oracle_table.pql

by *Pequel*

sample@youraddress.com

Oracle Tables Example Script

2.2

Table of Contents
Oracle Tables Example Script

SCRIPT NAME

oracle_table.pql

DESCRIPTION

Demonstrates the use of external Oracle tables. WARNING: this feature is alpha and would (probably) require some hand coding adjustments to the generated code. Requires Inline::C and DBI to be installed. The 'load table oracle' section will load the ASCII data contained in the file specified by the second parameter ('sample.data' inexample SAMPLE1 below) into an oracle table. The generated inline C code will access this table via Oracle OCI. The Oracle table will be re-created with the same name as specified by the first parameter('SAMPLE1' in this example). The data will be loaded via Oracle sqlldr. The 4th parameter KeyLoc specifies the location of the key field in sample.data (field numbers starting from 1). The next parameter KeyType specifies the Oracle type and size to use when creating the table. The Columns list specifies field and field-number (in the SourceData file) pairs. The 'merge' option can be used when the table is sorted by the same key as specified in the 'sort by' section. This will result in a substantial performance gain when looking up values in the table.

1. PROCESS DETAILS

Input records are read from standard input. The input record contains ***8*** fields. Fields are delimited by the '|' character.

Output records are written to standard output. The output record contains ***8*** fields. Fields are delimited by the '|' character.

Input stream is **sorted** by the input field ***PRODUCT_CODE*** (*string*).

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

1.1 *PRODUCT_CODE*

Output Field

Description
Set to input field ***PRODUCT_CODE***

1.2 *RECORD_COUNT*

Output Field

Description
Count aggregation.

1.3 *SALES_QTY_SAMPLE1*

Output Field

Description
Sum aggregation on input field ***SALES_QTY***.

Aggregation condition
exists %SAMPLE1(PRODUCT_CODE);

1.4 *S1_DESCRIPTION*

Output Field

Description

Set to input field ***S1_DESCRIPTION***

Derived Input Field Evaluation

```
=> %SAMPLE1(PRODUCT_CODE)->DESCRIPTION
```

1.5 *S1_LOCATION*

Output Field

Description

Set to input field ***S1_LOCATION***

Derived Input Field Evaluation

```
=> %SAMPLE1(PRODUCT_CODE)->LOCATION
```

1.6 *SALES_QTY_SAMPLE2*

Output Field

Description

Sum aggregation on input field ***SALES_QTY***.

Aggregation condition

exists %SAMPLE2(PRODUCT_CODE);

1.7 *S2_DESCRIPTION*

Output Field

Description

Set to input field ***S2_DESCRIPTION***

Derived Input Field Evaluation

```
=> %SAMPLE2(PRODUCT_CODE)->DESCRIPTION
```

1.8 *S2_LOCATION*

Output Field

Description

Set to input field ***S2_LOCATION***

Derived Input Field Evaluation

```
=> %SAMPLE2(PRODUCT_CODE)->LOCATION
```

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 *script_name*
script filename: oracle_table.pql

2.4 *header*
write header record to output.: 1

2.5 *optimize*
optimize generated code.: 1

2.6 *doc_title*
document title.: Oracle Tables Example Script

2.7 *doc_email*
document email entry.: sample@youraddress.com

2.8 *doc_version*
document version for pequel script.: 2.2

2.9 *inline_cc*
Inline: CC: CC

2.10 *inline_force_build*
Inline: force_build: 1

2.11 *inline_optimize*
Inline: OPTIMIZE: -xO5 -xinline=%auto

2.12 *inline_ccflags*
Inline: CCFLAGS: -xchip=ultra3 -DSS_64BIT_SERVER -DBIT64 -DMACHINE64

3. TABLES

3.1 *SAMPLE1*

Table Type: ***oracle***
Data Source Filename: ***sample.data***
Key Field Number: ***1***
Key Field Type: ***STRING(12)***
Database Filename: ***/_TABLE_SAMPLE1.oracle***

3.1.1 *DESCRIPTION = 3*
3.1.2 *LOCATION = 8*

3.2 *SAMPLE2*

Table Type: ***oracle merge***
Data Source Filename: ***sample.data***
Key Field Number: ***1***
Key Field Type: ***STRING(12)***
Database Filename: ***/_TABLE_SAMPLE2.oracle***

3.2.1 *DESCRIPTION = 3*
3.2.2 *LOCATION = 8*

4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

SAMPLE1 — ***1*** (*oracle*)
SAMPLE2 — ***2*** (*oracle merge*)

5. ORACLE_TABLE.PQL

options

```
pequeldoc(pdf)
detail(1)
script_name(oracle_table.pql)
header(1)
optimize(1)
doc_title(Oracle Tables Example Script)
doc_email(sample@youraddress.com)
doc_version(2.2)
inline_cc(CC)
inline_force_build(1)
inline_optimize(-xO5 -xinline=%auto)
inline_ccflags(-xchip=ultra3 -DSS_64BIT_SERVER -DBIT64 -DMACHINE64)
```

description

```
Demonstrates the use of external Oracle tables. WARNING: this feature is alpha and would
(probably) require some hand coding adjustments to the generated code.
Requires Inline::C and DBI to be installed.
The 'load table oracle' section will load the ASCII data contained in the file specified by
the second parameter ('sample.data' inexample SAMPLE1 below) into an oracle table. The
generated inline C code will access this table via Oracle OCI. The Oracle table will be
re-created with the same name as specified by the first parameter('SAMPLE1' in this example).
The data will be loaded via Oracle sqlldr. The 4th parameter KeyLoc specifies the location
of the key field in sample.data (field numbers starting from 1). The next parameter KeyType
specifies the Oracle type and size to use when creating the table. The Columns list specifies
field and field-number (in the SourceData file) pairs. The 'merge' option can be used when the
table is sorted by the same key as specified in the 'sort by' section. This will result in a
substantial performance gain when looking up values in the table.
```

