use Fcntl ':flock';
       0;
                                                  1;
       use constant _I_SALES_TOTAL use constant _I_LOCATION_NAME
                                       => int
                                                  2;
                                      => int
                                                  3;
       use constant _O_PRODUCT_CODE => int
                                                  1;
                                                  2;
       use constant _O_SALES_TOTAL
local $\="\n";
                                       => int
                                                3;
       local $,="|";
       use constant VERBOSE => int 10000;
       use constant LAST_ICELL => int 3;
       my @I VAL;
       my @O VAL;
       my $key__I_PRODUCT_CODE;
       my %previous_key_I_PRODUCT_CODE = undef;
foreach my $f (1..3) { $O_VAL[$f] = undef; }
     Sort:PRODUCT_CODE(asc:string)
       open(DATA, q{cat - | sort -t'|' -y -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
       if (open(COPY_OUTPUT_COPY_OUTPUT_COMBINER, '|-') == 0) # Fork -- write to child
       {
           &p_copy_output_copy_output_combiner::copy_output_copy_output_combiner;
           exit(0);
       }
       while (<DATA>)
           chomp;
           @I_VAL = split("[|]", $_);
           $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
           if (!defined($previous_key__I_PRODUCT_CODE))
               $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
           elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
               flock(STDOUT, LOCK_EX);
               print STDOUT
                   $0_VAL[_O_LOCATION_NAME],
                   $0_VAL[_O_PRODUCT_CODE],
                   $0_VAL[_O_SALES_TOTAL]
               flock(STDOUT, LOCK_UN);
               if ($0_VAL[_O_SALES_TOTAL] > 0)
                   flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
                   print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                       $0_VAL[_O_LOCATION_NAME],
                       $0_VAL[_O_PRODUCT_CODE],
                       $0_VAL[_O_SALES_TOTAL]
                   flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
               $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
               @O_VAL = undef;
           $I_VAL[_I_LOCATION_NAME] = 'South Australia';
           $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
            $0_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $0_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
       flock(STDOUT, LOCK_EX);
       print STDOUT
           $O_VAL[_O_LOCATION_NAME],
           $0_VAL[_O_PRODUCT_CODE],
           $0_VAL[_O_SALES_TOTAL]
       flock(STDOUT, LOCK_UN);
       if ($O_VAL[_O_SALES_TOTAL] > 0)
           flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
           print COPY_OUTPUT_COPY_OUTPUT_COMBINER
               $O_VAL[_O_LOCATION_NAME],
               $0_VAL[_O_PRODUCT_CODE]
               $0_VAL[_O_SALES_TOTAL]
```

```
flock(COPY OUTPUT COPY OUTPUT COMBINER, LOCK UN);
       close(COPY OUTPUT COPY OUTPUT COMBINER);
}
}
{
   package p_divert_input_copy_output_wa;
   sub divert_input_copy_output_wa
     vim: syntax=perl ts=4 sw=4
#-
 Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
               : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy output WA.pgl
    Created On : Tue Oct 25 09:25:28 2005
    For
#-+
    Options:
        optimize(1) optimize generated code.
        doc_title(Copy Output Record Example Script) document title.
        doc_email(sample@youraddress.com) document email entry.
       doc_version(2.3) document version for pequel script.
#-+-+-+-+-+-+-+-+
      use strict;
       use Fcntl ':flock';
       use constant _I_LOCATION
                                    => int
                                             0;
       use constant _I_PRODUCT_CODE => int
use constant _I_SALES_TOTAL => int
       use constant _I_LOCATION_NAME => int
use constant _O_LOCATION_NAME => int
                                              3;
                                              1;
       => int
                                              2;
       local \= \n'';
       local $,="|";
       use constant VERBOSE => int 10000;
       use constant LAST_ICELL => int 3;
       my @I_VAL;
       my @O_VAL;
       my $key__I_PRODUCT_CODE;
       my $previous_key__I_PRODUCT_CODE = undef;
       foreach my $f (1..3) { $0_VAL[$f] = undef; }
     Sort:PRODUCT_CODE(asc:string)
       open(DATA, q{cat - | sort -t'|' -y -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
       if (open(COPY_OUTPUT_COPY_OUTPUT_COMBINER, ' \mid -' \rangle == 0) # Fork -- write to child
       {
           &p_copy_output_copy_output_combiner::copy_output_copy_output_combiner;
           exit(0);
       }
       while (<DATA>)
       {
           chomp;
           @I_VAL = split("[|]", $_);
           $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
           if (!defined($previous_key__I_PRODUCT_CODE))
              $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
           }
           elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
              flock(STDOUT, LOCK_EX);
              print STDOUT
                  $0_VAL[_O_LOCATION_NAME],
                  $0_VAL[_O_PRODUCT_CODE],
                  $0_VAL[_O_SALES_TOTAL]
              flock(STDOUT, LOCK_UN);
              if ($0_VAL[_O_SALES_TOTAL] > 0)
                  flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
                  print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                      $O_VAL[_O_LOCATION_NAME],
                      $0_VAL[_O_PRODUCT_CODE]
                      $0_VAL[_O_SALES_TOTAL]
                  flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
              $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
              @O_VAL = undef;
```

```
}
          $I_VAL[_I_LOCATION_NAME] = 'Western Australia';
          $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
          $0_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
          $0_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
       flock(STDOUT, LOCK EX);
       print STDOUT
          $0_VAL[_O_LOCATION_NAME],
          $0_VAL[_O_PRODUCT_CODE],
          $0 VAL[ O SALES TOTAL]
       flock(STDOUT, LOCK_UN);
       if ($0 VAL[ O SALES TOTAL] > 0)
       {
           flock(COPY OUTPUT COPY OUTPUT COMBINER, LOCK EX);
          print COPY_OUTPUT_COPY_OUTPUT_COMBINER
              SO VAL O LOCATION NAME ].
