과제1. MFCOBOL 컴파일러를 이용한 Dataset 생성 및 JOB Submit

1. 개요

본 과제는 MFCOBOL 컴파일러를 이용한 COBOLAPP 생성 및 JOB Submit을 목적으로 한다. COBOLAPP은 JOB에서 생성한 NVSAM 형태의 PS(SDS) DATASET을 각각 OPEN/WRITE/REWRITE/CLOSE, OPEN/READ/CLOSE 하는 APP 두 본과 PDS MEMBER DATASET으로 동일한 과정을 수행하는 APP 두 본으로 총 네 본을 구현한다. 또한, Tmax 서버에서 수행될 JOB을 생성할 JCL SOURCE도 총 두 본 구현한다.

2. 구성

프로그램 파일	설명
COBAPP1.cob	PS(SDS) OPEN/WRITE/REWRITE/CLOSE COBOL SOURCE
COBAPP2.cob	PS(SDS) OPEN/READ/CLOSE COBOL SOURCE
COBAPP3.cob	PDS MEMBER OPEN/WRITE/REWRITE/CLOSE COBOL SOURCE
COBAPP4.cob	PDS MEMBER OPEN/READ/CLOSE COBOL SOURCE
cobapp.jcl	JCL SOURCE 1
cobapp2.jcl	JCL SOURCE 2

2.1 환경 설정

<rc-oframe-7>

```
# MF Cobol
export COBDIR=/opt/microfocus/cobol
export PATH=${COBDIR}/bin:${PATH}
export LD_LIBRARY_PATH=${COBDIR}/lib:${LD_LIBRARY_PATH}
export COBPATH=${COBDIR}/lib
export COBCPY=${COBDIR}/cpylib
```

2.2 COBOL

<COBAPP1.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. COBAPP1.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO OUTDS
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.

        DATA DIVISION.
```

```
        FILE SECTION.
        FD TMAX.
        01 TMAX-FILE.
            05 TMAX-ID PIC 9(7).
            05 TMAX-NAME PIC X(25).
            05 TMAX-TEAM PIC X(20).

        WORKING-STORAGE SECTION.
        01 WS-TMAX.
            05 WS-TMAX-ID PIC 9(7).
            05 WS-TMAX-NAME PIC X(25).
            05 WS-TMAX-TEAM PIC X(20).

        PROCEDURE DIVISION.
            OPEN OUTPUT TMAX.
                MOVE 0000000 TO TMAX-ID.
                MOVE 'HONG GILDONG' TO TMAX-NAME.
                MOVE 'TP2' TO TMAX-TEAM.
            WRITE TMAX-FILE
            END-WRITE.
            CLOSE TMAX.
            OPEN I-O TMAX.
            READ TMAX
            END-READ.
                MOVE 2017253 TO TMAX-ID.
                MOVE 'KO JAEJUN' TO TMAX-NAME.
                MOVE 'TP2' TO TMAX-TEAM.
            REWRITE TMAX-FILE
            END-REWRITE.
            CLOSE TMAX.
        STOP RUN.
```

<COBAPP2.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. COBAPP2.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
```

```
        SELECT TMAX ASSIGN TO OUTDS
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.

        DATA DIVISION.
          FILE SECTION.
          FD TMAX.
          01 TMAX-FILE.
              05 TMAX-ID PIC 9(7).
              05 TMAX-NAME PIC X(25).
              05 TMAX-TEAM PIC X(20).
          WORKING-STORAGE SECTION.
          01 WS-TMAX.
              05 WS-TMAX-ID PIC 9(7).
              05 WS-TMAX-NAME PIC X(25).
              05 WS-TMAX-TEAM PIC X(20).
          01 WS-EOF PIC A(1).
          PROCEDURE DIVISION.
              OPEN INPUT TMAX.
              PERFORM UNTIL WS-EOF='Y'
                 READ TMAX INTO WS-TMAX
                    AT END MOVE 'Y' TO WS-EOF
                    NOT AT END DISPLAY WS-TMAX
                 END-READ
              END-PERFORM.
              CLOSE TMAX.
        STOP RUN.
```

<COBAPP3.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. COBAPP3.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO PDSMEM
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.
```

```
        DATA DIVISION.
            FILE SECTION.
            FD TMAX.
            01 TMAX-FILE.
                05 TMAX-ID PIC 9(7).
                05 TMAX-NAME PIC X(25).
                05 TMAX-TEAM PIC X(20).

            WORKING-STORAGE SECTION.
            01 WS-TMAX.
                05 WS-TMAX-ID PIC 9(7).
                05 WS-TMAX-NAME PIC X(25).
                05 WS-TMAX-TEAM PIC X(20).

            PROCEDURE DIVISION.
                OPEN OUTPUT TMAX.
                    MOVE 0000000 TO TMAX-ID.
                    MOVE 'HONG GILDONG' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                WRITE TMAX-FILE
                END-WRITE.
                CLOSE TMAX.
                OPEN I-O TMAX.
                READ TMAX
                END-READ.
                    MOVE 2017253 TO TMAX-ID.
                    MOVE 'KO JAEJUN' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                REWRITE TMAX-FILE
                END-REWRITE.
                CLOSE TMAX.
            STOP RUN.
```

<COBAPP4.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. COBAPP4.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
```

```
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO PDSMEM
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.

        DATA DIVISION.
          FILE SECTION.
          FD TMAX.
          01 TMAX-FILE.
              05 TMAX-ID PIC 9(7).
              05 TMAX-NAME PIC X(25).
              05 TMAX-TEAM PIC X(20).
          WORKING-STORAGE SECTION.
          01 WS-TMAX.
              05 WS-TMAX-ID PIC 9(7).
              05 WS-TMAX-NAME PIC X(25).
              05 WS-TMAX-TEAM PIC X(20).
          01 WS-EOF PIC A(1).
          PROCEDURE DIVISION.
              OPEN INPUT TMAX.
              PERFORM UNTIL WS-EOF='Y'
                 READ TMAX INTO WS-TMAX
                    AT END MOVE 'Y' TO WS-EOF
                    NOT AT END DISPLAY WS-TMAX
                 END-READ
              END-PERFORM.
              CLOSE TMAX.
        STOP RUN.
```

<$OPENFRAME_HOME/volume_DEFVOL/SYS1.JCLLIB/cobapp.jcl>

```
//KOJOB   JOB   USER=ROOT,PASSWORD=SYS1
//JOBLIB DD     DSN=SYS1.COBLIB,DISP=SHR
//STEP1   EXEC PGM=COBAPP1
//OUTDS   DD     DSN=OUTDS,DISP=(NEW,CATLG,DELETE)
//SYSOUT DD     SYSOUT=*
//STEP2   EXEC PGM=COBAPP2
//OUTDS   DD     DSN=OUTDS,DISP=SHR
//SYSOUT DD     SYSOUT=*
```

