



examples/output_combiner.pql

by *Pequel*

sample@youraddress.com

Output Combiner Example Script

2.3

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 prefix	2
2.2 pequeldoc	2
2.3 detail	2
2.4 script_name	2
2.5 input_file	2
2.6 optimize	2
2.7 doc_title	2
2.8 doc_email	2
2.9 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

DescriptionSet to input field **SALES_TOTAL_FMT****Derived Input Field Evaluation**

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

1.2 LOCATION

Output Field

DescriptionSet to input field **LOCATION****1.3 DESCRIPTION**

Output Field

DescriptionSet to input field **DESCRIPTION**

2. CONFIGURATION SETTINGS

2.1 *prefix*

directory pathname prefix.: examples

2.2 *pequeldoc*

generate pod / pdf pequel script Reference Guide.: pdf

2.3 *detail*

Include Pequel Generated Program chapter in Pequeldoc: 1

2.4 *script_name*

script filename: examples/output_combiner.pql

2.5 *input_file*

input data filename: copy_output.pql

2.6 *optimize*

optimize generated code.: 1

2.7 *doc_title*

document title.: Output Combiner Example Script

2.8 *doc_email*

document email entry.: sample@youraddress.com

2.9 *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

```
prefix(examples)
pequeldoc(pdf)
detail(1)
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

```
#!/usr/bin/perl
#-----
# vim: syntax=perl ts=4 sw=4
#-----
#Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
#           : http://sourceforge.net/projects/pequel/
#Script Name : output_combiner.pql
#Created On  : Wed Nov 16 14:13:10 2005
#Perl Version: /usr/bin/perl 5.6.1 on solaris
#For         :
#-----
#Options:
#prefix(examples) directory pathname prefix.
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
#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;
use constant _I_DESCRIPTION       => int    1;
use constant _I_SALES_TOTAL       => int    2;
use constant _I_SALES_TOTAL_FMT   => int    3;
use constant _O_SALES_TOTAL       => int    1;
use constant _O_LOCATION          => int    2;
use constant _O_DESCRIPTION       => int    3;
local $\\="\n";
local $,="|";
print STDERR '[examples/output_combiner.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 3;
my @I_VAL;
my @O_VAL;
my $_inprec=0;
foreach my $f (1..3) { $_VAL[$f] = undef; }
if (open(READ_COPY_OUTPUT, '-|') == 0) # Fork -- read from child
{
    &p_read_copy_output::read_copy_output;
    exit(0);
}

print STDERR '[examples/output_combiner.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<READ_COPY_OUTPUT>)
{
    ++$_inprec;
    print STDERR '[examples/output_combiner.pql ' . localtime() . "] $_inprec records." if ($_inprec % VERBO
SE == 0);
    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]));
        $txt =~ s/(\d\d\d)(?=\d)(?!d*\.)/$1,/g;
        return (scalar reverse $txt) . $dec;
    }}
);
    $O_VAL[_O_SALES_TOTAL] = $I_VAL[_I_SALES_TOTAL_FMT];
    $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
    $O_VAL[_O_DESCRIPTION] = $I_VAL[_I_DESCRIPTION];
    print STDOUT
        $O_VAL[_O_SALES_TOTAL],
        $O_VAL[_O_LOCATION],
        $O_VAL[_O_DESCRIPTION]
    ;
}

close(READ_COPY_OUTPUT);
print STDERR '[examples/output_combiner.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
```