              $0_VAL[_O_PRODUCT_CODE],
              $0_VAL[_O_SALES_TOTAL]
          flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
       close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
           }
}
   package p_divert_input_copy_output_nt;
   sub divert_input_copy_output_nt
     vim: syntax=perl ts=4 sw=4
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
              : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy_output_NT.pql
    Created On : Tue Oct 25 09:25:36 2005
   Options:
       optimize(1) optimize generated code.
        doc_title(Copy Output Record Example Script) document title.
       doc_email(sample@youraddress.com) document email entry.
       doc_version(2.3) document version for pequel script.
       use strict;
       use Fcntl ':flock';
       use constant \_I\_LOCATION
                                    => int
       use constant _I_PRODUCT_CODE => int
                                             1;
       use constant _I_SALES_TOTAL
                                    => int
       use constant _I_LOCATION_NAME => int
       use constant _O_LOCATION_NAME => int
use constant _O_PRODUCT_CODE => int
use constant _O_SALES_TOTAL => int
       local \= \n'';
       local $,="|";
       use constant VERBOSE => int 10000;
       use constant LAST_ICELL => int 3;
       my @I_VAL;
       my @O_VAL;
       my $key__I_PRODUCT_CODE;
       my $previous_key__I_PRODUCT_CODE = undef;
       foreach my $f (1..3) { $0_VAL[$f] = undef; }
     Sort:PRODUCT_CODE(asc:string)
       {
           &p_copy_output_copy_output_combiner::copy_output_copy_output_combiner;
       while (<DATA>)
          @I_VAL = split("[|]", $_);
          $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
           if (!defined($previous_key__I_PRODUCT_CODE))
              $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
           }
```

```
elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
             flock(STDOUT, LOCK EX);
             print STDOUT
                 $0_VAL[_O_LOCATION_NAME],
                 SO VALI O PRODUCT CODE ].
                 $0_VAL[_O_SALES_TOTAL]
             flock(STDOUT, LOCK_UN);
             if ($0_VAL[_O_SALES_TOTAL] > 0)
                 flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
                 print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                    $0 VAL[ O LOCATION NAME],
                    $O_VAL[_O_PRODUCT_CODE],
                    SO VALI O SALES TOTAL 1
                 flock(COPY OUTPUT COPY OUTPUT COMBINER, LOCK UN);
             }
             $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
             @O_VAL = undef;
          }
          $I_VAL[_I_LOCATION_NAME] = 'Northern Territory';
          $0_VAL[_0_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
          $0_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
          $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
      flock(STDOUT, LOCK_EX);
      print STDOUT
          $0_VAL[_O_LOCATION_NAME],
          $0_VAL[_O_PRODUCT_CODE],
          $0_VAL[_O_SALES_TOTAL]
      flock(STDOUT, LOCK_UN);
      if ($0_VAL[_O_SALES_TOTAL] > 0)
          flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
          print COPY_OUTPUT_COPY_OUTPUT_COMBINER
             $0_VAL[_O_LOCATION_NAME],
             $0_VAL[_O_PRODUCT_CODE],
             $0_VAL[_O_SALES_TOTAL]
          flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
      close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
           package p_divert_input_copy_output_vic;
   sub divert_input_copy_output_vic
    vim: syntax=perl ts=4 sw=4
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
              : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy_output_VIC.pql
    Created On : Tue Oct 25 09:25:35 2005
    For
        optimize(1) optimize generated code.
       doc_title(Copy Output Record Example Script) document title.
