

EXERCISE 13

Database Link Access

```
xterm
% get_elem_conn link_access.db 8 default_group

Executed open db.
Executed group Id.= 1
Executed node count.= 27
Executed node Ids.

Executed elem count. = 8
Executed elem Ids.
1,2,3,4,5,6,7,8,
Executed get Topology.
1,28::2,28::3,28::4,28::5,28::6,28::7,28::8,28::
Executed get Associated nodes Connectivity.
Element 1 : Nodes 1,2,5,4,10,11,14,13,
Element 2 : Nodes 2,3,6,5,11,12,15,14,
Element 3 : Nodes 4,5,8,7,13,14,17,16,
Element 4 : Nodes 5,6,9,8,14,15,18,17,
Element 5 : Nodes 10,11,14,13,19,20,23,22,
Element 6 : Nodes 11,12,15,14,20,21,24,23,
Element 7 : Nodes 13,14,17,16,22,23,26,25,
Element 8 : Nodes 14,15,18,17,23,24,27,26,
Executed close db.
%
```

Objectives:

- Link a program to written in the unix environment to the interbase utilities to access a patran database from outside patran.
- Become familiar with the naming conventions of the C functions which access a patran database.

Exercise Description:

Write a program that will open a database, count the nodes, and elements and find the element topology. As a secondary exercise, find the location of each node, and the connectivity of each element. Print this information to the standard out. This is the type of information that you would need to create your own analysis deck.

Exercise Procedure:

1. Change your directory to `ex13` and edit the C function in the file `exercise_13.template`.

Replace the blanks with the appropriate C expressions. Rename the file to `count.c` when you are done.

2. Edit the script `Link.template` so that it will link `count.c` to the PATRAN libraries and call the new executable `LinkCAccess`.
3. Link the function by typing `LinkCAccess` at the prompt.
4. Test the function running PATRAN and playing the session file `test13.ses`.

Your screen should show something similar to this:

```

InterBase/sgi (access method), version "SG-T4.0D"
on disk structure version 8.0
InterBase/sgi (access method), version "SG-T4.0D"
on disk structure version 8.0
Database version 1.8 created by Release 1.4  successfully opened.
Executed open db.
Executed group Id.= 1
Executed node count.= 27
Executed node Ids.
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
Executed get nodes.
1 :0 :0
0.000000,0.000000,0.000000
2 :0 :0
0.500000,0.000000,0.000000
3 :0 :0
1.000000,0.000000,0.000000
4 :0 :0
0.000000,0.500000,0.000000
5 :0 :0
0.500000,0.500000,0.000000
6 :0 :0
1.000000,0.500000,0.000000
7 :0 :0
0.000000,1.000000,0.000000
8 :0 :0
0.500000,1.000000,0.000000
9 :0 :0
1.000000,1.000000,0.000000
10 :0 :0
0.000000,0.000000,0.500000
11 :0 :0
0.500000,0.000000,0.500000
12 :0 :0
1.000000,0.000000,0.500000
13 :0 :0

```

```

0.000000,0.500000,0.500000
14 :0 :0
0.500000,0.500000,0.500000
15 :0 :0
1.000000,0.500000,0.500000
16 :0 :0
0.000000,1.000000,0.500000
17 :0 :0
0.500000,1.000000,0.500000
18 :0 :0
1.000000,1.000000,0.500000
19 :0 :0
0.000000,0.000000,1.000000
20 :0 :0
0.500000,0.000000,1.000000
21 :0 :0
1.000000,0.000000,1.000000
22 :0 :0
0.000000,0.500000,1.000000
23 :0 :0
0.500000,0.500000,1.000000
24 :0 :0
1.000000,0.500000,1.000000
25 :0 :0
0.000000,1.000000,1.000000
26 :0 :0
0.500000,1.000000,1.000000
27 :0 :0
1.000000,1.000000,1.000000
Executed elem count. = 8
Executed elem Ids.
1,2,3,4,5,6,7,8,
Executed get Topology.
1,28::2,28::3,28::4,28::5,28::6,28::7,28::8,28::
Executed get Associated nodes Connectivity.
Element 1 : Nodes 1,2,5,4,10,11,14,13,
Element 2 : Nodes 2,3,6,5,11,12,15,14,
Element 3 : Nodes 4,5,8,7,13,14,17,16,
Element 4 : Nodes 5,6,9,8,14,15,18,17,
Element 5 : Nodes 10,11,14,13,19,20,23,22,
Element 6 : Nodes 11,12,15,14,20,21,24,23,
Element 7 : Nodes 13,14,17,16,22,23,26,25,
Element 8 : Nodes 14,15,18,17,23,24,27,26,
Executed close db.

```

Exercise Template:

```

#include <stdio.h>
#include <math.h>
#include <string.h>
#define MAX 50

/*
 * argv[1] == database name
 * argv[2] == maximum # of nodes/element(default is 8)
 * argv[3] == group name (default is default_group)
 */
main(argc, argv)
    int argc;
    char *argv[];
    {
int i,j, connectivity[4*MAX],old_num_nodes,old_num_elems,
    old_elem_ids[MAX],topo_codes[MAX],
    old_node_ids[MAX];
int group_id,flag,max_connect=8,ref_coords[MAX],
    analy_coords[MAX];
static char group_name[80] = "default_group";

float new_xyz[3 * MAX], old_xyz[3 * MAX],glob_xyzs[3 * MAX];

    if ( argc > 2 ) max_connect = atoi(argv[2]);
    if ( argc > 3 ) strcpy(group_name, argv[3]);

    /* open P3 database to read data */

    flag = DbOpenDatabase(*****1*****);
    if(flag != 0) {
        printf("\n Cannot Open Database.\n");
        exit ( flag );
    }
    printf("Executed open db.\n");

    flag = DbGetGroupId(*****2*****);
    if(flag != 0) {
        printf("\n Group Id failed.\n");
        exit( flag );
    }
    printf("Executed group Id.= %d\n", group_id);

/* Get number of nodes in the group */

    flag = DbFCountNodesInGroup(group_id,&old_num_nodes);
    if(flag != 0) {
        printf("\n Node count failed.\n");
        exit (flag);
    }
    printf("Executed node count.= %d\n", old_num_nodes);

/* Get node IDs */

    flag = DbFGetAllNodeIdsInGroup(*****3*****);
    if(*****4*****) {

```

```

        printf("\n NodeIDs failed.\n");
        exit (flag);
    }
    printf("Executed node Ids.\n");
    printf("%d", old_node_ids[0]);
    for ( i = 1 ; i < old_num_nodes ; i++ )
        { printf(",%d", old_node_ids[i]); }
    printf("\n");
/* Get node information */

    flag = *****5***** (old_num_nodes,old_node_ids,ref_coords,
        analy_coords,old_xyz);
    if(flag != 0) {
        printf("\n Node information failed.\n");
        exit (flag);
    }
    printf("Executed get nodes.\n");
    for ( i = 0 ; i < old_num_nodes ; i++ )
        {
            printf("%d :", old_node_ids[i]);
            printf("%d :", ref_coords[i]);
            printf("%d\n", analy_coords[i]);
            printf("%f,%f,%f\n", old_xyz[(i)*3],
                old_xyz[(i)*3 + 1], old_xyz[(i)*3 + 2]);
        }
/*****

/* Get number of elem in the group */

    flag = *****6***** (group_id,&old_num_elems);
    if(flag != 0) {
        printf("\n Elem count failed.\n");
        exit (flag);
    }
    printf("Executed elem count. = %d\n", old_num_elems);

/* Get elem IDs */

    flag = DbFGGetElemIdsInGroup(old_num_elems,group_id,old_elem_ids);
    if(flag != 0) {
        printf("\n ElemsIDs failed.\n");
        exit (flag);
    }
    printf("Executed elem Ids.\n");
    for ( i = 0 ; i < old_num_elems ; i++ )
        { printf("%d,", old_elem_ids[i]); }
    printf("\n");

/* Get Elem topology information */

    flag = *****7***** (old_num_elems, old_elem_ids, topo_codes);
    if(flag != 0) {
        printf("\n Topology information failed.\n");
        exit (flag);
    }
    printf("Executed get Topology.\n");
    for ( i = 0 ; i < old_num_elems ; i++ )

```

```

        { printf("%d,%d:", old_elem_ids[i], topo_codes[i]); }
printf("\n");

flag = DbFGetNodesForElems(old_num_elems, max_connect, old_elem_ids,
                           connectivity);

if(flag != 0) {
    printf("\n Associated node information failed.\n");
    exit (flag);
}

printf("Executed get Associated nodes Connectivity.\n");
for ( i = 0 ; i < old_num_elems ; i++ )
    {
    printf("Element %d : Nodes ",old_elem_ids[i]);

    for ( j = 0 ; j < max_connect ; j++ )
        printf("%d,",connectivity[i*max_connect + j]);
    printf("\n");
    }

flag = *****8***** (argv[1]);
if(flag != 0) {
    printf("\n Cannot close Database.\n");
    exit (flag);
}

printf("Executed close db.\n");

exit(0);

}

```

LinkCAccess

```
#!/bin/sh
```

```
# This script is delivered to the user in order to demonstrate how the
# delivered libraries should be linked with any user written C
# programs. This script can also be used to test the validity of
# the delivered libraries.
```

```
# This script is run by simply typing in its name, e.g.
# Prompt> $P3_HOME/customization/LinkCAccess
```

```
# If this script is successful, it should produce an executable called
# "CAccessCalls" in the current directory without any error
# messages from the linker. If run, this executable will merely write
# the phrase "Hello world." to standard output.
```

```
# Note that the user may have to explicitly specify the location
# of the FORTRAN system libraries if it differs from what is listed
# below.
```

```
# In general the format of the compilation/link should be as follows -
```

```
# cc -o <name_of_resulting_executable> \
#     <user_source_files> \
#     <user_object_files> \
#     <user_libraries> \
#     "$P3_HOME"/customization/dbaccess_stubs.o \
#     "$P3_HOME"/customization/dbaccess.a \
#     /usr/interbase/lib/gds_b.a \
#     <needed_FORTRAN_system_libraries>
```

```

# Fetch value of $P3_HOME or set to default value of "/patran/patran3"
# if this environment variable is not set

if [ -z "$P3_HOME" ] ; then
    p3_home="/patran/patran3"
else
    p3_home="$P3_HOME"
fi

# Identify the necessary system libraries

SYSDEFS=$p3_home/customization/SYS_DEFS
if [ -f $SYSDEFS ] ; then
    . $SYSDEFS
else
    echo "$0: can't locate $SYSDEFS!"
    exit 1
fi

# Link the dummy C program

cc -o *****1***** \
    $CCOPTIONS \
    *****2***** \
    "$p3_home"/customization/dbaccess_stubs.o \
    "$p3_home"/customization/dbaccess.a \
    $LIBIB $LIBFORT $LIBMATH $LIBDLOADER

```

```

*1* argv[]
*2* group_name, group_id
*3* old_num_nodes, group_id, old_node_ids
*4* flag = 0
*5* DBRGetNodes
*6* DBRCountItemsInGroup
*7* DBRGetItemEItemFop
*8* DBRCloseDatabase
/* LinkAccess */
*1* ./run_me
*2* count.c

```