



examples/unpack_input.pql

by *Pequel*

sample@youraddress.com

Unpack Input Example Script

2.3

Table of Contents

Unpack Input Example Script

SCRIPT NAME	1
DESCRIPTION	1
1. PROCESS DETAILS	1
1.1 LOCATION	1
Description	1
1.2 PRODUCT_COUNT	1
Description	1
1.3 MIN_SALES_QTY	1
Description	1
1.4 MAX_SALES_QTY	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 header	2
2.8 optimize	2
2.9 hash	2
2.10 nulls	2
2.11 doc_title	2
2.12 doc_email	2
2.13 doc_version	2
2.14 discard_header	2
2.15 unpack_input	2
2.16 input_pack_fmt	3
3. TABLES	4
4. TABLE INFORMATION SUMMARY	5
4.1 Table List Sorted By Table Name	5
5. EXAMPLES/UNPACK_INPUT.PQL	6
options	6
description	6
input section	6
group by	6
output section	6
6. PEQUEL GENERATED PROGRAM	7
7. ABOUT PEQUEL	9
COPYRIGHT	9

SCRIPT NAME

examples/unpack_input.pql

DESCRIPTION

Demonstrates unpacking packed or fixed input data stream records. Use the 'unpack_input' and 'input_pack_fmt' options together to specify packed/fixed record input data stream. The formats specified in 'input_pack_fmt' are the same as for the Perl pack/unpack functions. The format may be enclosed in square brackets to indicate that this is a repeat format — that is the format will be used for each field — so in this example the '[A3/Z*]' will expand to 'A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*' to reflect the 7 field input record format. Please refer to the Perl perlpacktut manual for format definitions. The 'unpack_input' option may not be used when the 'sort by' section is used unless that 'hash' option is also specified.

1. PROCESS DETAILS

Input records are read from sample_fixed.dat. The input record contains **3** fields. Fields are delimited by the '|' character.

Output records are written to standard output. The output record contains **4** fields. Fields are delimited by the '|' character.

Input records are **grouped** by the input field **LOCATION** (*string*).

1.1 LOCATION

Output Field

Description

Set to input field **LOCATION**

1.2 PRODUCT_COUNT

Output Field

Description

Distinct aggregation on input field **PRODUCT_CODE**.

1.3 MIN_SALES_QTY

Output Field

Description

Min aggregation on input field **SALES_QTY**.

1.4 MAX_SALES_QTY

Output Field

Description

Max aggregation on input field **SALES_QTY**.

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/unpack_input.pql

2.6 *input_file*

input data filename: sample_fixed.dat

2.7 *header*

write header record to output.: 1

2.8 *optimize*

optimize generated code.: 1

2.9 *hash*

Generate in memory. Input data can be unsorted.: 1

2.10 *nulls*

print zero for null numeric/decimal.: 1

2.11 *doc_title*

document title.: Unpack Input Example Script

2.12 *doc_email*

document email entry.: sample@youraddress.com

2.13 *doc_version*

document version for pequel script.: 2.3

2.14 *discard_header*

Input file has header record - must be discarded.: 1

2.15 *unpack_input*

Unpack input data stream: 1

2.16 *input_pack_fmt*

Pack format for input data stream: [A3/Z*]

3. TABLES

4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

5. EXAMPLES/UNPACK_INPUT.PQL

options

```

pequeldoc(pdf)
detail(1)
noverbose(1)
prefix(examples)
script_name(examples/unpack_input.pql)
input_file(sample_fixed.dat)
header(1)
optimize(1)
hash(1)
nulls(1)
doc_title(Unpack Input Example Script)
doc_email(sample@youraddress.com)
doc_version(2.3)
discard_header(1)
unpack_input(1)
input_pack_fmt([A3/Z*])

```

description

Demonstrates unpacking packed or fixed input data stream records. Use the 'unpack_input' and 'input_pack_fmt' options together to specify packed/fixed record input data stream. The formats specified in 'input_pack_fmt' are the same as for the Perl pack/unpack functions. The format may be enclosed in square brackets to indicate that this is a repeat format -- that is the format will be used for each field -- so in this example the '[A3/Z*]' will expand to 'A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*A3/Z*' to reflect the 7 field input record format. Please refer to the Perl perlpacktut manual for format definitions. The 'unpack_input' option may not be used when the 'sort by' section is used unless that 'hash' option is also specified.

input section

```

LOCATION
PRODUCT_CODE
SALES_QTY

```

group by

```

LOCATION string

```

output section

```

string      LOCATION      LOCATION
numeric     PRODUCT_COUNT distinct PRODUCT_CODE
decimal     MIN_SALES_QTY min SALES_QTY
decimal     MAX_SALES_QTY max SALES_QTY

```

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/unpack_input.pql
#Created On  : Tue Oct 25 09:24:49 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/unpack_input.pql) script filename
#input_file(sample_fixed.dat) input data filename
#header(1) write header record to output.
#optimize(1) optimize generated code.
#hash(1) Generate in memory. Input data can be unsorted.
#nulls(1) print zero for null numeric/decimal.
#doc_title(Unpack Input Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.3) document version for pequel script.
#discard_header(1) Input file has header record - must be discarded.
#unpack_input(1) Unpack input data stream
#input_pack_fmt([A3/Z*]) Pack format for input data stream
#-----
use strict;
use constant _I_LOCATION      => int    0;
use constant _I_PRODUCT_CODE  => int    1;
use constant _I_SALES_QTY     => int    2;
use constant _O_LOCATION      => int    1;
use constant _O_PRODUCT_COUNT => int    2;
use constant _O_MIN_SALES_QTY => int    3;
use constant _O_MAX_SALES_QTY => int    4;
local $\\="\\n";
local $,="|";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 2;
use constant INPUT_PACK_FMT => 'A3/Z*' x (LAST_ICELL+1);
my @I_VAL;
my %O_VAL;
my $key;
my %DISTINCT;
open(DATA, q{examples/sample_fixed.dat}) || die "Cannot open examples/sample_fixed.dat: $!";
&PrintHeader();
my $discard_header = <DATA>;
while (<DATA>)
{
    chomp;
    @I_VAL = unpack(INPUT_PACK_FMT, $_);
    $key = ( $I_VAL[_I_LOCATION] );
    $O_VAL{$key}{_O_LOCATION} = $I_VAL[_I_LOCATION];
    $O_VAL{$key}{_O_PRODUCT_COUNT}++
    if (defined($I_VAL[_I_PRODUCT_CODE]) && ++$DISTINCT{$key}{_O_PRODUCT_COUNT}{qq{$I_VAL[_I_PRODUCT_CODE]
}} == 1);
    $O_VAL{$key}{_O_MIN_SALES_QTY} = $I_VAL[_I_SALES_QTY]
    if (!defined($O_VAL{$key}{_O_MIN_SALES_QTY}) || $I_VAL[_I_SALES_QTY] < $O_VAL{$key}{_O_MIN_SALES_QTY})
;
    $O_VAL{$key}{_O_MAX_SALES_QTY} = $I_VAL[_I_SALES_QTY]
    if (!defined($O_VAL{$key}{_O_MAX_SALES_QTY}) || $I_VAL[_I_SALES_QTY] > $O_VAL{$key}{_O_MAX_SALES_QTY})
;
}

foreach $key (sort keys %O_VAL)
{
    print STDOUT
        $O_VAL{$key}{_O_LOCATION},
        $O_VAL{$key}{_O_PRODUCT_COUNT},
        $O_VAL{$key}{_O_MIN_SALES_QTY},
        $O_VAL{$key}{_O_MAX_SALES_QTY}
    ;
}

#-----
sub PrintHeader
{
    local $\\="\\n";
    local $,="|";
    print
        'LOCATION',
```

```
        'PRODUCT_COUNT' ,  
        'MIN_SALES_QTY' ,  
        'MAX_SALES_QTY' '  
    ;  
}
```

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