       \verb"doc_email(sample@youraddress.com")" document email entry.
       doc_version(2.3) document version for pequel script.
use strict;
      use Fcntl ':flock';
      use constant _I_LOCATION
                                  => int
                                           0;
      use constant _I_PRODUCT_CODE => int
                                           1;
      3;
                                            1;
                                            3;
      local $\="\n";
local $,="|";
      use constant VERBOSE => int 10000;
      use constant LAST_ICELL => int 3;
```

```
my @I_VAL;
        my @O_VAL;
        my $key I PRODUCT CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..3) { $0_VAL[$f] = undef; }
      Sort:PRODUCT_CODE(asc:string)
open(DATA, q{cat - | sort -t'|' -y -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
        if (open(COPY_OUTPUT_COPY_OUTPUT_COMBINER, '|-') == 0) # Fork -- write to child
        {
            &p copy output copy output combiner::copy output copy output combiner;
            exit(0);
        }
        while (<DATA>)
        {
            chomp;
            @I_VAL = split("[|]", $_);
            $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
            if (!defined($previous_key__I_PRODUCT_CODE))
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
            }
            elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
            {
                flock(STDOUT, LOCK_EX);
                print STDOUT
                    $0_VAL[_O_LOCATION_NAME],
                    $0_VAL[_O_PRODUCT_CODE],
                    $0_VAL[_O_SALES_TOTAL]
                flock(STDOUT, LOCK_UN);
                if ($0_VAL[_O_SALES_TOTAL] > 0)
                    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
                    print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                        $0_VAL[_O_LOCATION_NAME],
                        $O_VAL[_O_PRODUCT_CODE],
                        $0_VAL[_O_SALES_TOTAL]
                    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
                }
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }
            $I_VAL[_I_LOCATION_NAME] = 'Victoria';
            $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
            $0_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
        flock(STDOUT, LOCK_EX);
        print STDOUT
            $0_VAL[_O_LOCATION_NAME],
            $0_VAL[_O_PRODUCT_CODE],
            $0_VAL[_O_SALES_TOTAL]
        flock(STDOUT, LOCK_UN);
        if ($O_VAL[_O_SALES_TOTAL] > 0)
            flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
            print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                $O_VAL[_O_LOCATION_NAME],
                $O_VAL[_O_PRODUCT_CODE],
                $0_VAL[_O_SALES_TOTAL]
            flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
        close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
}
    package p_divert_input_copy_output_nsw;
    sub divert_input_copy_output_nsw
      vim: syntax=perl ts=4 sw=4
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
                 : http://sourceforge.net/projects/pequel/
```

```
Script Name : examples/copy_output_NSW.pql
    Created On : Tue Oct 25 09:25:33 2005
#
    Options:
        optimize(1) optimize generated code.
        doc title(Copy Output Record Example Script) document title.
        doc email(sample@youraddress.com) document email entry.
       doc version(2.3) document version for pequel script.
#------
                                                           use strict;
       use Fcntl ':flock';
       use constant \_I\_LOCATION
                                      => int
                                               0;
       use constant _I_PRODUCT_CODE
                                     => int
                                               1;
       use constant _I_SALES_TOTAL => int
use constant _I_LOCATION_NAME => int
                                                2;
                                                3;
       use constant _O_PRODUCT_CODE => int
                                               1;
                                                2;
       use constant _O_SALES_TOTAL
                                    => int
       local $\="\n";
       local $,="|";
       use constant VERBOSE => int 10000;
       use constant LAST_ICELL => int 3;
       my @I VAL;
       my @O_VAL;
       my $key__I_PRODUCT_CODE;
       my $previous_key__I_PRODUCT_CODE = undef;
       foreach my $f (1..3) { $0_VAL[$f] = undef; }
     Sort:PRODUCT_CODE(asc:string)
       open(DATA, q{cat - | sort -t'|' -y -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
       if (open(COPY_OUTPUT_COPY_OUTPUT_COMBINER, '|-') == 0) # Fork -- write to child
       {
           &p_copy_output_copy_output_combiner::copy_output_copy_output_combiner;