load table oracle

```
SAMPLE1 /* Table Name */ \
    sample.data /* Data Source Filename */ \
    1 /* Key Column Number */ \
    STRING(12) /* Key Type */ \
    DESCRIPTION = 3 \
    LOCATION = 8
```

load table oracle merge

```
SAMPLE2 /* Table Name */ \
    sample.data /* Data Source Filename */ \
    1 /* Key Column Number */ \
    STRING(12) /* Key Type */ \
    DESCRIPTION = 3 \
    LOCATION = 8
```

sort by

```
PRODUCT_CODE string
```

group by

```
PRODUCT_CODE string
```

input section

```
PRODUCT_CODE
COST_PRICE
DESCRIPTION
SALES_CODE
SALES_PRICE
SALES_QTY
SALES_DATE
LOCATION
S1_DESCRIPTION => %SAMPLE1(PRODUCT_CODE)->DESCRIPTION

S1_LOCATION => %SAMPLE1(PRODUCT_CODE)->LOCATION

S2_DESCRIPTION => %SAMPLE2(PRODUCT_CODE)->DESCRIPTION

S2_LOCATION => %SAMPLE2(PRODUCT_CODE)->LOCATION
```

output section

```
string    PRODUCT_CODE      PRODUCT_CODE
numeric   RECORD_COUNT      count *
numeric   SALES_QTY_SAMPLE1 sum SALES_QTY where exists %SAMPLE1(PRODUCT_CODE)
string    S1_DESCRIPTION    S1_DESCRIPTION
string    S1_LOCATION       S1_LOCATION
numeric   SALES_QTY_SAMPLE2 sum SALES_QTY where exists %SAMPLE2(PRODUCT_CODE)
string    S2_DESCRIPTION    S2_DESCRIPTION
string    S2_LOCATION       S2_LOCATION
```

6. PEQUEL GENERATED PROGRAM

```
# vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Generated By: pequel Version 2.3-2, Build: Thursday September 29 19:56:03 BST 2005
#            : https://sourceforge.net/projects/pequel/
#Script Name : oracle_table.pql
#Created On  : Thu Sep 29 14:05:14 2005
#For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Options:
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
#script_name(oracle_table.pql) script filename
#header(1) write header record to output.
#optimize(1) optimize generated code.
#doc_title(Oracle Tables Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.2) document version for pequel script.
#inline_cc(CC) Inline: CC
#inline_force_build(1) Inline: force_build
#inline_optimize(-xO5 -xinline=%auto) Inline: OPTIMIZE
#inline_ccflags(-xchip=ultra3 -DSS_64BIT_SERVER -DBIT64 -DMACHINE64) Inline: CCFLAGS
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
use strict;
use Fcntl;
local $\="\n"; local $,="|";
print STDERR '[oracle_table.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 11;
my @I_VAL;
my @O_VAL;
my $key__I_PRODUCT_CODE;
my $previous_key__I_PRODUCT_CODE = undef;
foreach my $f (1..8) { $O_VAL[$f] = undef; }
&LoadTableSAMPLE2; # Create database for SAMPLE2
&LoadTableSAMPLE1; # Create database for SAMPLE1
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_S1_DESCRIPTION       => int    8;
use constant _I_S1_LOCATION          => int    9;
use constant _I_S2_DESCRIPTION       => int   10;
use constant _I_S2_LOCATION          => int   11;
use constant _O_PRODUCT_CODE         => int    1;
use constant _O_RECORD_COUNT         => int    2;
use constant _O_SALES_QTY_SAMPLE1    => int    3;
use constant _O_S1_DESCRIPTION       => int    4;
use constant _O_S1_LOCATION          => int    5;
use constant _O_SALES_QTY_SAMPLE2    => int    6;
use constant _O_S2_DESCRIPTION       => int    7;
use constant _O_S2_LOCATION          => int    8;
use constant _T_SAMPLE2_FLD_DESCRIPTION   => int    0;
use constant _T_SAMPLE2_FLD_LOCATION      => int    1;
use constant _T_SAMPLE1_FLD_DESCRIPTION   => int    0;
use constant _T_SAMPLE1_FLD_LOCATION      => int    1;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_KEY          => int   12;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_DESCRIPTION  => int   13;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_LOCATION     => int   14;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_KEY          => int   15;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_DESCRIPTION  => int   16;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_LOCATION     => int   17;
open(DATA, q{cat - | sort -t'|' -y -k 1,1 |}) || die "Cannot open input: $!";
my $fd = fileno(DATA);
OpenSortStream($fd);
&PrintHeader();
print STDERR '[oracle_table.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
OracleConnect($fd);
print STDERR '[oracle_table.pql ' . localtime() . "] Tables opened.";
my $i;
while (readsplit(\@I_VAL))
{
    ++$i;
    print STDERR '[oracle_table.pql ' . localtime() . "] $i records." if ($i % VERBOSE == 0);
    $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)
    {
        print
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_RECORD_COUNT],
            $O_VAL[_O_SALES_QTY_SAMPLE1],
            $O_VAL[_O_S1_DESCRIPTION],
            $O_VAL[_O_S1_LOCATION],
            $O_VAL[_O_SALES_QTY_SAMPLE2],
            $O_VAL[_O_S2_DESCRIPTION],
            $O_VAL[_O_S2_LOCATION]
        ;
        $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
        @O_VAL = undef;
    }

    $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
    $O_VAL[_O_RECORD_COUNT]++;
    $I_VAL[_I_S1_DESCRIPTION] = $I_VAL[_I_SAMPLE1_PRODUCT_CODE_FLD_DESCRIPTION];
    $O_VAL[_O_S1_DESCRIPTION] = $I_VAL[_I_S1_DESCRIPTION];
    $I_VAL[_I_S1_LOCATION] = $I_VAL[_I_SAMPLE1_PRODUCT_CODE_FLD_LOCATION];
    $O_VAL[_O_S1_LOCATION] = $I_VAL[_I_S1_LOCATION];
    $I_VAL[_I_S2_DESCRIPTION] = $I_VAL[_I_SAMPLE2_PRODUCT_CODE_FLD_DESCRIPTION];
    $O_VAL[_O_S2_DESCRIPTION] = $I_VAL[_I_S2_DESCRIPTION];
    $I_VAL[_I_S2_LOCATION] = $I_VAL[_I_SAMPLE2_PRODUCT_CODE_FLD_LOCATION];
    $O_VAL[_O_S2_LOCATION] = $I_VAL[_I_S2_LOCATION];

    if (exists $I_VAL[_I_SAMPLE1_PRODUCT_CODE_FLD_KEY] ne '') {
        $O_VAL[_O_SALES_QTY_SAMPLE1] += $I_VAL[_I_SALES_QTY] unless ($I_VAL[_I_SALES_QTY] eq '');
    }

    if (exists $I_VAL[_I_SAMPLE2_PRODUCT_CODE_FLD_KEY] ne '') {
        $O_VAL[_O_SALES_QTY_SAMPLE2] += $I_VAL[_I_SALES_QTY] unless ($I_VAL[_I_SALES_QTY] eq '');
    }
}

print
    $O_VAL[_O_PRODUCT_CODE],
    $O_VAL[_O_RECORD_COUNT],
    $O_VAL[_O_SALES_QTY_SAMPLE1],
    $O_VAL[_O_S1_DESCRIPTION],
    $O_VAL[_O_S1_LOCATION],
    $O_VAL[_O_SALES_QTY_SAMPLE2],
    $O_VAL[_O_S2_DESCRIPTION],
    $O_VAL[_O_S2_LOCATION]
;
OracleDisconnect();
print STDERR '[oracle_table.pql ' . localtime() . "] $i records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[oracle_table.pql ' . localtime() . "] Code statistics: @{[timestr($benchmark_timediff)]}";
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#++++++ Table SAMPLE1 --> Type :Pequel::Type::Table::Oracle ++++++
sub LoadTableSAMPLE1
{
    my %_TABLE_SAMPLE1;
    print STDERR '[oracle_table.pql ' . localtime() . "] Table SAMPLE1";
    print STDERR '[oracle_table.pql ' . localtime() . "] -->datasource:sample.data...";
    use DBI;
    my $password = 'gprsdev';
    if ($password eq '')
    {
        $password = `grep -iw OSCADEV2 \$HOME/.password | grep -iw gprsdev | awk '{ print \$3 }'`;
        chomp($password);
    }

    my $dbh = DBI->connect
    (
        'dbi:Oracle:OSCADEV2', 'gprsdev', $password,
        { RowCacheSize => 5000, RaiseError => 0, AutoCommit => 0 }
    )

    or die "Cannot connect to OSCADEV2:$!";
    $dbh->{PrintError} = 0;
    my @tables = $dbh->tables();
    map(s/\"//g, @tables);
    my $table_exists = grep(m/\.SAMPLE1$/i, @tables);
    if (!$table_exists)
    {
        print STDERR '[oracle_table.pql ' . localtime() . "] -->creating database table...";
```

```
        my @flist =
        (
            'key',
            'description',
            'location'
        );

        my %flist =
        (
            key => 'STRING(12) PRIMARY KEY',
            description => 'varchar2(128)',
            location => 'varchar2(128)'
        );

        $dbh->do("CREATE TABLE SAMPLE1 ( @{[ join(',', map(qq{$_ $flist{$_}}, @flist)) ]} )");
        $dbh->commit or die $dbh->errstr;
        open(CTL, '>/_TABLE_SAMPLE1.oracle.ctl');
        print CTL "LOAD";
        print CTL "append into table SAMPLE1";
        print CTL "FIELDS TERMINATED BY '|' TRAILING NULLCOLS";
        print CTL "(";
        print CTL "@{[ join(qq{,\n}, @flist) ]}";
        print CTL ")";
        close(CTL);
        system("sort -u -t'|' -k 1 | cut -d'|' -f1,3,8  sample.data > /_TABLE_SAMPLE1.oracle.dat");
        my $sqlldr =
            'control=/_TABLE_SAMPLE1.oracle.ctl '
            . 'direct=true '
            . 'data=/_TABLE_SAMPLE1.oracle.dat '
            . 'bad=/_TABLE_SAMPLE1.oracle.bad '
            . 'log=/_TABLE_SAMPLE1.oracle.log '
            . 'rows=100000';
        open(SQLLDR, ">/_TABLE_SAMPLE1.oracle.sqlldr");
        print SQLLDR "UID=gprsdev; export UID";
        ($password eq '')
            ? print SQLLDR "PWD=`grep -iw OSCADEV2 \$HOME/.password | grep -iw gprsdev | awk '{ print \$3 }'`;
 export PWD"
            : print SQLLDR "PWD=$password; export PWD";
        print SQLLDR "DB=OSCADEV2; export DB";
        print SQLLDR "sqlldr $sqlldr <<EOF";
        print SQLLDR "\$UID/\$PWD\@\$DB";
        print SQLLDR "EOF";
        close(SQLLDR);
        system("sh /_TABLE_SAMPLE1.oracle.sqlldr 2>&1 >/_TABLE_SAMPLE1.oracle.err");
    }

    my $sth = $dbh->prepare("SELECT count(1) FROM SAMPLE1");
    $sth->execute;
    my $loaded = $sth->fetchrow_array;
    print STDERR '[oracle_table.pql ' . localtime() . "] -->$loaded records.";
    $sth->finish;
    $dbh->disconnect;
}

#++++++ Table SAMPLE2 --> Type :Pequel::Type::Table::Oracle::Merge ++++++
sub LoadTableSAMPLE2
{
    my %_TABLE_SAMPLE2;
    print STDERR '[oracle_table.pql ' . localtime() . "] Table SAMPLE2";
    print STDERR '[oracle_table.pql ' . localtime() . "] -->datasource:sample.data...";
    use DBI;
    my $password = 'gprsdev';
    if ($password eq '')
    {
        $password = `grep -iw OSCADEV2 \$HOME/.password | grep -iw gprsdev | awk '{ print \$3 }'`;
        chomp($password);
    }

    my $dbh = DBI->connect
    (
        'dbi:Oracle:OSCADEV2', 'gprsdev', $password,
        { RowCacheSize => 5000, RaiseError => 0, AutoCommit => 0 }
    )

    or die "Cannot connect to OSCADEV2:$!";
    $dbh->{PrintError} = 0;
    my @tables = $dbh->tables();
    map(s/\"//g, @tables);
    my $table_exists = grep(m/\.SAMPLE2$/i, @tables);
    if (!$table_exists)
    {
        print STDERR '[oracle_table.pql ' . localtime() . "] -->creating database table...";
        my @flist =
        (
            'key',
```

```
            'description',
            'location'
        );

        my %flist =
        (
            key => 'STRING(12) ',
            description => 'varchar2(128)',
            location => 'varchar2(128)'
        );

        $dbh->do("CREATE TABLE SAMPLE2 ( @{[ join(',', map(qq{$_ $flist{$_}}, @flist)) ]} )");
        $dbh->commit or die $dbh->errstr;
        open(CTL, '>/_TABLE_SAMPLE2.oracle.ctl');
        print CTL "LOAD";
        print CTL "append into table SAMPLE2";
        print CTL "FIELDS TERMINATED BY '|' TRAILING NULLCOLS";
        print CTL "(";
        print CTL "@{[ join(qq{,\n}, @flist) ]}";
        print CTL ")";
        close(CTL);
        system("sort -u -t'|' -k 1 | cut -d'|' -f1,3,8  sample.data > /_TABLE_SAMPLE2.oracle.dat");
        my $sqlldr =
            'control=/_TABLE_SAMPLE2.oracle.ctl '
            . 'direct=true '
            . 'data=/_TABLE_SAMPLE2.oracle.dat '
            . 'bad=/_TABLE_SAMPLE2.oracle.bad '
            . 'log=/_TABLE_SAMPLE2.oracle.log '
            . 'rows=100000';
        open(SQLLDR, ">/_TABLE_SAMPLE2.oracle.sqlldr");
        print SQLLDR "UID=gprsdev; export UID";
        ($password eq '')
            ? print SQLLDR "PWD=`grep -iw OSCADEV2 \$HOME/.password | grep -iw gprsdev | awk '{ print \$3 }'`;
 export PWD"
            : print SQLLDR "PWD=$password; export PWD";
        print SQLLDR "DB=OSCADEV2; export DB";
        print SQLLDR "sqlldr $sqlldr <<EOF";
        print SQLLDR "\$UID/\$PWD\@\$DB";
        print SQLLDR "EOF";
        close(SQLLDR);
        system("sh /_TABLE_SAMPLE2.oracle.sqlldr 2>&1 >/_TABLE_SAMPLE2.oracle.err");
    }

    my $sth = $dbh->prepare("SELECT count(1) FROM SAMPLE2");
    $sth->execute;
    my $loaded = $sth->fetchrow_array;
    print STDERR '[oracle_table.pql ' . localtime() . "] -->$loaded records.";
    $sth->finish;
    $dbh->disconnect;
}

sub PrintHeader
{
    local $\="\n";
    local $,="|";
    print
        'PRODUCT_CODE',
        'RECORD_COUNT',
        'SALES_QTY_SAMPLE1',
        'S1_DESCRIPTION',
        'S1_LOCATION',
        'SALES_QTY_SAMPLE2',
        'S2_DESCRIPTION',
        'S2_LOCATION'
        ;
}

#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#***** I N L I N E *****
use Inline
    C => Config =>
    NAME => 'oracle_table',
    CC => 'CC',
    CLEAN_AFTER_BUILD => '1',
    CLEAN_BUILD_AREA => '1',
    PRINT_INFO => '0',
    BUILD_NOISY => '0',
    BUILD_TIMERS => '0',
    FORCE_BUILD => '1',
    LIBS => '  -L/opt/app/oracle/product/9.2.0/lib -L/opt/app/oracle/product/9.2.0/rdbms/lib -lpthread -lclnts
h -lc',
    INC => '  -I/opt/app/oracle/product/9.2.0/rdbms/demo -I/opt/app/oracle/product/9.2.0/rdbms/public',
    CCFLAGS => '-xchip=ultra3 -DSS_64BIT_SERVER -DBIT64 -DMACHINE64  ',
    OPTIMIZE => '-xO5 -xinline=%auto  '
    ;
```

```
use Inline C => q~

#include <oci.h>

#define GFMAXPIPBUFFER      3072
#define GFMAXPIPFLDS        13
#define GFCACHERECS         100

#define _I_PRODUCT_CODE            0
#define _I_COST_PRICE              1
#define _I_DESCRIPTION             2
#define _I_SALES_CODE              3
#define _I_SALES_PRICE             4
#define _I_SALES_QTY               5
#define _I_SALES_DATE              6
#define _I_LOCATION                7
#define _I_S1_DESCRIPTION          8
#define _I_S1_LOCATION             9
#define _I_S2_DESCRIPTION         10
#define _I_S2_LOCATION            11

#define _I_SAMPLE1_PRODUCT_CODE_FLD_KEY            12
#define _I_SAMPLE1_PRODUCT_CODE_FLD_DESCRIPTION    13
#define _I_SAMPLE1_PRODUCT_CODE_FLD_LOCATION       14
#define _I_SAMPLE2_PRODUCT_CODE_FLD_KEY            15
#define _I_SAMPLE2_PRODUCT_CODE_FLD_DESCRIPTION    16
#define _I_SAMPLE2_PRODUCT_CODE_FLD_LOCATION       17

int oracle_open_all();
int oracle_prep_all();

sword status;
text errbuf[512];
sb4 errcode =0;

#define STD_FLD_LEN 128
#define _OracleMergeFetchNumeric(tbl,ikey) \
    if (tbl##_KEY == 0 && last_step_##tbl == OCI_SUCCESS) \
    { \
        last_step_##tbl = OCIStmtFetch(stmthp_##tbl, errhp, (ub4) 1, (ub4) OCI_FETCH_NEXT, (ub4) OCI_DEFAULT);
 \
    } \
    while \
    ( \
        last_step_##tbl == OCI_SUCCESS \
        && tbl##_KEY != 0 \
        && atol(fields[current_cache_rec][ikey]) > tbl##_KEY \
        && (last_step_##tbl = OCIStmtFetch(stmthp_##tbl, errhp, (ub4) 1, (ub4) OCI_FETCH_NEXT, (ub4) OCI_DEFAU
LT)) == OCI_SUCCESS \
    ) \
    { \
        if (tbl##_KEY == 0) break; \
        if (atol(fields[current_cache_rec][ikey]) <= tbl##_KEY) break; \
    } \
    if (tbl##_KEY != 0 && atol(fields[current_cache_rec][ikey]) == tbl##_KEY)

#define _OracleMergeFetchString(tbl,ikey) \
    if (tbl##_KEY == 0 && last_step_##tbl == OCI_SUCCESS) \
    { \
        last_step_##tbl = OCIStmtFetch(stmthp_##tbl, errhp, (ub4) 1, (ub4) OCI_FETCH_NEXT, (ub4) OCI_DEFAULT);
 \
    } \
    while \
    ( \
        last_step_##tbl == OCI_SUCCESS \
        && tbl##_KEY != 0 \
        && strcmp(fields[current_cache_rec][ikey], (const char*)tbl##_KEY) > 0 \
        && (last_step_##tbl = OCIStmtFetch(stmthp_##tbl, errhp, (ub4) 1, (ub4) OCI_FETCH_NEXT, (ub4) OCI_DEFAU
LT)) == OCI_SUCCESS \
    ) \
    { \
        if (tbl##_KEY == 0) break; \
        if (strcmp(fields[current_cache_rec][ikey], (const char*)tbl##_KEY) <= 0) break; \
    } \
    if (tbl##_KEY != 0 && strcmp(fields[current_cache_rec][ikey], (const char*)tbl##_KEY) == 0)

#define _av_store_numeric(tbl,fld) \
    if (!indicator_##tbl##_##fld) \
        av_store(I_VAL, _I_##tbl##_FLD_##fld, newSVpvf("%ld", tbl##_##fld));

#define _av_store_string(tbl,fld) \
    if (!indicator_##tbl##_##fld) \
        av_store(I_VAL, _I_##tbl##_FLD_##fld, newSVpvf("%s", tbl##_##fld));

static OCIEnv* envhp = (OCIEnv*)0; // environment handle
```

```
static OCIError* errhp = (OCIError*)0; // error handle
static OCISvcCtx* svchp_OSCADEV2 = (OCISvcCtx*)0; // service context handle
static OCIServer* srvhp_OSCADEV2 = (OCIServer*)0; // server handle
static OCISession* authp_OSCADEV2 = (OCISession*)0; // user session (authentication) handle

static OCIStmt* stmthp_SAMPLE2_PRODUCT_CODE = (OCIStmt*)0;
static OCIDefine* define_SAMPLE2_PRODUCT_CODE_KEY = (OCIDefine*)0;
static sb2 indicator_SAMPLE2_PRODUCT_CODE_KEY;
static text SAMPLE2_PRODUCT_CODE_KEY[STD_FLD_LEN];
static text SAMPLE2_PRODUCT_CODE_DESCRIPTION[STD_FLD_LEN];
static OCIDefine* define_SAMPLE2_PRODUCT_CODE_DESCRIPTION = (OCIDefine*)0;
static sb2 indicator_SAMPLE2_PRODUCT_CODE_DESCRIPTION;
static text SAMPLE2_PRODUCT_CODE_LOCATION[STD_FLD_LEN];
static OCIDefine* define_SAMPLE2_PRODUCT_CODE_LOCATION = (OCIDefine*)0;
static sb2 indicator_SAMPLE2_PRODUCT_CODE_LOCATION;

static OCIStmt* stmthp_SAMPLE1_PRODUCT_CODE = (OCIStmt*)0;
static OCIBind* bndhp_SAMPLE1_PRODUCT_CODE = (OCIBind*)0;
static OCIDefine* define_SAMPLE1_PRODUCT_CODE_KEY = (OCIDefine*)0;
static sb2 indicator_SAMPLE1_PRODUCT_CODE_KEY;
static text SAMPLE1_PRODUCT_CODE_KEY[STD_FLD_LEN];
static text SAMPLE1_PRODUCT_CODE_DESCRIPTION[STD_FLD_LEN];
static OCIDefine* define_SAMPLE1_PRODUCT_CODE_DESCRIPTION = (OCIDefine*)0;
static sb2 indicator_SAMPLE1_PRODUCT_CODE_DESCRIPTION;
static text SAMPLE1_PRODUCT_CODE_LOCATION[STD_FLD_LEN];
static OCIDefine* define_SAMPLE1_PRODUCT_CODE_LOCATION = (OCIDefine*)0;
static sb2 indicator_SAMPLE1_PRODUCT_CODE_LOCATION;

static const char *fields[GFCACHERECS][GFMAXPIPFLDS];
static FILE *fstream = (FILE*)0;

int OracleConnect (int fd)
{
    oracle_open_all();
    oracle_prep_all();
    fstream = fdopen(fd, "r");
    return 1;
}

void oracle_checkerr(OCIError *errhp, sword status, text *msg)
{
    text errbuf[512];
    sb4 errcode = 0;

    switch (status)
    {
        case OCI_SUCCESS:
            break;
        case OCI_SUCCESS_WITH_INFO:
            (void) fprintf(stderr, "** Oracle Error - OCI_SUCCESS_WITH_INFO\n");
            break;
        case OCI_NEED_DATA:
            (void) fprintf(stderr, "** Oracle Error - OCI_NEED_DATA\n");
            break;
        case OCI_NO_DATA:
            (void) fprintf(stderr, "** Oracle Error - OCI_NODATA\n");
            break;
        case OCI_ERROR:
            (void) OCIErrorGet((dvoid *)errhp, (ub4) 1, (text *) NULL,&errcode,errbuf, (ub4) sizeof(errbuf), O
CI_HTYPE_ERROR);
            (void) fprintf(stderr, "** Oracle Error - %s - %.*s\n", msg, 512, errbuf);
            break;
        case OCI_INVALID_HANDLE:
            (void) fprintf(stderr, "** Oracle Error - OCI_INVALID_HANDLE\n");
            break;
        case OCI_STILL_EXECUTING:
            (void) fprintf(stderr, "** Oracle Error - OCI_STILL_EXECUTE\n");
            break;
        case OCI_CONTINUE:
            (void) fprintf(stderr, "** Oracle Error - OCI_CONTINUE\n");
            break;
        default:
            break;
    }

}

int oracle_open_all ()
{

    ub4 init_mode = OCI_DEFAULT; // or OCI_OBJECT
    ub4 credt = OCI_CRED_RDBMS;
    sword status;

    // initialize OCI and set up handles
```

```
    if
    (
        OCIInitialize
        (
            OCI_THREADED, (dvoid*)0,
            (dvoid* (*)(dvoid*, size_t))0,
            (dvoid* (*)(dvoid*, dvoid*, size_t))0,
            (void (*)(dvoid*, dvoid*))0
        ) != OCI_SUCCESS
    )
    {
        fprintf(stderr, "ERROR: failed to initialize OCI\n");
        exit(1);
    }
    //  fprintf(stderr, "initialized OCI\n");

    if ((status = OCIEnvInit(&envhp, OCI_DEFAULT, (size_t)0, (dvoid**)0)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"Unable to initialize environment handle");
        exit(1);
    }
    //  fprintf(stderr, "initialized environment handle\n");

    if
    (
        (status = OCIHandleAlloc(envhp,
        (dvoid**)&errhp, OCI_HTYPE_ERROR,
        (size_t)0, (dvoid**)0)) != OCI_SUCCESS
    )
    {
        oracle_checkerr(errhp, status, (text*)"Unable to allocate error handle");
    }
    //  fprintf(stderr, "allocated error handle\n");

//-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//++++++ Db OSCADEV2 --> Type :Pequel::Type::Db::Oracle::Element ++++++
//-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    text* username_OSCADEV2 = (text*)"gprsdev";
    text* password_OSCADEV2 = (text*)"gprsdev";
    text* server_OSCADEV2 = (text*)"OSCADEV2";

    if
    (
        (status = OCIHandleAlloc(envhp,
        (dvoid**)&authp_OSCADEV2, OCI_HTYPE_SESSION,
        (size_t)0, (dvoid**)0)) != OCI_SUCCESS
    )
    {
        oracle_checkerr(errhp, status, (text*)"Unable to allocate authenication handle");
    }
    //    fprintf(stderr, "allocated authenication handle\n");

    // attach server
    if
    (
        (status = OCIHandleAlloc(envhp,
        (dvoid**)&srvhp_OSCADEV2, OCI_HTYPE_SERVER,
        (size_t)0, (dvoid**)0)) != OCI_SUCCESS
    )
    {
        oracle_checkerr(errhp, status, (text*)"Unable to allocate server handle");
    }
    //    fprintf(stderr, "allocated server handle\n");

    if
    (
        (status = OCIServerAttach(srvhp_OSCADEV2,
        errhp,
        server_OSCADEV2,
        (sb4)strlen((char*)server_OSCADEV2), OCI_DEFAULT)) != OCI_SUCCESS
    )
    {
        oracle_checkerr(errhp, status, (text*)"Unable to attach server");
    //  fprintf(stderr, "Error (%d): Unable to attach server '%s'\n", status, (const char*)server_OSCADEV2);
    }
    //  fprintf(stderr, "attached server '%s'\n", (const char*)server_OSCADEV2);

    if
    (
        (status = OCIHandleAlloc(envhp,
        (dvoid**)&svchp_OSCADEV2, OCI_HTYPE_SVCCTX,
        (size_t)0, (dvoid**)0)) != OCI_SUCCESS
    )
    {
```

```
            oracle_checkerr(errhp, status, (text*)"Unable to allocate service handle");
        }
        //  fprintf(stderr, "allocated service handle\n");

        if
        (
            (status = OCIAttrSet(svchp_OSCADEV2, OCI_HTYPE_SVCCTX,
            (dvoid*)srvhp_OSCADEV2, (ub4)0, OCI_ATTR_SERVER,
            errhp)) != OCI_SUCCESS
        )
        {
            oracle_checkerr(errhp, status, (text*)"Unable to set server handle in service handle");
        }
        //  fprintf(stderr, "set server handle in service handle\n");

        // log on
        if
        (
            (status = OCIAttrSet(authp_OSCADEV2, OCI_HTYPE_SESSION,
            (dvoid*)username_OSCADEV2,
            (ub4)strlen((char*)username_OSCADEV2), OCI_ATTR_USERNAME,
            errhp)) != OCI_SUCCESS
        )
        {
            oracle_checkerr(errhp, status, (text*)"Unable to set username as attributes on authentication handle")
;
        }
        if
        (
            (status = OCIAttrSet(authp_OSCADEV2, OCI_HTYPE_SESSION,
            (dvoid*)password_OSCADEV2,
            (ub4)strlen((char*)password_OSCADEV2), OCI_ATTR_PASSWORD,
            errhp)) != OCI_SUCCESS
        )
        {
            oracle_checkerr(errhp, status, (text*)"Unable to set password as attributes on authentication handle")
;
        }
        //  fprintf(stderr, "set username/password as attributes on authentication handle\n");

        if
        (
            (status = OCISessionBegin(svchp_OSCADEV2,
            errhp,
            authp_OSCADEV2,
            credt, OCI_DEFAULT)) != OCI_SUCCESS
        )
        {
            oracle_checkerr(errhp, status, (text*)"Unable to log on");
        //  fprintf(stderr, "Error (%d): Unable to log on as '%s'\n", status, (const char*)username_OSCADEV2);
        }
        //  fprintf(stderr, "logged on as '%s'\n", (const char*)username_OSCADEV2);

        if
        (
            (status = OCIAttrSet(svchp_OSCADEV2, OCI_HTYPE_SVCCTX,
            (dvoid*)authp_OSCADEV2, (ub4)0, OCI_ATTR_SESSION,
            errhp)) != OCI_SUCCESS
        )
        {
            oracle_checkerr(errhp, status, (text*)"Unable to set authentication handle in service handle");
        }
        //  fprintf(stderr, "set authentication handle in service handle\n");

        return 1;
    }

    int oracle_prep_all ()
    {
        ub4 prefetch = 100;
        sword status;

    //++++++ Table SAMPLE2 --> Type :Pequel::Type::Table::Oracle::Merge ++++++
        if ((status = OCIHandleAlloc(envhp, (dvoid **)&stmthp_SAMPLE2_PRODUCT_CODE, OCI_HTYPE_STMT, 0, 0)) != OCI_
    SUCCESS)
        {
            oracle_checkerr(errhp, status, (text*)"allocate statement handle SAMPLE2_PRODUCT_CODE");
        }
        text *sql_SAMPLE2_PRODUCT_CODE = (text*)"select key, description, location from SAMPLE2 FULL";
        if ((status = OCIStmtPrepare(stmthp_SAMPLE2_PRODUCT_CODE, errhp, sql_SAMPLE2_PRODUCT_CODE,
            strlen((const char*)sql_SAMPLE2_PRODUCT_CODE), OCI_NTV_SYNTAX, 0)) != OCI_SUCCESS)
        {
            oracle_checkerr(errhp, status, (text*)"prepare statement SAMPLE2_PRODUCT_CODE");
        }
        if ((status = OCIAttrSet(stmthp_SAMPLE2_PRODUCT_CODE, OCI_HTYPE_STMT, &prefetch, (ub4)0, OCI_ATTR_PREFETCH
```

```
_ROWS, errhp)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"prefetch attribute SAMPLE2_PRODUCT_CODE");
    }
    if ((status = OCIStmtExecute(svchp_OSCADEV2, stmthp_SAMPLE2_PRODUCT_CODE, errhp,
        (ub4)0, (ub4)0, (CONST OCISnapshot*)NULL, (OCISnapshot*)NULL, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"execute statement SAMPLE2_PRODUCT_CODE");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE2_PRODUCT_CODE, &define_SAMPLE2_PRODUCT_CODE_KEY, errhp,
        (ub4)1, SAMPLE2_PRODUCT_CODE_KEY, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE2_PRODUCT_CODE_KEY, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE2_PRODUCT_CODE_KEY");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE2_PRODUCT_CODE, &define_SAMPLE2_PRODUCT_CODE_DESCRIPTION, errhp,

        (ub4)2, SAMPLE2_PRODUCT_CODE_DESCRIPTION, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE2_PRODUCT_CODE_DESCRIPTION, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE2_PRODUCT_CODE_DESCRIPTION");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE2_PRODUCT_CODE, &define_SAMPLE2_PRODUCT_CODE_LOCATION, errhp,
        (ub4)3, SAMPLE2_PRODUCT_CODE_LOCATION, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE2_PRODUCT_CODE_LOCATION, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE2_PRODUCT_CODE_LOCATION");
    }

//++++++ Table SAMPLE1 --> Type :Pequel::Type::Table::Oracle ++++++
    if ((status = OCIHandleAlloc(envhp, (dvoid **)&stmthp_SAMPLE1_PRODUCT_CODE, OCI_HTYPE_STMT, 0, 0)) != OCI_
SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"allocate statement handle SAMPLE1_PRODUCT_CODE");
    }
    text *sql_SAMPLE1_PRODUCT_CODE = (text*)"select key, description, location from SAMPLE1 where key = :key";
    if ((status = OCIStmtPrepare(stmthp_SAMPLE1_PRODUCT_CODE, errhp, sql_SAMPLE1_PRODUCT_CODE,
        strlen((const char*)sql_SAMPLE1_PRODUCT_CODE), OCI_NTV_SYNTAX, 0)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"prepare statement SAMPLE1_PRODUCT_CODE");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE1_PRODUCT_CODE, &define_SAMPLE1_PRODUCT_CODE_KEY, errhp,
        (ub4)1, SAMPLE1_PRODUCT_CODE_KEY, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE1_PRODUCT_CODE_KEY, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE1_PRODUCT_CODE_KEY");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE1_PRODUCT_CODE, &define_SAMPLE1_PRODUCT_CODE_DESCRIPTION, errhp,

        (ub4)2, SAMPLE1_PRODUCT_CODE_DESCRIPTION, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE1_PRODUCT_CODE_DESCRIPTION, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE1_PRODUCT_CODE_DESCRIPTION");
    }
    if ((status = OCIDefineByPos(stmthp_SAMPLE1_PRODUCT_CODE, &define_SAMPLE1_PRODUCT_CODE_LOCATION, errhp,
        (ub4)3, SAMPLE1_PRODUCT_CODE_LOCATION, STD_FLD_LEN+1, (ub2)SQLT_STR,
        (void*)&indicator_SAMPLE1_PRODUCT_CODE_LOCATION, 0, 0, OCI_DEFAULT)) != OCI_SUCCESS)
    {
        oracle_checkerr(errhp, status, (text*)"define SAMPLE1_PRODUCT_CODE_LOCATION");
    }
    return 1;
}

void OracleDisconnect ()
{
    OCIHandleFree((dvoid *)stmthp_SAMPLE2_PRODUCT_CODE, (ub4)OCI_HTYPE_STMT);
    OCIHandleFree((dvoid *)stmthp_SAMPLE1_PRODUCT_CODE, (ub4)OCI_HTYPE_STMT);
    OCISessionEnd(svchp_OSCADEV2, errhp, authp_OSCADEV2, (ub4)OCI_DEFAULT);
    OCIServerDetach(srvhp_OSCADEV2, errhp, (ub4)OCI_DEFAULT );
    OCIHandleFree((dvoid *)authp_OSCADEV2, (ub4)OCI_HTYPE_SESSION);
    OCIHandleFree((dvoid *)srvhp_OSCADEV2, (ub4)OCI_HTYPE_SERVER);
    OCIHandleFree((dvoid *)svchp_OSCADEV2, (ub4)OCI_HTYPE_SVCCTX);
    OCIHandleFree((dvoid *)errhp, (ub4)OCI_HTYPE_ERROR);
    OCIHandleFree((dvoid *)envhp, (ub4)OCI_HTYPE_ENV);
}

int readcache ()
{

    register char *p;
    register int recs;
    register int f;
    static char inp[GFCACHERECS][GFMAXPIPBUFFER];
    static eof=0;
```

```
    if (eof) return 0;
    recs = 0;
    while (recs < GFCACHERECS)
    {
        f=0;
        if (!fgets(inp[recs], GFMAXPIPBUFFER, fstream) ) { eof=1; return recs; }
        inp[recs][strlen(inp[recs])-1] = '\0';
        memset(fields[recs], 0, sizeof(fields[recs]));
        p = inp[recs];
        fields[recs][0] = p;
        while (*p)
        {
            if (*p == '|')
            {
                fields[recs][++f] = p + 1;
                *p = '\0';
            }
            p++;
        }
        recs++;
    }
    return recs;

}
int readsplit (SV* I_VAL_ref)
{
    char sql[4096];
    int ret;
    char *pzErrMsg = 0;
    static int current_cache_maxrecs=GFCACHERECS;
    static int current_cache_rec=GFCACHERECS;
    if (++current_cache_rec >= current_cache_maxrecs)
    {
        if (current_cache_maxrecs < GFCACHERECS) return 0;
        if ((current_cache_maxrecs = readcache()) == 0) return 0;
        current_cache_rec = 0;
    }

    register AV* I_VAL = (AV*)SvRV(I_VAL_ref);
    if (!SvROK(I_VAL_ref)) croak("I_VAL_ref is not a reference");
    av_clear(I_VAL);
    register int f=0;
    for (f=0; f < GFMAXPIPFLDS; f++)
    {
        if (fields[current_cache_rec][f] == 0) av_store(I_VAL, f, newSVpvn("", 0));
        else av_store(I_VAL, f, newSVpvn(fields[current_cache_rec][f], strlen(fields[current_cache_rec][f])));
    }

    int pN;
    ub4 prefetch = 100;
    static int last_step_SAMPLE2_PRODUCT_CODE = OCI_SUCCESS;
//++++++ Table SAMPLE1 (PRODUCT_CODE) --> Type :Pequel::Type::Table::Oracle ++++++
    if ((status = OCIBindByPos(stmthp_SAMPLE1_PRODUCT_CODE, &bndhp_SAMPLE1_PRODUCT_CODE, errhp, (ub4)1,
        (dvoid *)fields[current_cache_rec][_I_PRODUCT_CODE], (sb4)strlen(fields[current_cache_rec][_I_PRODUCT_
CODE])+1, SQLT_STR,
        (dvoid *)0, (ub2 *)0, (ub2 *)0, (ub4)0, (ub4 *)0, (ub4)OCI_DEFAULT)) != OCI_SUCCESS)
    {
        (void) OCIErrorGet(errhp, (ub4)1, (text *)NULL, &errcode, errbuf, (ub4) sizeof(errbuf), OCI_HTYPE_ERRO
R);
        oracle_checkerr(errhp, status, (text*)"bind SAMPLE1_PRODUCT_CODE");
    }
    if (OCIStmtExecute(svchp_OSCADEV2, stmthp_SAMPLE1_PRODUCT_CODE, errhp, (ub4) 1,
        (ub4) 0, (CONST OCISnapshot *) NULL, (OCISnapshot *) NULL, OCI_DEFAULT) == OCI_SUCCESS)
    {
        _av_store_string(SAMPLE1_PRODUCT_CODE, KEY);
        _av_store_string(SAMPLE1_PRODUCT_CODE, DESCRIPTION);
        _av_store_string(SAMPLE1_PRODUCT_CODE, LOCATION);
    }

//++++++ Table SAMPLE2 (PRODUCT_CODE) --> Type :Pequel::Type::Table::Oracle::Merge ++++++
    _OracleMergeFetchString(SAMPLE2_PRODUCT_CODE,_I_PRODUCT_CODE)
    {
        _av_store_string(SAMPLE2_PRODUCT_CODE, KEY);
        _av_store_string(SAMPLE2_PRODUCT_CODE, DESCRIPTION);
        _av_store_string(SAMPLE2_PRODUCT_CODE, LOCATION);
    }

    return 1;
}

int OpenSortStream (int fd)
{
    if ((fstream = fdopen(fd, "r")) == (FILE*)0)
        croak("oracle_table.pql:Unable to open input file stream.");
    return 1;
```

```
}

~; #End of Inline-C Code
```

7. ABOUT PEQUEL

This document was generated by Pequel.

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