```

print STDERR '[examples/output_combiner.pql ' . localtime() . "] Code statistics: @{" . timestr($benchmark_timedi
ff) . "}";
#-----
{
    package p_read_copy_output;
    sub read_copy_output
    {
# !/usr/bin/perl
#-----
# vim: syntax=perl ts=4 sw=4
#-----
# Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
# : http://sourceforge.net/projects/pequel/
# Script Name : copy_output.pql
# Created On : Wed Nov 16 14:13:09 2005
# Perl Version: /usr/bin/perl 5.6.1 on solaris
# For :
#-----
# Options:
# 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 => int 0;
    use constant _I_PRODUCT_CODE => int 1;
    use constant _I_SALES_TOTAL => int 2;
    use constant _I_LOCATION_DESC => int 3;
    use constant _I_DESCRIPTION => int 4;
    use constant _O_LOCATION => int 1;
    use constant _O_DESCRIPTION => int 2;
    use constant _O_SALES_TOTAL => int 3;
    use constant _T_LOC_DESCRIPT_FLD_1 => int 0;
    use constant _I_LOC_DESCRIPT_LOCATION_FLD_KEY => int 5;
    use constant _I_LOC_DESCRIPT_LOCATION_FLD_1 => int 6;
    local $\\="\\n";
    local $,="|";
    print STDERR '[examples/copy_output.pql ' . localtime() . "] Init";
    use constant VERBOSE => int 10000;
    use constant LAST_ICELL => int 4;
    my @I_VAL;
    my @O_VAL;
    my $_inprec=0;
    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(READ_CHAIN_PEQUEL_PT1, '-|') == 0) # Fork -- read from child
    {
        &p_read_chain_pequel_pt1::read_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);
    }
}

```

```

}

print STDERR '[examples/copy_output.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<READ_CHAIN_PQUEL_PT1>)
{
    ++$_inprec;
    print STDERR '[examples/copy_output.pql ' . localtime() . "] $_inprec records." if ($_inprec % V
ERBOSE == 0);
    chomp;
    @I_VAL = split("||", $_);
    next unless ($I_VAL[_I_LOCATION] eq 'WA' || $I_VAL[_I_LOCATION] eq 'SA' || $I_VAL[_I_LOCATION] eq
'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 $_;
        next;
    }

    if (($I_VAL[_I_LOCATION] eq 'SA'))
    {
        print DIVERT_INPUT_COPY_OUTPUT_SA $_;
        next;
    }

    if (($I_VAL[_I_LOCATION] eq 'NSW'))
    {
        print DIVERT_INPUT_COPY_OUTPUT_NSW $_;
        next;
    }

    if (($I_VAL[_I_LOCATION] eq 'VIC'))
    {
        print DIVERT_INPUT_COPY_OUTPUT_VIC $_;
        next;
    }

    if (($I_VAL[_I_LOCATION] eq 'NT'))
    {
        print DIVERT_INPUT_COPY_OUTPUT_NT $_;
        next;
    }

    $key__I_LOCATION = $I_VAL[_I_LOCATION];
    if (!defined($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
            $O_VAL[_O_LOCATION],
            $O_VAL[_O_DESCRIPTION],
            $O_VAL[_O_SALES_TOTAL]
        if
        (
            $O_VAL[_O_SALES_TOTAL] > 0
        );
        flock(STDOUT, LOCK_UN);
        $previous_key__I_LOCATION = $key__I_LOCATION;
        @O_VAL = undef;
    }

    $I_VAL[_I_LOCATION_DESC] = $$TABLE_LOC_DESCRIPTOR{qq{$I_VAL[_I_LOCATION]}};
    $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION_DESC];
    $I_VAL[_I_DESCRIPTION] = 'State Total';
    $O_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],
    $O_VAL[_O_DESCRIPTION],
    $O_VAL[_O_SALES_TOTAL]
if
(
    $O_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);
close(READ_CHAIN_PEQUEL_PT1);
print STDERR '[examples/copy_output.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/copy_output.pql ' . localtime() . "] Code statistics: @{{timestr($benchmark_ti
mediff)}}";
#-----
#      ++++++ Table LOC_DESCRIPTOR --> Type :ETL::Pequel::Type::Table::Local ++++++
sub InitLookupLOC_DESCRIPTOR
{
    my %_TABLE_LOC_DESCRIPTOR;
    %_TABLE_LOC_DESCRIPTOR =
    (
        'NSW' => 'New South Wales',
        'NT'  => 'Northern Territory',
        'QLD' => 'Queensland',
        'SA'  => 'South Australia',
        'VIC' => 'Victoria',
        'WA'  => 'Western Australia'
    );
    return \%_TABLE_LOC_DESCRIPTOR;
}

}

{
    package p_divert_input_copy_output_sa;
    sub divert_input_copy_output_sa
    {
        #!/usr/bin/perl
#-----
#      vim: syntax=perl ts=4 sw=4
#-----
#      Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
#                  : http://sourceforge.net/projects/pequel/
#      Script Name : copy_output_SA.pql
#      Created On  : Wed Nov 16 14:13:03 2005
#      Perl Version: /usr/bin/perl 5.6.1 on solaris
#      For         :
#-----
#      Options:
#      optimize(1) optimize generated code.
#      hash(1) Generate in memory. Input data can be unsorted.
#      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    2;
        use constant _I_LOCATION_NAME => int    3;
        use constant _O_LOCATION_NAME => int    1;
        use constant _O_PRODUCT_CODE  => int    2;
        use constant _O_SALES_TOTAL   => int    3;
        local $\\="\\n";
        local $,="|";
        print STDERR '[examples/copy_output_SA.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my %O_VAL;
        my $key;
        my $_inprec=0;
        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);
        }

        print STDERR '[examples/copy_output_SA.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<STDIN>)
        {
            ++$_inprec;
            print STDERR '[examples/copy_output_SA.pql ' . localtime() . "] $_inprec records." if ($_inprec
% VERBOSE == 0);

```

```

chomp;
@I_VAL = split("[|]", $_);
$key = ( $I_VAL[_I_PRODUCT_CODE] );
$I_VAL[_I_LOCATION_NAME] = 'South Australia';
$O_VAL{$key}{_O_LOCATION_NAME} = $I_VAL[_I_LOCATION_NAME];
$O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
$O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
}

foreach $key (sort keys %O_VAL)
{
    flock(STDOUT, LOCK_EX);
    print STDOUT
        $O_VAL{$key}{_O_LOCATION_NAME},
        $O_VAL{$key}{_O_PRODUCT_CODE},
        $O_VAL{$key}{_O_SALES_TOTAL}
    ;
    flock(STDOUT, LOCK_UN);
    if ($O_VAL{$key}{_O_SALES_TOTAL} > 0)
    {
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
        print COPY_OUTPUT_COPY_OUTPUT_COMBINER
            $O_VAL{$key}{_O_LOCATION_NAME},
            $O_VAL{$key}{_O_PRODUCT_CODE},
            $O_VAL{$key}{_O_SALES_TOTAL}
        ;
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
    }
}

close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
close(STDIN);
print STDERR '[examples/copy_output_SA.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/copy_output_SA.pql ' . localtime() . "] Code statistics: @{"timestr($benchmark
_timediff)"}]";
#-----
}

}

{
    package p_read_chain_pequel_pt1;
    sub read_chain_pequel_pt1
    {
        # !/usr/bin/perl
        #-----
        # vim: syntax=perl ts=4 sw=4
        #-----
        # Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
        # : http://sourceforge.net/projects/pequel/
        # Script Name : chain_pequel_pt1.pql
        # Created On : Wed Nov 16 14:12:58 2005
        # Perl Version: /usr/bin/perl 5.6.1 on solaris
        # For :
        #-----
        # Options:
        # input_file(sample.data) input data filename
        # optimize(1) optimize generated code.
        # 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 0;
        use constant _I_COST_PRICE => int 1;
        use constant _I_DESCRIPTION => int 2;
        use constant _I_SALES_CODE => int 3;
        use constant _I_SALES_PRICE => int 4;
        use constant _I_SALES_QTY => int 5;
        use constant _I_SALES_DATE => int 6;
        use constant _I_LOCATION => int 7;
        use constant _I_SALES_TOTAL => int 8;
        use constant _O_LOCATION => int 1;
        use constant _O_PRODUCT_CODE => int 2;
        use constant _O_SALES_TOTAL => int 3;
        local $\\="\n";
        local $,="|";
        print STDERR '[examples/chain_pequel_pt1.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 8;
        my @I_VAL;
        my @O_VAL;

```

```

my $_inprec=0;
my $key__I_LOCATION;
my $previous_key__I_LOCATION = undef;
my $key__I_PRODUCT_CODE;
my $previous_key__I_PRODUCT_CODE = undef;
foreach my $f (1..3) { $O_VAL[$f] = undef; }
# Sort:LOCATION(asc:string) PRODUCT_CODE(asc:string)
open(DATA, q{sort -t'|' -y -k 8,8 -k 1,1 examples/sample.data 2>/dev/null |});
open(STDOUT, '|-', q{sort -t'|' -y -k 1,1 2>/dev/null});
print STDERR "[examples/chain_pequel_ptl.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<DATA>)
{
    ++$_inprec;
    print STDERR "[examples/chain_pequel_ptl.pql ' . localtime() . "] $_inprec records." if ($_inprec
s % VERBOSE
== 0);
    chomp;
    @I_VAL = split("[|]", $_);
    $key__I_LOCATION = $I_VAL[_I_LOCATION];
    $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
    if (!defined($previous_key__I_LOCATION) || !defined($previous_key__I_PRODUCT_CODE))
    {
        $previous_key__I_LOCATION = $key__I_LOCATION;
        $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
    }

    elsif ($previous_key__I_LOCATION ne $key__I_LOCATION || $previous_key__I_PRODUCT_CODE ne $key__I_P
RODUCT_CODE)
    {
        print STDOUT
            $O_VAL[_O_LOCATION],
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
        $previous_key__I_LOCATION = $key__I_LOCATION;
        $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
        @O_VAL = undef;
    }

    $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
    $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
    $I_VAL[_I_SALES_TOTAL] = $I_VAL[_I_SALES_QTY] * $I_VAL[_I_SALES_PRICE];
    $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
}

print STDOUT
    $O_VAL[_O_LOCATION],
    $O_VAL[_O_PRODUCT_CODE],
    $O_VAL[_O_SALES_TOTAL]
;
close(STDOUT);
close(DATA);
print STDERR "[examples/chain_pequel_ptl.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR "[examples/chain_pequel_ptl.pql ' . localtime() . "] Code statistics: @{{timestr($benchma
rk_timediff)}}";
#-----
}

{
    package p_divert_input_copy_output_wa;
    sub divert_input_copy_output_wa
    {
        #!/usr/bin/perl
        #-----
        # vim: syntax=perl ts=4 sw=4
        #-----
        # Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
        # : http://sourceforge.net/projects/pequel/
        # Script Name : copy_output_WA.pql
        # Created On : Wed Nov 16 14:13:02 2005
        # Perl Version: /usr/bin/perl 5.6.1 on solaris
        # For :
        #-----
        # Options:
        # optimize(1) optimize generated code.
        # hash(1) Generate in memory. Input data can be unsorted.
        # 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    2;
use constant _I_LOCATION_NAME => int    3;
use constant _O_LOCATION_NAME => int    1;
use constant _O_PRODUCT_CODE  => int    2;
use constant _O_SALES_TOTAL   => int    3;
local $\\="\\n";
local $,="|";
print STDERR '[examples/copy_output_WA.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 3;
my @I_VAL;
my %O_VAL;
my $key;
my $_inprec=0;
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);
}

print STDERR '[examples/copy_output_WA.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<STDIN>)
{
    ++$_inprec;
    print STDERR '[examples/copy_output_WA.pql ' . localtime() . "] $_inprec records." if ($_inprec
% VERBOSE == 0);
    chomp;
    @I_VAL = split("[|]", $_);
    $key = ( $I_VAL[_I_PRODUCT_CODE] );
    $I_VAL[_I_LOCATION_NAME] = 'Western Australia';
    $O_VAL{$key}{_O_LOCATION_NAME} = $I_VAL[_I_LOCATION_NAME];
    $O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
    $O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
}

foreach $key (sort keys %O_VAL)
{
    flock(STDOUT, LOCK_EX);
    print STDOUT
        $O_VAL{$key}{_O_LOCATION_NAME},
        $O_VAL{$key}{_O_PRODUCT_CODE},
        $O_VAL{$key}{_O_SALES_TOTAL}
    ;
    flock(STDOUT, LOCK_UN);
    if ($O_VAL{$key}{_O_SALES_TOTAL} > 0)
    {
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
        print COPY_OUTPUT_COPY_OUTPUT_COMBINER
            $O_VAL{$key}{_O_LOCATION_NAME},
            $O_VAL{$key}{_O_PRODUCT_CODE},
            $O_VAL{$key}{_O_SALES_TOTAL}
        ;
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
    }
}

close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
close(STDIN);
print STDERR '[examples/copy_output_WA.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/copy_output_WA.pql ' . localtime() . "] Code statistics: @{{timestr($benchmark
_timediff)}}";
#-----
}

{
    package p_divert_input_copy_output_nsw;
    sub divert_input_copy_output_nsw
    {
        #!/usr/bin/perl
        #-----
        # vim: syntax=perl ts=4 sw=4
        #-----
        # Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
        # : http://sourceforge.net/projects/pequel/
    }
}

```



```

# Script Name : copy_output_NSW.pql
# Created On  : Wed Nov 16 14:13:05 2005
# Perl Version: /usr/bin/perl 5.6.1 on solaris
# For        :
#-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
# Options:
#   optimize(1) optimize generated code.
#   hash(1) Generate in memory. Input data can be unsorted.
#   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    2;
use constant _I_LOCATION_NAME => int    3;
use constant _O_LOCATION_NAME => int    1;
use constant _O_PRODUCT_CODE  => int    2;
use constant _O_SALES_TOTAL   => int    3;
local $\\="\n";
local $,="|";
print STDERR "[examples/copy_output_NSW.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 3;
my @I_VAL;
my %O_VAL;
my $key;
my $_inprec=0;
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);
}

print STDERR "[examples/copy_output_NSW.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<STDIN)
{
    ++$_inprec;
    print STDERR "[examples/copy_output_NSW.pql ' . localtime() . "] $_inprec records." if ($_inprec
% VERBOSE == 0);
    chomp;
    @I_VAL = split("[|]", $_);
    $key = ( $I_VAL[_I_PRODUCT_CODE] );
    $I_VAL[_I_LOCATION_NAME] = 'New South Wales';
    $O_VAL{$key}{_O_LOCATION_NAME} = $I_VAL[_I_LOCATION_NAME];
    $O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
    $O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
}

foreach $key (sort keys %O_VAL)
{
    flock(STDOUT, LOCK_EX);
    print STDOUT
        $O_VAL{$key}{_O_LOCATION_NAME},
        $O_VAL{$key}{_O_PRODUCT_CODE},
        $O_VAL{$key}{_O_SALES_TOTAL}
    ;
    flock(STDOUT, LOCK_UN);
    if ($O_VAL{$key}{_O_SALES_TOTAL} > 0)
    {
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
        print COPY_OUTPUT_COPY_OUTPUT_COMBINER
            $O_VAL{$key}{_O_LOCATION_NAME},
            $O_VAL{$key}{_O_PRODUCT_CODE},
            $O_VAL{$key}{_O_SALES_TOTAL}
        ;
        flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
    }
}

close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
close(STDIN);
print STDERR "[examples/copy_output_NSW.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR "[examples/copy_output_NSW.pql ' . localtime() . "] Code statistics: @[timestr($benchmark
k_timediff)]";
#-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
}

```

16 November 2005 14:13

```

    }

    close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
    close(STDIN);
    print STDERR '[examples/copy_output_VIC.pql ' . localtime() . "] $_inprec records.";
    my $benchmark_end = new Benchmark;
    my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
    print STDERR '[examples/copy_output_VIC.pql ' . localtime() . "] Code statistics: @{" . timestr($benchmark_end - $benchmark_start) . "}";
}

}

{
    package p_divert_input_copy_output_nt;
    sub divert_input_copy_output_nt
    {
        # !/usr/bin/perl
        # vim: syntax=perl ts=4 sw=4
        # Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
        # : http://sourceforge.net/projects/pequel/
        # Script Name : copy_output_NT.pql
        # Created On : Wed Nov 16 14:13:09 2005
        # Perl Version: /usr/bin/perl 5.6.1 on solaris
        # For :
        # Options:
        # optimize(1) optimize generated code.
        # hash(1) Generate in memory. Input data can be unsorted.
        # 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 2;
        use constant _I_LOCATION_NAME => int 3;
        use constant _O_LOCATION_NAME => int 1;
        use constant _O_PRODUCT_CODE => int 2;
        use constant _O_SALES_TOTAL => int 3;
        local $\\ = "\n";
        local $, = "|";
        print STDERR '[examples/copy_output_NT.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my %O_VAL;
        my $key;
        my $_inprec=0;
        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);
        }

        print STDERR '[examples/copy_output_NT.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<STDIN>)
        {
            ++$_inprec;
            print STDERR '[examples/copy_output_NT.pql ' . localtime() . "] $_inprec records." if ($_inprec
% VERBOSE == 0);
            chomp;
            @I_VAL = split("||", $_);
            $key = ( $I_VAL[_I_PRODUCT_CODE] );
            $I_VAL[_I_LOCATION_NAME] = 'Northern Territory';
            $O_VAL{$key}{_O_LOCATION_NAME} = $I_VAL[_I_LOCATION_NAME];
            $O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
        }

        foreach $key (sort keys %O_VAL)
        {
            flock(STDOUT, LOCK_EX);
            print STDOUT
                $O_VAL{$key}{_O_LOCATION_NAME},
                $O_VAL{$key}{_O_PRODUCT_CODE},
                $O_VAL{$key}{_O_SALES_TOTAL}
        }
    }
}

```

```

flock(STDOUT, LOCK_UN);
if ($O_VAL{$key}{_O_SALES_TOTAL} > 0)
{
    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
    print COPY_OUTPUT_COPY_OUTPUT_COMBINER
        $O_VAL{$key}{_O_LOCATION_NAME},
        $O_VAL{$key}{_O_PRODUCT_CODE},
        $O_VAL{$key}{_O_SALES_TOTAL}
    ;
    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
}

}

close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
close(STDIN);
print STDERR '[examples/copy_output_NT.pql ' . localtime() . "] $_inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/copy_output_NT.pql ' . localtime() . "] Code statistics: @{"timestr($benchmark
_timediff)}]";
#-----
}

}

{
    package p_copy_output_copy_output_combiner;
    sub copy_output_copy_output_combiner
    {
        #!/usr/bin/perl
        #-----
        # vim: syntax=perl ts=4 sw=4
        #-----
        # Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
        # : http://sourceforge.net/projects/pequel/
        # Script Name : copy_output_combiner.pql
        # Created On : Wed Nov 16 14:13:01 2005
        # Perl Version: /usr/bin/perl 5.6.1 on solaris
        # 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 0;
        use constant _I_PRODUCT_CODE => int 1;
        use constant _I_SALES_TOTAL => int 2;
        use constant _I_DESCRIPTION => int 3;
        use constant _O_LOCATION_NAME => int 1;
        use constant _O_DESCRIPTION => int 2;
        use constant _O_SALES_TOTAL => int 3;
        local $\\="\\n";
        local $,="|";
        print STDERR '[examples/copy_output_combiner.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 1000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my @O_VAL;
        my $_inprec=0;
        my $key__I_LOCATION_NAME;
        my $previous_key__I_LOCATION_NAME = undef;
        foreach my $f (1..3) { $O_VAL[$f] = undef; }
        # Sort:LOCATION_NAME(asc:string)
        open(DATA, q{cat - | sort -t'|' -y -k 1,1 2>/dev/null |}) || die "Cannot open input: $!";
        print STDERR '[examples/copy_output_combiner.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<DATA>)
        {
            ++$_inprec;
            print STDERR '[examples/copy_output_combiner.pql ' . localtime() . "] $_inprec records." if ($_in
            prec % VERBOSE == 0);
            chomp;
            @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;
            }
        }
    }
}

```

```

elseif ($previous_key__I_LOCATION_NAME ne $key__I_LOCATION_NAME)
{
    flock(STDOUT, LOCK_EX);
    print STDOUT
        $O_VAL[_O_LOCATION_NAME],
        $O_VAL[_O_DESCRIPTION],
        $O_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';
$O_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],
    $O_VAL[_O_DESCRIPTION],
    $O_VAL[_O_SALES_TOTAL]
;
flock(STDOUT, LOCK_UN);
close(DATA);
print STDERR '[examples/copy_output_combiner.pql ' . localtime() . "] $inprec records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/copy_output_combiner.pql ' . localtime() . "] Code statistics: @{{timestr($ben
chmark_timediff)}}";
#-+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
}
}

```

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