           exit(0);
       }
       while (<DATA>)
           chomp;
           @I_VAL = split("[|]", $_);
           $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
           if (!defined($previous_key__I_PRODUCT_CODE))
           {
               $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
           }
           elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
               flock(STDOUT, LOCK_EX);
              print STDOUT
                  $O_VAL[_O_LOCATION_NAME],
                  $O_VAL[_O_PRODUCT_CODE],
                  $0_VAL[_O_SALES_TOTAL]
               flock(STDOUT, LOCK_UN);
               if ($O_VAL[_O_SALES_TOTAL] > 0)
               {
                   flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
                  print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                      $0_VAL[_O_LOCATION_NAME],
                      $0_VAL[_O_PRODUCT_CODE],
                      $0_VAL[_O_SALES_TOTAL]
                   flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
               $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
               @O_VAL = undef;
           $I_VAL[_I_LOCATION_NAME] = 'New South Wales';
           $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
           $0_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
           $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
       flock(STDOUT, LOCK_EX);
       print STDOUT
           $0_VAL[_O_LOCATION_NAME],
           $O_VAL[_O_PRODUCT_CODE],
           $O_VAL[_O_SALES_TOTAL]
       flock(STDOUT, LOCK_UN);
       if ($0_VAL[_O_SALES_TOTAL] > 0)
```

```
flock(COPY OUTPUT COPY OUTPUT COMBINER, LOCK EX);
          print COPY_OUTPUT_COPY_OUTPUT_COMBINER
              $0 VAL[ O LOCATION NAME],
              $0 VAL[ O PRODUCT CODE],
             $0 VAL[ O SALES TOTAL]
          flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
      }
      close(COPY OUTPUT COPY OUTPUT COMBINER);
   }
{
   package p_copy_output_copy_output_combiner;
   sub copy_output_copy_output_combiner
    vim: syntax=perl ts=4 sw=4
#-+
    Generated By: pequel Version 2.3-6, Build: Monday October 24 23:16:49 BST 2005
              : http://sourceforge.net/projects/pequel/
    Script Name : examples/copy_output_combiner.pql
    Created On : Tue Oct 25 09:25:30 2005
   For
        Options:
       optimize(1) optimize generated code.
       doc_title(Copy Output Record Example Script) document title.
       doc_email(sample@youraddress.com) document email entry.
       doc_version(2.3) document version for pequel script.
use strict;
      use Fcntl ':flock';
      use constant _I_LOCATION_NAME => int
      use constant _I_PRODUCT_CODE => int
use constant _I_SALES_TOTAL => int
       use constant _I_DESCRIPTION
                                   => int
      use constant _O_LOCATION_NAME => int
      local \= \n'';
       local $,="|";
       use constant VERBOSE => int 10000;
       use constant LAST_ICELL => int 3;
       my @I_VAL;
      my @O_VAL;
      my $key__I_LOCATION_NAME;
      my $previous_key__I_LOCATION_NAME = undef;
       foreach my $f (1..3) { $0_VAL[$f] = undef; }
     Sort:LOCATION_NAME(asc:string)
       open(DATA, q{cat - | sort -t'|' -y -k 1,1 2>/dev/null |}) || die "Cannot open input: !";
       while (<DATA>)
       {
          @I_VAL = split("[|]", $_);
          $key__I_LOCATION_NAME = $I_VAL[_I_LOCATION_NAME];
          if (!defined($previous_key__I_LOCATION_NAME))
              $previous_key__I_LOCATION_NAME = $key__I_LOCATION_NAME;
          elsif ($previous_key__I_LOCATION_NAME ne $key__I_LOCATION_NAME)
              flock(STDOUT, LOCK_EX);
                 $0_VAL[_O_LOCATION_NAME],
                 $0_VAL[_O_DESCRIPTION],
                 $0_VAL[_O_SALES_TOTAL]
              flock(STDOUT, LOCK_UN);
              $previous_key__I_LOCATION_NAME = $key__I_LOCATION_NAME;
              @O_VAL = undef;
          $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
          $I_VAL[_I_DESCRIPTION] = 'State Total';
          $0_VAL[_O_DESCRIPTION] = $I_VAL[_I_DESCRIPTION];
          $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
       flock(STDOUT, LOCK_EX);
      print STDOUT
          $O_VAL[_O_LOCATION_NAME],
```

#### 7. ABOUT PEQUEL

This document was generated by Pequel.

https://sourceforge.net/projects/pequel/

#### **COPYRIGHT**

Copyright ©1999-2005, Mario Gaffiero. All Rights Reserved. 'Pequel' TM Copyright ©1999-2005, Mario Gaffiero. All Rights Reserved.

This program and all its component contents is copyrighted free software by Mario Gaffiero and is released under the GNU General Public License (GPL), Version 2, a copy of which may be found at http://www.opensource.org/licenses/gpl-license.html

Pequel is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Pequel is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Pequel; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA