Tech info.(vol.3)

Wrapper를 사용한 Online 경보 스크립트

다음은 onconfig에서 ALARMPROGRAM 매개변수를 사용하는 두 가지 예를 나타낸 것입니다.

첫째 예는 둘째 예에 대한 wrapper입니다. 둘째 예는 매개변수를 사용하여 OnLine이 경보 프로그램을 전달합니다.

첫째 스크립트의 제일 처음에 나오는 완전 경로로 ALARMPROGRAM을 설정해야 할 것입니다.

```
ALARMPROGRAM   /full/path/to/almprog.sh
```

almprog.sh

```
#!/bin/sh

/full/path/to/online_alarm.sh "$1" "$2" "$3" "$4" "$5" >>

 /full/path/to/almprog.out 2>&1
```

online_alarm.sh

```
#!/bin/sh

############################################################

# Program:

# online_alarm.sh

#
```

```
# Author:

# David Ranney

#

# To Compile:

# chmod 555 online_alarm.sh

#

# To Run:

# online_alarm.sh severity class-id class-msg specific-msg see-also

#       severity: Category of event

#       class-id: Class identifier

#       class-msg: string containing text of message

#       specific-msg: string containing specific information

#       see-also: path to a see-also file

#
```

```
# Description:

# This script was written to conform to the calling convention

# required for an online event alarm file according to the July 8

# 1994 document.

#

# This script sends email and pages the systems group when necessary.

#

#####################################################################

if

 (`test $# -ne 5`) then

    echo "online_alarm.sh requires 5 arguments"

    exit 1

fi

# Get the arguments themselves

severity=$1
```

```
class_id=$2

class_msg=$3

specific_msg=$4

see_also=$5

datevar=`date`

if (`test $severity -ge 3`) then

   # Severity ATTENTION or better - send email

     echo "====="

     echo "Reasonably severe OnLine event: "

     echo "Severity: $severity"

     echo "Class Id: $class_id"

     echo "Class msg: $class_msg"

     echo "Specific msg: $specific_msg"

     echo "See Also: $see_also"
```

```
echo $datevar



# An event of 3 or greater importance has occurred; send email



case $severity in

    3) sev_msg="ATTENTION LEVEL ONLINE EVENT";

        ;;

    4) sev_msg="EMERGENCY LEVEL ONLINE EVENT";

        ;;

    5) sev_msg="FATAL LEVEL ONLINE EVENT";

        ;;

    *) sev_msg="UNKNOW LEVEL ONLINE EVENT - $severity - ";

        ;;

esac
```

```
case $class_id in

    1)

        class_text="Table failure: $class_msg"

        ;;

    2)

        class_text="Index failure: $class_msg"

        ;;

    3)

        class_text="Blob failure: $class_msg"

        ;;

    4)

        class_text="Chunk is off-line, mirror is active: $class_msg"

        ;;

    5)
```

```
        class_text="DBSpace is off-line: $class_msg"
        ;;
    6)
        class_text="Internal Subsystem Failure: $class_msg"
        ;;
    7)
        class_text="OnLine Initialization failure"
        ;;
    8)
        class_text="Physical Restore failure"
        ;;
    9)
        class_text="Physical Recovery failure"
        ;;
    10)
```

```
        class_text="Logical Recovery failure"
        ;;
    11)
        class_text="Cannot Open Chunk: $class_msg"
        ;;
    12)
        class_text="Cannot Open DBSpace: $class_msg"
        ;;
    13)
        class_text="Performance Improvement possible"
        ;;
    14)
        class_text="Database failure: $class_msg"
        ;;
    15)
        class_text="DR failure"
        ;;
    16)
        class_text="Archive completed: $class_msg"
        ;;
```

```
        17)
            class_text="Archive aborted: $class_msg"
            ;;
        18)
            class_text="Log Backup Completed: $class_msg"
            ;;
        19)
            class_text="Log Backup Aborted: $class_msg"
            ;;
        20)
            class_text="Logical Logs are FULL"
            # Handle special case for full logs
            /usr/ucb/mail -s "$sev_msg" 3871683@skymail.com <<!
$specific_msg -- $datevar
!
            ;;
        21)
            class_text="OnLine resource overflow: $class_msg"
            ;;
        22)
            class_text="Long Transaction Detected"
            ;;
    esac

   # Send email to those who may be interested

    /usr/ucb/mail -s "$sev_msg" mp-sm mp-rt <<!
```

Oracle DECODE 함수를 시뮬레이션하기 위한 내장 프로시저

Oracle의 DECODE 함수는 다음과 같이 사용됩니다.

```
SELECT DECODE (deptNo, 10, "Accounting",
                       20, "Research",
                       30, "Sales",
                       40, "Operation",
                       50, "None")
FROM departments
```

따라서 deptNo가 10이라면 "Accounting"이 리턴되고, 나머지 경우도 마찬가지입니다.

WHERE 절이 동등 표현식과 함께 사용되면 일치하는 행만 DECODE됩니다.

아래는 매개변수의 수(9쌍의 패턴/값까지)를 가변적으로 처리하고 (모든 형식은 varchar(255)로 변환될 수 있다는 가정하에) 서로 다른 형식의 인수를 취할 수 있는 내장 프로시져입니다.

```
create procedure decode(basis varchar (255),
                        case1 varchar (255),
                        retn1 varchar (255),
                        case2 varchar (255) default null,
```

retn2 varchar (255) default null,

case3 varchar (255) default null,

retn3 varchar (255) default null,

case4 varchar (255) default null,

retn4 varchar (255) default null,

case5 varchar (255) default null,

retn5 varchar (255) default null,

case6 varchar (255) default null,

retn6 varchar (255) default null,

case7 varchar (255) default null,

retn7 varchar (255) default null,

case8 varchar (255) default null,

retn8 varchar (255) default null,

case9 varchar (255) default null,

```
                                retn9 varchar (255) default null,

                                deflt varchar (255) default null)

returning varchar (255);




if basis = case1 then

    return retn1;

end if;




if case2 is null then { no default value available. }

    return null;

else

    if retn2 is null then { case2 is default value.}

        return case2;

    else { matching pattern found. }

        if basis = case2 then
```

```
            return retn2;

        end if;

    end if;

end if;



if case3 is null then { no default value available. }

    return null;

else

    if retn3 is null then { case3 is default value.}

        return case3;

    else { matching pattern found. }

        if basis = case3 then

            return retn3;

        end if;
```

```
    end if;

end if;




if case4 is null then { no default value available. }

    return null;

else

    if retn4 is null then { case4 is default value.}

        return case4;

    else { matching pattern found. }

        if basis = case4 then

            return retn4;

        end if;

    end if;

end if;
```

```
if case5 is null then { no default value available. }

    return null;

else

    if retn5 is null then { case5 is default value.}

        return case5;

    else { matching pattern found. }

        if basis = case5 then

            return retn5;

        end if;

    end if;

end if;



if case6 is null then { no default value available. }

    return null;
```

```
else

    if retn6 is null then { case6 is default value.}

        return case6;

    else { matching pattern found. }

        if basis = case6 then

            return retn6;

        end if;

    end if;

end if;




if case7 is null then { no default value available. }

    return null;

else

    if retn7 is null then { case7 is default value.}

        return case7;
```

```
        else { matching pattern found. }

            if basis = case7 then

                return retn7;

            end if;

        end if;

end if;



if case8 is null then { no default value available. }

        return null;

else

        if retn8 is null then { case8 is default value.}

            return case8;

        else { matching pattern found. }

            if basis = case8 then
```

```
            return retn8;

        end if;

    end if;

end if;



if case9 is null then { no default value available. }

    return null;

else

    if retn9 is null then { case9 is default value.}

        return case9;

    else { matching pattern found. }

        if basis = case9 then

            return retn9;

        end if;

    end if;
```

```
end if;

if deflt is null then

    return null;

end if;



return deflt;



end procedure;
```

평일과 달의 마지막 날을 계산하기 위한 4GL 함수

다음은 4GL의 날짜 관련 함수 2가지입니다.

last_day() - 현재 날짜의 이전이나 이후의 달 수를 입력하면 해당되는 달의 마지막 날을 리턴합니다.

week_days() - 주어진 두 날짜 사이의 평일 수를 리턴합니다. 작업 일수를 계산하기 위하여 휴일 테이블을 참조할 수도 있습니다.

```
#-------------------------------------------------------#

  FUNCTION last_day(mths)

# Arguments: Counter +/-/0 of months
```

```
# Purpose:     Determine the last day of the month mths months

#    ahead/back

#           eg: last_day(0) RETURNS last day this month

#                last_day(1) RETURNS last day next month

#                last_day(-12) RETURNS last day this month a year ago

# Returns:     DATE

#-------------------------------------------------------#

DEFINE mths              SMALLINT

DEFINE mm,dd,yy        SMALLINT

DEFINE dte               DATE




# get the month and year of today

LET mm = MONTH(TODAY)

LET yy = YEAR(TODAY)
```

```
# get the month and year of the month mths months from now

WHILE mths != 0

    IF mths < 0 THEN

        # going backward in time

        IF mm = 1 THEN

            # get December last year

            LET mm = 12

            LET yy = yy - 1

        ELSE

            LET mm = mm - 1

        END IF

        LET mths = mths + 1

    ELSE

        # going forward in time
```

```
            IF mm = 12 THEN

                # get January next year

                LET mm = 1

                LET yy = yy + 1

            ELSE

                LET mm = mm + 1

            END IF

            LET mths = mths - 1

        END IF

END WHILE



# now we need to get the month after the one we want

IF mm = 12 THEN

        LET mm = 1
```

```
    LET yy = yy + 1

ELSE

    LET mm = mm + 1

END IF



# set dte to 1st day of the month we set up

LET dte = MDY(mm,1,yy)

# decrment by 1

LET dte = dte - 1



# Viola'! dte is now set to the last day of chosen month



RETURN dte



END FUNCTION
```

last_day(mths)

#---#

FUNCTION week_days(beg_date,end_date)

Arguments: beginning and ending date

Purpose: Determine the number of week days (Mon-Fri)

for the date range (inclusive)

including an optional hook to a holiday table

Returns: SMALLINT, number of week days

#---#

DEFINE beg_date DATE

DEFINE end_date DATE

DEFINE tst_date DATE

DEFINE weekdays SMALLINT

```
## OPTIONAL: the variable below assumes a holiday table with a list

## of holidays that should not be considered weekdays

#DEFINE lholi_date    LIKE holiday.holi_date



LET weekdays = NULL



IF beg_date IS NULL OR end_date IS NULL OR end_date < beg_date THEN

    # this is illegal

     RETURN weekdays

END IF



LET weekdays = 0

LET tst_date = beg_date

WHILE tst_date <= end_date

    # use the built-in WEEKDAY function to determine if this is
```

```
    # Monday (1) through Friday (5)

     IF WEEKDAY(tst_date) > 0 AND WEEKDAY(tst_date) < 6 THEN

          LET weekdays = weekdays + 1

     END IF

     LET tst_date = tst_date + 1 UNITS DAY

END WHILE



## OPTIONAL: if you have a holiday table, work this sort of thing in

#DECLARE c_holiday CURSOR FOR

#      SELECT holi_date FROM holiday

#      WHERE holi_date BETWEEN beg_date AND end_date

#

#FOREACH c_holiday INTO lholi_date

#    # is this date is a weekday, we need to decrement
```

```
#       IF WEEKDAY(lholi_date) > 0 AND WEEKDAY(lholi_date) < 6 THEN

#           LET weekdays = weekdays - 1

#       END IF

#END FOREACH



RETURN weekdays



END FUNCTION

# week_days(beg_date,end_date)
```