<cobapp2.jcl>

```
//KOJOB   JOB   USER=ROOT,PASSWORD=SYS1
//JOBLIB DD     DSN=SYS1.COBLIB,DISP=SHR
//STEP1   EXEC PGM=COBAPP3
//PDSMEM DD     DSN=SYS1.PDSLIB(MEMBER),DISP=(NEW,CATLG,DELETE)
//SYSOUT DD     SYSOUT=*
//STEP2   EXEC PGM=COBAPP4
//PDSMEM DD     DSN=SYS1.PDSLIB(MEMBER),DISP=SHR
//SYSOUT DD     SYSOUT=*
```

3. 수행 과정

3.1 libcobsw.so 링크 변경

```
cd $OPENFRAME_HOME/lib
ln -sf libswmfcob.so.* libcobsw.so
```

```
ko@ko-tmax:~/oframe_7/lib$ ll libcobsw.so
lrwxrwxrwx 1 ko ko 24  7월  24 14:47 libcobsw.so -> libswmfcob.so.64.7_0_3_0*
```

3.2 COBOL APP 컴파일

```
cob -zvP COBAPP1.cob -C CALLFH\"TEXTFH\" -C ASSIGN\"EXTERNAL\" -L$OPENFRAME_HOME/lib -ltextfh3
cob -zvP COBAPP2.cob -C CALLFH\"TEXTFH\" -C ASSIGN\"EXTERNAL\" -L$OPENFRAME_HOME/lib -ltextfh3
cob -zvP COBAPP3.cob -C CALLFH\"TEXTFH\" -C ASSIGN\"EXTERNAL\" -L$OPENFRAME_HOME/lib -ltextfh3
cob -zvP COBAPP4.cob -C CALLFH\"TEXTFH\" -C ASSIGN\"EXTERNAL\" -L$OPENFRAME_HOME/lib -ltextfh3
```

```
cp COBAPP*.so $OPENFRAME_HOME/volume_DEFVOL/SYS1.COBLIB
```

3.3 JOB SUBMIT

- PS(SDS)

```
tjesmgr
```

```
|>
|Command : [run cobapp.jcl]
|Node name :  A N Y
|(JOB00004) /home/ko/oframe_7/volume_DEFVOL/SYS1.JCLLIB/cobapp.jcl is submitted as KOJOB(JOB00004).
```

- PDS MEMBER

```
tjesmgr
```

```
|>
|Command : [run cobapp2.jcl]
|Node name :  A N Y
|(JOB00008) /home/ko/oframe_7/volume_DEFVOL/SYS1.JCLLIB/cobapp2.jcl is submitted as KOJOB(JOB00008).
+-----------------------------------------------------------------------------------------------
```

4. 실행 결과

4.1 JOB SUBMIT

- PS(SDS) 생성 및 READ

tjesmgr

```
>
Command : [ps]
 JOBNAME  JOBID    CLASS   STATUS   RC     NODE     START-TIME        END-TIME          JCL
-------------------------------------------------------------------------------------------------
 KOJOB    JOB00004   A     Done     R00000 NODE1    20170724/14:48:34 20170724/14:48:35 cobapp.jcl
-------------------------------------------------------------------------------------------------
```

```
2017253KO JAEJUN                TP2
```

- PDS MEMBER 생성 및 READ

tjesmgr

```
 KOJOB    JOB00008   A     Done     R00000 NODE1    20170724/16:25:21 20170724/16:25:22 cobapp2.jcl
```

```
2017253KO JAEJUN                TP2
```

4.2 PS(SDS) DATASET

dsview OUTDS

```
00000001: 2017253KO JAEJUN                TP2
----END----
OUTDS                  Line 1 of 1, Column 52
Hex: 3230313732353334B4F204A41454A554E20202020202020202020202020202020545032202020202020202020202020202020202020
```

4.3 PDS MEMBER

dsview SYS1.PDSLIB MEMBER

```
00000001: 2017253KO JAEJUN                TP2
----END----
SYS1.PDSLIB(MEMBER)     MEMBER         Line 1 of 1, Column 52
Hex: 3230313732353334B4F204A41454A554E20202020202020202020202020202020545032202020202020202020202020202020202020
```

5. 이슈 사항

- External File Handler를 지원하는 MFCOBOL 컴파일러의 경우 External File Handler를 지정하기 위해서는 컴파일할 때 -C′ CALLFH" TEXTFH"′ 옵션과 –C′ ASSIGN" EXTERNAL" 옵션을 추가해 External File Handler가 사용할 함수를 지정하고 libtextfh.so를 연결(link)해야 한다.

과제2. OFCOBOL 컴파일러를 이용한 Dataset 생성 및 JOB Submit

1. 개요

본 과제는 OFCOBOL 컴파일러를 이용한 COBOLAPP 생성 및 JOB Submit을 목적으로 한다. COBOLAPP은 JOB에서 생성한 NVSAM 형태의 PS(SDS) DATASET을 각각 OPEN/WRITE/REWRITE/CLOSE, OPEN/READ/CLOSE 하는 APP 두 본과 PDS MEMBER DATASET으로 동일한 과정을 수행하는 APP 두 본으로 총 네 본을 구현한다. 또한, Tmax 서버에서 수행될 JOB을 생성할 JCL SOURCE도 총 두 본 구현한다.

2. 구성

프로그램 파일	설명
OFCOBAPP1.cob	PS(SDS) OPEN/WRITE/REWRITE/CLOSE COBOL SOURCE
OFCOBAPP2.cob	PS(SDS) OPEN/READ/CLOSE COBOL SOURCE
OFCOBAPP3.cob	PDS MEMBER OPEN/WRITE/REWRITE/CLOSE COBOL SOURCE
OFCOBAPP4.cob	PDS MEMBER OPEN/READ/CLOSE COBOL SOURCE
ofcobapp1.jcl	JCL SOURCE 1
ofcobapp2.jcl	JCL SOURCE 2

2.1 환경 설정

<rc-oframe-7>

```
# OF Cobol
export OFCOB_HOME=${OPENFRAME_HOME}/ofcobol
export PATH=${OFCOB_HOME}/bin:${OFCOB_HOME}/cobolparser/bin:${PATH}
export
LD_LIBRARY_PATH=${OFCOB_HOME}/lib:${OFCOB_HOME}/cobolparser/lib:${LD_LIBRARY_PATH}
export LLVM_HOME=${OFCOB_HOME}/llvm
export LD_LIBRARY_PATH=${LLVM_HOME}/lib:${LD_LIBRARY_PATH}
```

2.2 COBOL

<OFCOBAPP1.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. OFCOBAPP1.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO OFDS
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.
```

```
        DATA DIVISION.
            FILE SECTION.
            FD TMAX.
            01 TMAX-FILE.
                05 TMAX-ID PIC 9(7).
                05 TMAX-NAME PIC X(25).
                05 TMAX-TEAM PIC X(20).

            WORKING-STORAGE SECTION.
            01 WS-TMAX.
                05 WS-TMAX-ID PIC 9(7).
                05 WS-TMAX-NAME PIC X(25).
                05 WS-TMAX-TEAM PIC X(20).

            PROCEDURE DIVISION.
                OPEN OUTPUT TMAX.
                    MOVE 0000000 TO TMAX-ID.
                    MOVE 'HONG GILDONG' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                WRITE TMAX-FILE
                END-WRITE.
                CLOSE TMAX.
                OPEN I-O TMAX.
                READ TMAX
                END-READ.
                    MOVE 2017253 TO TMAX-ID.
                    MOVE 'KO JAEJUN' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                REWRITE TMAX-FILE
                END-REWRITE.
                CLOSE TMAX.
            STOP RUN.
```

<OFCOBAPP2.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. OFCOBAPP2.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
```

```
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO OFDS
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.

        DATA DIVISION.
            FILE SECTION.
            FD TMAX.
            01 TMAX-FILE.
                05 TMAX-ID PIC 9(7).
                05 TMAX-NAME PIC X(25).
                05 TMAX-TEAM PIC X(20).
            WORKING-STORAGE SECTION.
            01 WS-TMAX.
                05 WS-TMAX-ID PIC 9(7).
                05 WS-TMAX-NAME PIC X(25).
                05 WS-TMAX-TEAM PIC X(20).
            01 WS-EOF PIC A(1).
            PROCEDURE DIVISION.
                OPEN INPUT TMAX.
                PERFORM UNTIL WS-EOF='Y'
                    READ TMAX INTO WS-TMAX
                        AT END MOVE 'Y' TO WS-EOF
                        NOT AT END DISPLAY WS-TMAX
                    END-READ
                END-PERFORM.
                CLOSE TMAX.
        STOP RUN.
```

<OFCOBAPP3.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. OFCOBAPP3.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO OFPDS
            ORGANIZATION IS SEQUENTIAL
            ACCESS IS SEQUENTIAL.
```

```
        DATA DIVISION.
            FILE SECTION.
            FD TMAX.
            01 TMAX-FILE.
                05 TMAX-ID PIC 9(7).
                05 TMAX-NAME PIC X(25).
                05 TMAX-TEAM PIC X(20).

            WORKING-STORAGE SECTION.
            01 WS-TMAX.
                05 WS-TMAX-ID PIC 9(7).
                05 WS-TMAX-NAME PIC X(25).
                05 WS-TMAX-TEAM PIC X(20).

            PROCEDURE DIVISION.
                OPEN OUTPUT TMAX.
                    MOVE 0000000 TO TMAX-ID.
                    MOVE 'HONG GILDONG' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                WRITE TMAX-FILE
                END-WRITE.
                CLOSE TMAX.
                OPEN I-O TMAX.
                READ TMAX
                END-READ.
                    MOVE 2017253 TO TMAX-ID.
                    MOVE 'KO JAEJUN' TO TMAX-NAME.
                    MOVE 'TP2' TO TMAX-TEAM.
                REWRITE TMAX-FILE
                END-REWRITE.
                CLOSE TMAX.
            STOP RUN.
```

<OFCOBAPP4.cob>

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. OFCOBAPP4.

        ENVIRONMENT DIVISION.
        INPUT-OUTPUT SECTION.
```

```
        FILE-CONTROL.
        SELECT TMAX ASSIGN TO OFPDS
              ORGANIZATION IS SEQUENTIAL
              ACCESS IS SEQUENTIAL.

        DATA DIVISION.
            FILE SECTION.
            FD TMAX.
            01 TMAX-FILE.
                 05 TMAX-ID PIC 9(7).
                 05 TMAX-NAME PIC X(25).
                 05 TMAX-TEAM PIC X(20).
            WORKING-STORAGE SECTION.
            01 WS-TMAX.
                 05 WS-TMAX-ID PIC 9(7).
                 05 WS-TMAX-NAME PIC X(25).
                 05 WS-TMAX-TEAM PIC X(20).
            01 WS-EOF PIC A(1).
            PROCEDURE DIVISION.
                 OPEN INPUT TMAX.
                 PERFORM UNTIL WS-EOF='Y'
                    READ TMAX INTO WS-TMAX
                       AT END MOVE 'Y' TO WS-EOF
                       NOT AT END DISPLAY WS-TMAX
                    END-READ
                 END-PERFORM.
                 CLOSE TMAX.
        STOP RUN.
```

2.3 JCL

<$OPENFRAME_HOME/volume_DEFVOL/SYS1.JCLLIB/ofcobapp1.jcl>

```
//OFJOB   JOB   USER=ROOT,PASSWORD=SYS1
//JOBLIB DD    DSN=SYS1.COBLIB,DISP=SHR
//STEP1   EXEC PGM=OFCOBAPP1
//OFDS    DD    DSN=OFDS,DISP=(NEW,CATLG,DELETE)
//SYSOUT DD    SYSOUT=*
//STEP2   EXEC PGM=OFCOBAPP2
//OFDS    DD    DSN=OFDS,DISP=SHR
//SYSOUT DD    SYSOUT=*
```

<ofcobapp2.jcl>

```
//OFJOB   JOB   USER=ROOT,PASSWORD=SYS1
//JOBLIB DD     DSN=SYS1.COBLIB,DISP=SHR
//STEP1   EXEC PGM=OFCOBAPP3
//OFPDS   DD     DSN=SYS1.OFPDS(OFMEM),DISP=(NEW,CATLG,DELETE)
//SYSOUT DD     SYSOUT=*
//STEP2   EXEC PGM=OFCOBAPP4
//OFPDS   DD     DSN=SYS1.OFPDS(OFMEM),DISP=SHR
//SYSOUT DD     SYSOUT=*
```

3. 수행 과정

3.1 libcobsw.so 링크 변경

```
cd $OPENFRAME_HOME/lib
ln -sf libswofcob.so.* libcobsw.so
```

```
ko@ko-tmax:~/oframe_7/lib$ ll libcobsw.so
lrwxrwxrwx 1 ko ko 24  7월  24 16:36 libcobsw.so -> libswofcob.so.64.7_0_3_0*
```

3.2 COBOL APP 컴파일

```
ko@ko-tmax:~/oframe_7/lib$ ofcob -o OFCOBAPP1.so OFCOBAPP1.cob -L$OPENFRAME_HOME/lib -ltextfh3
ko@ko-tmax:~/oframe_7/lib$ ofcob -o OFCOBAPP2.so OFCOBAPP2.cob -L$OPENFRAME_HOME/lib -ltextfh3
ko@ko-tmax:~/oframe_7/lib$ ofcob -o OFCOBAPP3.so OFCOBAPP3.cob -L$OPENFRAME_HOME/lib -ltextfh3
ko@ko-tmax:~/oframe_7/lib$ ofcob -o OFCOBAPP4.so OFCOBAPP4.cob -L$OPENFRAME_HOME/lib -ltextfh3
```

```
cp OFCOBAPP*.so $OPENFRAME_HOME/volume_DEFVOL/SYS1.COBLIB
```

3.3 JOB SUBMIT

- PS(SDS)

```
tjesmgr
```

```
|>
|Command : [run ofcobapp1.jcl]
|Node name :  A N Y
|(JOB00009) /home/ko/oframe_7/volume_DEFVOL/SYS1.JCLLIB/ofcobapp1.jcl is submitted as OFJOB(JOB00009).
+-----------------------------------------------------------------------------------------------
```

- PDS MEMBER

```
tjesmgr
```

```
|>
|Command : [run ofcobapp2.jcl]
|Node name :  A N Y
|(JOB00010) /home/ko/oframe_7/volume_DEFVOL/SYS1.JCLLIB/ofcobapp2.jcl is submitted as OFJOB(JOB00010).
|>
```

4. 실행 결과

4.1 JOB SUBMIT

- PS(SDS) 생성 및 READ

tjesmgr

```
|>
|Command : [ps]
| JOBNAME  JOBID    CLASS   STATUS   RC     NODE     START-TIME        END-TIME          JCL
|-------------------------------------------------------------------------------------------------------------
| OFJOB    JOB00009   A     Done     R00000 NODE1    20170724/16:40:41 20170724/16:40:42 ofcobapp1.jcl
+-------------------------------------------------------------------------------------------------------------
 2017253KO JAEJUN                TP2
```

- PDS MEMBER 생성 및 READ

tjesmgr

```
| OFJOB    JOB00010   A     Done     R00000 NODE1    20170724/16:42:36 20170724/16:42:37 ofcobapp2.jcl
 2017253KO JAEJUN                TP2
```

4.2 PS(SDS) DATASET

dsview OUTDS

```
00000001: 2017253KO JAEJUN                TP2
----END----
OFDS                  Line 1 of 1, Column 52
Hex: 323031373235334B4F204A41454A554E20202020202020202020202020202020545032202020202020202020202020202020202020
```

4.3 PDS MEMBER

dsview SYS1.OFPDS OFMEM

```
00000001: 2017253KO JAEJUN                TP2
----END----
SYS1.OFPDS(OFMEM)       OFMEM          Line 1 of 1, Column 52
Hex: 323031373235334B4F204A41454A554E20202020202020202020202020202020545032202020202020202020202020202020202020
```

과제3. tbESQL/COBOL를 이용한 데이터베이스 INSERT/SELECT COBOL APP 구현

1. 개요

본 과제는 ESQL 문을 포함하는 COBOL APPLICATION을 구현하는 것을 목적으로 한다. COBOL APPLICATION은 준비된 데이터베이스 테이블에 데이터를 INSERT하는 것과 테이블의 전체 데이터를 읽어오는 것, 총 두 본을 구현한다.

2. 구성

프로그램 파일	설명
ESQLAPP1.tbco	tbESQL/COBOL DB INSERT SOURCE
ESQLAPP2.tbco	tbESQL/COBOL DB SELECT SOURCE
esqlapp.jcl	JCL SOURCE

2.1 환경 설정

<rc-oframe-7>

```
# Tibero6
export TB_HOME=${OPENFRAME_HOME}/tibero
export PATH=${TB_HOME}/bin:${TB_HOME}/client/bin:${PATH}
export LD_LIBRARY_PATH=${TB_HOME}/lib:${TB_HOME}/client/lib:${LD_LIBRARY_PATH}
export TB_SID=tb_oframe7

# UnixODBC
export unixODBC_HOME=/usr/lib/x86_64-linux-gnu
export LD_LIBRARY_PATH=${unixODBC_HOME}/lib:${LD_LIBRARY_PATH}
```

2.2 데이터베이스 스키마

```
CREATE TABLE TMAX_INFO (
        USERID NUMBER(7) PRIMARY KEY,
        NAME    VARCHAR(20),
        TEAM    VARCHAR(20)
);
```

2.3 COBOL

<ESQLAPP1.tbco>

```
          IDENTIFICATION DIVISION.
          PROGRAM-ID. ESQLAPP1.

          ENVIRONMENT DIVISION.
          DATA DIVISION.
               WORKING-STORAGE SECTION.
```

```
        EXEC SQL BEGIN DECLARE SECTION END-EXEC.
        01 USERPASS PIC X(30) VALUE Z"tibero/tmax".
        01 USERID     PIC S9(7).
        01 USERNAME PIC X(25).
        01 USERTEAM PIC X(20).
        EXEC SQL END DECLARE SECTION END-EXEC.

        EXEC SQL INCLUDE SQLCA END-EXEC.

    PROCEDURE DIVISION.
        EXEC SQL CONNECT :USERPASS END-EXEC.
        DISPLAY 'SUCCESSFULLY CONNECTED!'.

        EXEC SQL
            SELECT (MAX(USERID)+1) INTO :USERID FROM TMAX_INFO
        END-EXEC.

        MOVE Z"KO JAEJUN" TO USERNAME.
        MOVE Z"TP2" TO USERTEAM.

        EXEC SQL
            INSERT INTO TMAX_INFO (USERID, NAME, TEAM)
                VALUES (:USERID, :USERNAME, :USERTEAM)
        END-EXEC.

        EXEC SQL COMMIT WORK RELEASE END-EXEC.
    STOP RUN.
```

<ESQLAPP2.tbco>

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID. ESQLAPP2.

    ENVIRONMENT DIVISION.
    DATA DIVISION.
        WORKING-STORAGE SECTION.

        EXEC SQL BEGIN DECLARE SECTION END-EXEC.
        01 USERPASS PIC X(30) VALUE Z"tibero/tmax".
        01 USERID     PIC S9(7).
```

```
        01 USERNAME PIC X(25).
        01 USERTEAM PIC X(20).
        EXEC SQL END DECLARE SECTION END-EXEC.

        EXEC SQL INCLUDE SQLCA END-EXEC.

    PROCEDURE DIVISION.
        EXEC SQL DECLARE KO-CURSOR CURSOR FOR
            SELECT USERID, NAME, TEAM FROM TMAX_INFO
        END-EXEC.

        EXEC SQL CONNECT :USERPASS END-EXEC.
        DISPLAY 'SUCCESSFULLY CONNECTED!'.

        EXEC SQL OPEN KO-CURSOR END-EXEC.
        EXEC SQL WHENEVER NOT FOUND GOTO FETCH-END END-EXEC.
        FETCH-LOOP.
            EXEC SQL FETCH KO-CURSOR
                        INTO :USERID, :USERNAME, :USERTEAM
            END-EXEC.
            DISPLAY 'USERID: ' USERID '   NAME: ' USERNAME
                        '   TEAM: ' USERTEAM
            GO TO FETCH-LOOP.
        FETCH-END.
        EXEC SQL CLOSE KO-CURSOR END-EXEC.

        EXEC SQL COMMIT WORK RELEASE END-EXEC.
    STOP RUN.
```

2.4 JCL

<esqlapp.jcl>

```
//EJOB     JOB   USER=ROOT,PASSWORD=SYS1
//JOBLIB DD     DSN=SYS1.COBLIB,DISP=SHR
//STEP1    EXEC PGM=ESQLAPP1
//SYSOUT DD     SYSOUT=*
//STEP2    EXEC PGM=ESQLAPP2
//SYSOUT DD     SYSOUT=*
```

3. 수행 과정

3.1 COBOL 프리 컴파일

```
tbpcb ESQLAPP1.tbco
tbpcb ESQLAPP2.tbco
```

3.2 COBOL 컴파일(MFCOBOL 컴파일러 사용)

```
cob -zvP ESQLAPP1.cob -L$OPENFRAME_HOME/lib -ltextfh3 -L$TB_HOME/client/lib -ltbertl -ltbcli
cp ESQLAPP1.so $OPENFRAME_HOME/volume_DEFVOL/SYS1.COBLIB

cob -zvP ESQLAPP2.cob -L$OPENFRAME_HOME/lib -ltextfh3 -L$TB_HOME/client/lib -ltbertl –ltbcli
cp ESQLAPP2.so $OPENFRAME_HOME/volume_DEFVOL/SYS1.COBLIB
```

3.3 JOB SUBMIT

tjesmgr

```
|>
|Command : [run esqlapp.jcl]
|Node name :  A N Y
|(JOB00011) /home/ko/oframe_7/volume_DEFVOL/SYS1.JCLLIB/esqlapp.jcl is submitted as EJOB(JOB00011).
+-----------------------------------------------------------------------------------------------
```

4. 실행 결과

4.1 JOB SUBMIT

```
|>
|Command : [ps]
| JOBNAME  JOBID    CLASS   STATUS   RC     NODE     START-TIME       END-TIME         JCL
|-----------------------------------------------------------------------------------------------
| EJOB     JOB00011   A     Done     R00000 NODE1    20170724/17:05:46 20170724/17:05:47 esqlapp.jcl
+-----------------------------------------------------------------------------------------------
SUCCESSFULLY CONNECTED!
USERID: 0000000+  NAME: HONG GILDONG              TEAM: TP2
USERID: 1111111+  NAME: HELLO                    TEAM: HI
USERID: 2017253+  NAME: KO JAEJUN^@               TEAM: TP2^@
USERID: 2017254+  NAME: KO JAEJUN^@               TEAM: TP2^@
USERID: 2017255+  NAME: KO JAEJUN^@               TEAM: TP2^@
USERID: 2017256+  NAME: KO JAEJUN^@               TEAM: TP2^@
USERID: 2017257+  NAME: KO JAEJUN^@               TEAM: TP2^@
```

4.2 데이터베이스

```
SQL> select * from tmax_info;

    USERID NAME                      TEAM
---------- ------------------------- --------------------
         0 HONG GILDONG              TP2
   1111111 HELLO                     HI
   2017253 KO JAEJUN                 TP2
   2017254 KO JAEJUN                 TP2
   2017255 KO JAEJUN                 TP2
   2017256 KO JAEJUN                 TP2
   2017257 KO JAEJUN                 TP2

7 rows selected.
```

5. 이슈 사항

- VARYNG keyword

Tibero tbESQL/COBOL 가이드 문서에는 VARCHAR type 의 COBOL 변수 선언 시 VARYNG keyword 를 붙이게 되어 있으나 실제로 붙일 경우 Database INSERT 가 정상적으로 수행되지 않는다는 문제가 있다.

VARCHAR 타입의 변수 선언

VARCHAR 타입의 변수 선언은 COBOL 프로그래밍 언어에서 PIC X(n) 타입의 배열 변수를 선언하는 것과 동일한데, 마지막에 VARYING 키워드를 삽입한다. PIC X(n)과 마찬가지로 문자열의 최대 크기를 반드시 지정해 주어야 한다.

다음은 VARCHAR 타입의 변수를 선언하는 예이다.

[예 2.4] VARCHAR 타입의 변수 선언

```
01 USERNAME PIC X(16) VARYING.
```

과제4. tbESQL/C와 Tmax 서버를 이용한 계좌 입출금 서비스 구현

1. 개요

본 과제는 tbESQL/C와 Tmax 서버를 이용한 계좌 입출금 서비스를 구현하는 것을 목적으로 한다. 이를 위해 ESQL 문이 삽입된 C 기반의 클라이언트/서버 프로그램을 각각 구현하고, 구현된 서버 프로그램을 Tmax 서버에 올려 클라이언트의 요청을 처리한다. 클라이언트-서버 간 통신은 동기형 통신을 사용하고, 서버 프로그램의 구현은 TCS로 한다. 또한, 서버 프로그램과 연동할 데이터베이스로는 Tibero 6를 사용한다.

2. 구성

프로그램 파일	설명
demo.f	필드 버퍼
sample.m	환경 설정
bankcli.c	클라이언트 프로그램
banksvr.tbc	서버 프로그램

2.1 환경 설정

<rc-oframe-7>

```
# Tibero6
export TB_HOME=${OPENFRAME_HOME}/tibero
export PATH=${TB_HOME}/bin:${TB_HOME}/client/bin:${PATH}
export LD_LIBRARY_PATH=${TB_HOME}/lib:${TB_HOME}/client/lib:${LD_LIBRARY_PATH}
export TB_SID=tb_oframe7

# TMAX
export TMAXDIR=${OPENFRAME_HOME}/tmax
export PATH=${TMAXDIR}/bin:${TMAXDIR}/bin:${PATH}
export TMAX_HOST_ADDR=127.0.0.1
export TMAX_HOST_PORT=9999
export FDLFILE=${TMAXDIR}/sample/fdl/tmax.fdl
export LD_LIBRARY_APTH=${TMAXDIR}/lib:${LD_LIBRARY_PATH}
```

<$TMAXDIR/config/sample.m>

```
*DOMAIN
kotmax
    SHMKEY      = 94000,    MAXUSER     = 256,    MINCLH      = 1,
    MAXCLH      = 1,    BLOCKTIME   = 15,    MAXCPC      = 256,
    MAXSPR      = 512,    MAXSVR      = 128,    MAXSVC      = 512,
    DOMAINID    = 4,    IPCPERM     = 0777,    MAXSACALL   = 1024,
    MAXCACALL   = 1024
```

```
*NODE
DEFAULT:
    HOSTNAME = "ko-tmax",
    DOMAINNAME = "kotmax"
NODE1
    TMAXHOME = "/home/ko/oframe_7/tmax",
    TMAXDIR = "/home/ko/oframe_7/tmax",
    APPDIR   = "/home/ko/oframe_7/tmax/appbin",
    TLOGDIR = "/home/ko/oframe_7/tmax/log/tlog",
    ULOGDIR = "/home/ko/oframe_7/tmax/log/ulog",
    SLOGDIR = "/home/ko/oframe_7/tmax/log/slog",
    CLHOPT = " -o /home/ko/oframe_7/tmax/log/clh.log -e
/home/ko/oframe_7/tmax/log/clh.err",
    TPORTNO = 9999, SHMKEY = 94000, RACPORT = 9450

*SVRGROUP
svg_domain
    NODENAME = "NODE1"

*SERVER
kotmaxsvr       SVGNAME = svg_domain,
                    MIN = 1, MAX = 1,
                    CLOPT="-o $(SVR)$(DATE).out -e $(SVR)$(DATE).err"

*SERVICE
DEPOSIT             SVRNAME = kotmaxsvr
WITHDRAW            SVRNAME = kotmaxsvr
```

<$TB_HOME/client/config/tbpc.cfg>

```
INCLUDE=$TB_HOME/client/include
INCLUDE=/usr/lib/gcc/x86_64-linux-gnu/5.4.0/include
INCLUDE=/usr/include
```

2.2 데이터베이스 스키마

```
CREATE TABLE BANK_ACCOUNT1 (
      ACCT_NUM VARCHAR(10) PRIMARY KEY,
      ACCT_BAL   NUMBER(10)
);
```

2.3 필드 버퍼

<demo.f>

```
#common
*base    100
ACCTNUM       1    string   -    -
AMOUNT        2    string   -    -
MESSAGE       3    string   -    -
```

2.4 클라이언트 프로그램

<bankcli.c>

```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <sample/fdl/demo_fdl.h>

int main(void) {

        FBUF *sendbuf, *recvbuf;
        long rlen;
        int retval = 0;
        char retdat[1024];
        char cont;
        char acct_num[100];
        char menu[20];
        char amount[100];

        printf("*** START BANK PROGRAM ***₩n");
        if (tpstart((TPSTART_T*)NULL) == -1) {
                fprintf(stderr, "Tpstart failed₩n");
                exit(1);
        }
        if ((sendbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
                fprintf(stderr, "Error allocation send buffer₩n");
                tpend();
                exit(1);
        }
        if ((recvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
                fprintf(stderr, "Error allocation recv buffer₩n");
```

```
                tpend();
                exit(1);
        }
        while(1) {
                printf("Please Input Your Account Number > ");
                scanf("%s", acct_num);

                fbput(sendbuf, ACCTNUM, acct_num, 0);

                printf("Input the Menu & Amount ex) DEPOSIT|WITHDRAW 1000 > ");
                scanf("%s %s", menu, amount);

                if(!strcmp(menu, "DEPOSIT") || !strcmp(menu, "WITHDRAW")) {
                        fbput(sendbuf, AMOUNT, amount, 0);
                        retval = tpcall(menu, (char *)sendbuf, 0, (char **)&recvbuf, &rlen,
TPNOFLAGS);
                        if(retval < 0) {
                                fprintf(stderr, "Cannot send request to service HELLOMSG-
>%s₩n", tpstrerror(tperrno));
                                tpfree((char *)sendbuf);
                                tpfree((char *)recvbuf);
                                tpend();
                                exit(1);
                        }
                        retval = fbget(recvbuf, MESSAGE, retdat, 0);
                        printf("%s", retdat);

                        printf("Continue: Press ANY key. Exit: n > ");
                        scanf("%*c%c", &cont);

                        if(cont == 'n') {
                                printf("*** END BANK PROGRAM ***₩n");
                                break;
                        }
                        fbinit(sendbuf, 1024);
                        fbinit(recvbuf, 1024);
                } else {
                        printf("Invalid Input!₩n");
                        tpfree((char *)sendbuf);
```

```
                        tpfree((char *)recvbuf);
                        tpend();
                        exit(1);
                }
        }

        tpfree((char *)sendbuf);
        tpfree((char *)recvbuf);

        tpend();
}
```

2.5 서버 프로그램

<banksvr.tbc>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <sample/fdl/demo_fdl.h>
#include <sqlca.h>

#define USERPASS "tibero/tmax"

EXEC SQL BEGIN DECLARE SECTION;
        char acct_num[100];
        int   acct_bal;
        int   acct_check;
        int   amount;
EXEC SQL END DECLARE SECTION;



void DEPOSIT(TPSVCINFO *tpsvcinfo) {

        FBUF *recvbuf = (FBUF *)tpsvcinfo->data;
        FBUF *sendbuf = NULL;
        char ret_msg[1024] = {'₩0',};
        char buf[1024];
```

```
	acct_check = 0;

	fbget(recvbuf, ACCTNUM, buf, 0);
	strcpy(acct_num, buf);

	EXEC SQL CONNECT :USERPASS;

	EXEC SQL
		SELECT COUNT(*) INTO :acct_check FROM BANK_ACCOUNT1 WHERE
ACCT_NUM = :acct_num;

	sendbuf = (FBUF *)tpalloc("FIELD", NULL, 0);

	if(acct_check == 0) {
		sprintf(ret_msg, "Wrong Account Number : %s₩n", acct_num);
		fbput(sendbuf, MESSAGE, ret_msg, 0);

		EXEC SQL COMMIT WORK RELEASE;

		tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
	} else {
		fbget(recvbuf, AMOUNT, buf, 0);
		amount = atoi(buf);

		EXEC SQL
			UPDATE BANK_ACCOUNT1 SET ACCT_BAL = ACCT_BAL + :amount
WHERE ACCT_NUM = :acct_num;

		sprintf(ret_msg, "SUCCESSFULLY COMPLETED₩n");
		fbput(sendbuf, MESSAGE, ret_msg, 0);

		EXEC SQL COMMIT WORK RELEASE;

		tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
	}

}
void WITHDRAW(TPSVCINFO *tpsvcinfo) {
```

```
	FBUF *recvbuf = (FBUF *)tpsvcinfo->data;
	FBUF *sendbuf = NULL;
	char ret_msg[1024] = {'₩0',};
	char buf[1024];

	acct_check = 0;

	fbget(recvbuf, ACCTNUM, buf, 0);
	strcpy(acct_num, buf);

	EXEC SQL CONNECT :USERPASS;

	EXEC SQL
		SELECT COUNT(*) INTO :acct_check FROM BANK_ACCOUNT1 WHERE
ACCT_NUM = :acct_num;

	sendbuf = (FBUF *)tpalloc("FIELD", NULL, 0);

	if(acct_check == 0) {
		sprintf(ret_msg, "Wrong Account Number : %s₩n", acct_num);
		fbput(sendbuf, MESSAGE, ret_msg, 0);

		EXEC SQL COMMIT WORK RELEASE;

		tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
	} else {
		fbget(recvbuf, AMOUNT, buf, 0);
		amount = atoi(buf);

		EXEC SQL
			SELECT ACCT_BAL INTO :acct_bal FROM BANK_ACCOUNT1 WHERE
ACCT_NUM = :acct_num;

		if(amount > acct_bal) {
			sprintf(ret_msg, "Cannot Withdraw (LOW BALANCE)₩n");
			fbput(sendbuf, MESSAGE, ret_msg, 0);

			EXEC SQL COMMIT WORK RELEASE;
```

```
                tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
        } else {
                EXEC SQL
                        UPDATE BANK_ACCOUNT1 SET ACCT_BAL = ACCT_BAL
- :amount WHERE ACCT_NUM = :acct_num;

                sprintf(ret_msg, "SUCCESSFULLY COMPLETED₩n");
                fbput(sendbuf, MESSAGE, ret_msg, 0);

                EXEC SQL COMMIT WORK RELEASE;

                tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
        }
    }
}
```

3. 수행 과정

3.1 환경 설정(sample.m) 적용

```
tmdown
```

```
ko@ko-tmax:~/oframe_7/tmax/config$ cfl -i sample.m
CFL is done successfully for node(NODE1)
```

3.2 서비스 테이블(kotmaxsvr_svctab.c) 생성

```
ko@ko-tmax:~/oframe_7/tmax/config$ gst
SVC tables are successfully generated
GST is successfully done
```

```
ko@ko-tmax:~/oframe_7/tmax/svct$ ls
kotmaxsvr_svctab.c  obmjinit_svctab.c  obmjspbk_svctab.c
koxasvr_svctab.c    obmjmsvr_svctab.c  obmtsmgr_svctab.c
obmjhist_svctab.c   obmjschd_svctab.c  ofrcmsvr_svctab.c
```

3.3 서비스 테이블(kotmaxsvr_svctab.c) 컴파일(오브젝트 파일 생성)

```
cc -c -I$TMAXDIR/svct/kotmaxsvr_svctab.c
```

3.4 필드 버퍼(demo.f) 컴파일

```
ko@ko-tmax:~/oframe_7/tmax/sample/fdl$ fdlc -c -i demo.f
FDLC is successfully done
ko@ko-tmax:~/oframe_7/tmax/sample/fdl$ ls
demo.f  demo_fdl.h  tmax.fdl
```

3.5 클라이언트 프로그램 컴파일

```
ko@ko-tmax:~/oframe_7/tmax/sample/client$ cc -O -I$TMAXDIR -c bankcli.c
cc -O -I$TMAXDIR -L$TMAXDIR/lib -o bank bankcli.o -lcli -lnsl
```

3.6 서버 프로그램 프리 컴파일 및 컴파일

```
tbpc INCLUDE=$(TMAXDIR) INCLUDE=$(TB_HOME)/client/include kotmaxsvr.tbc
cc -c -I$(TB_HOME)/client/include -I$(TMAXDIR) kotmaxsvr.c -L$(TMAXDIR)/lib -lsvr -lnodb -L$(TB_HOME)/client/lib -ltbertl -ltbcli
cc -o kotmaxsvr kotmaxsvr.o $(TMAXDIR)/lib/sdl.o kotmaxsvr_svctab.o -L$(TMAXDIR)/lib -lsvr -lnodb -L$(TB_HOME)/client/lib -ltbertl -ltbcli

cp kotmaxsvr $(TMAXDIR)/appbin
tmboot
```

4. 실행 결과

4.1 Tmax 서버

```
ko@ko-tmax:~/oframe_7/tmax/sample/server$ tmadmin
--- Welcome to Tmax Admin (Type "quit" to leave) ---

$$1 NODE1 (tmadm): si
------------------------------------------------------------------------
  clh   svrname    (svri)   status     count   qcount   qpcount  emcount
------------------------------------------------------------------------
    0   kotmaxsvr  (   4)      RDY         1        0         0        0
```

4.2 클라이언트 프로그램

```
ko@ko-tmax:~/oframe_7/tmax/sample/client$ ./bank
*** START BANK PROGRAM ***
Please Input Your Account Number > 11111
Input the Menu & Amount ex) DEPOSIT|WITHDRAW 1000 > DEPOSIT 5000
SUCCESSFULLY COMPLETED
Continue: Press ANY key. Exit: n > y
Please Input Your Account Number > 11111
Input the Menu & Amount ex) DEPOSIT|WITHDRAW 1000 > WITHDRAW 2000
SUCCESSFULLY COMPLETED
Continue: Press ANY key. Exit: n > n
*** END BANK PROGRAM ***
```

4.3 데이터베이스

```
SQL> select * from bank_account1;

ACCT_NUM     ACCT_BAL
---------- ----------
11111            3000

1 row selected.
```

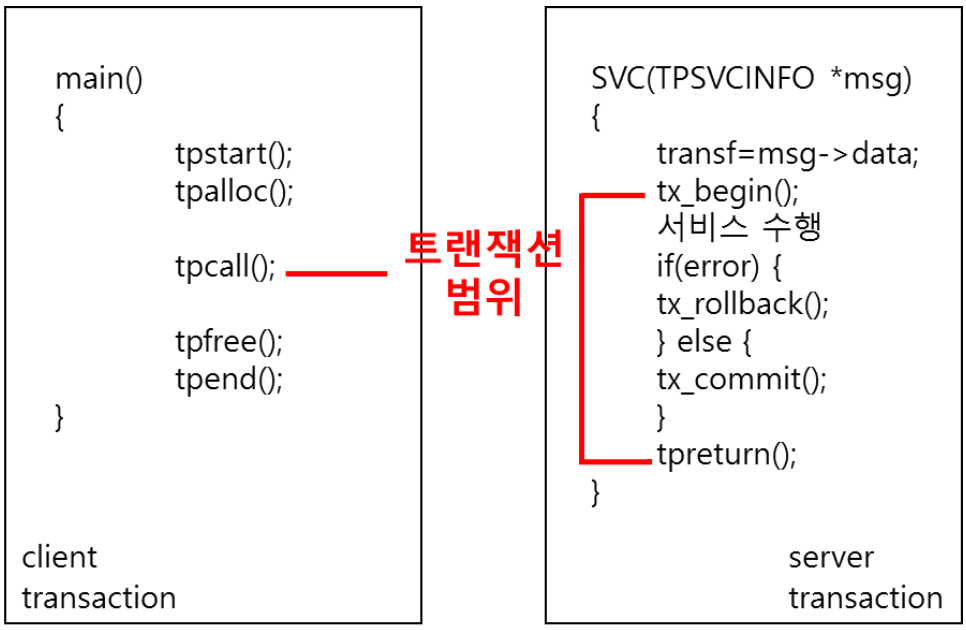
과제5. XA 모드의 계좌 입출금 서비스 구현

1. 개요

본 과제는 tbESQL/C와 Tmax 서버를 이용한 계좌 입출금 서비스를 **XA 모드**로 구현하는 것을 목적으로 한다. 이를 위해 ESQL 문이 삽입된 C 기반의 클라이언트/서버 프로그램을 각각 구현하고, 구현된 서버 프로그램을 Tmax 서버에 올려 클라이언트의 요청을 처리한다. 클라이언트-서버 간 통신은 동기형 통신을 사용하고, 서버 프로그램의 구현은 TCS로 한다. 또한, 서버 프로그램과 연동할 데이터베이스로는 Tibero 6를 사용한다.

1.1 XA 모드

XA 모드의 Tmax 시스템은 별도의 트랜잭션 관리자(TMS)를 두어 트랜잭션 전체를 관리하도록 한다. XA 모드로 운영되는 경우 모든 트랜잭션은 전역 트랜잭션으로 간주되며 데이터의 무결성 확보를 위해 2PC(2 Phase Commit)를 사용한다.



2. 구성

프로그램 파일	설명
demo.f	필드 버퍼
sample.m	환경 설정
tbs_tbr.mk	TMS Makefile for Tibero
dumy.c	TMS 생성을 위한 dummy source
bankclixa.c	클라이언트 프로그램
koxasvr.tbc	서버 프로그램

2.1 환경 설정

<rc-oframe-7>, <tbpc.cfg>

Non-XA 모드와 동일

<$TB_HOME/client/config/tbdsn.tbr>

```
tb_oframe7=(
    (INSTANCE=(HOST=localhost)
            (PORT=8629)
            (DB_NAME=tb_oframe7)
    )
)

tb_start=(
    (INSTANCE=(HOST=192.168.4.16)
            (PORT=8941)
            (DB_NAME=tb_start)
    )
)
```

<sample.m>

```
*DOMAIN
kotmax
    SHMKEY      = 94000,    MAXUSER     = 256,    MINCLH      = 1,
    MAXCLH      = 1,    BLOCKTIME   = 15,    MAXCPC      = 256,
    MAXSPR      = 512,    MAXSVR      = 128,    MAXSVC      = 512,
    DOMAINID    = 4,    IPCPERM     = 0777,    MAXSACALL   = 1024,
    MAXCACALL   = 1024

*NODE
DEFAULT:
    HOSTNAME = "ko-tmax",
    DOMAINNAME = "kotmax"
NODE1
    TMAXHOME = "/home/ko/oframe_7/tmax",
    TMAXDIR = "/home/ko/oframe_7/tmax",
    APPDIR  = "/home/ko/oframe_7/tmax/appbin",
    TLOGDIR = "/home/ko/oframe_7/tmax/log/tlog",
    ULOGDIR = "/home/ko/oframe_7/tmax/log/ulog",
    SLOGDIR = "/home/ko/oframe_7/tmax/log/slog",
```

```
    CLHOPT = " -o /home/ko/oframe_7/tmax/log/clh.log -e
/home/ko/oframe_7/tmax/log/clh.err",
    TPORTNO = 9999, SHMKEY = 94000, RACPORT = 9450

*SVRGROUP
svg_s1
    NODENAME = "NODE1",
    SVGTYPE  = STMAX,
    DBNAME   = TIBERO,
    OPENINFO = "TIBERO_XA:user=tibero,pwd=tmax,sestm=60,db=tb_oframe7,conn_id=db1",
    TMSNAME  = tms_tbr,
    RMID     = 1

svg_s2
    NODENAME = "NODE1",
    SVGTYPE  = STMAX,
    DBNAME   = TIBERO,
    OPENINFO = "TIBERO_XA:user=tibero,pwd=tmax,sestm=60,db=tb_start,conn_id=db2",
    TMSNAME  = tms_tbr,
    RMID     = 2

svg_xa
    NODENAME = "NODE1",
    SVGTYPE  = MTMAX,
    SVGLIST  = "svg_s1, svg_s2"

*SERVER
koxasvr         SVGNAME = svg_xa,
                    MIN = 1, MAX = 1,
                    CLOPT="-o $(SVR)$(DATE).out -e $(SVR)$(DATE).err"



*SERVICE
KOXASVC         SVRNAME = koxasvr
```

2.2 TMS

<dumy.c>

```
int i;
```

<tms_tbr.mk>

```
# TMS Makefile for Tibero
# Linux

TBLIBDIR = $(TB_HOME)/client/lib
TBLIB =-ltbxa -ltbertl -ltbcli -lm -lpthread

TARGET    = tms_tbr
APOBJ     = dumy.o

APPDIR    = $(TMAXDIR)/appbin
TMAXLIBD= $(TMAXDIR)/lib
TMAXLIBS= -ltms -ltbs

CFLAGS    =
LDFLAGS =

all : $(TARGET)

$(TARGET): $(APOBJ)
        $(CC) $(CFLAGS) $(LDFLAGS) -o $(TARGET) $(TMAXINC) -L$(TMAXLIBD) $(TMAXLIBS)
$(APOBJ) -L$(TBLIBDIR) $(TBLIB)
        mv $(TARGET) $(APPDIR)/.

$(APOBJ):
        $(CC) $(CFLAGS) $(LDFLAGS) -c dumy.c

clean:
        -rm -f *.o core $(APPDIR)/$(TARGET)
```

2.3 데이터베이스 스키마

db1(tb_oframe):

```
CREATE TABLE BANK_ACCOUNT1 (
        ACCT_NUM VARCHAR(10) PRIMARY KEY,
        ACCT_BAL    NUMBER(10) );
```

db2(tb_start):

```
CREATE TABLE BANK_ACCOUNT2 (
        ACCT_NUM VARCHAR(10) PRIMARY KEY,
        ACCT_BAL    NUMBER(10) );
```

2.4 필드 버퍼

<demo.f>

```
#common
*base    100
ACCTNUM      1    string   -    -
AMOUNT       2    string   -    -
MESSAGE      3    string   -    -
TARGETNUM    4    string   -    -
```

2.5 클라이언트 프로그램

<bankclixa.c>

```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <sample/fdl/demo_fdl.h>

int main(void) {

        FBUF *sendbuf, *recvbuf;
        long rlen;
        int retval = 0;
        char retdat[1024];
        char cont;
        char acct_num[100], menu[20], amount[100];

        printf("*** START ACCOUNT TRANSFER PROGRAM ***₩n");

        if ((sendbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
                fprintf(stderr, "Error allocation send buffer₩n");
                tpend();
                exit(1);
        }
        if ((recvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
                fprintf(stderr, "Error allocation recv buffer₩n");
                tpend();
                exit(1);
        }
        while(1) {
```

```
                printf("[YOUR] Account Number > ");
                scanf("%s", acct_num);
                fbput(sendbuf, ACCTNUM, acct_num, 0);

                printf("[TARGET] Account Number > ");
                scanf("%s", acct_num);
                fbput(sendbuf, TARGETNUM, acct_num, 0);

                printf("Please Input Amount > ");
                scanf("%s", amount);
                fbput(sendbuf, AMOUNT, amount, 0);

                retval = tpcall("KOXASVC", (char *)sendbuf, 0, (char **)&recvbuf, &rlen,
TPNOFLAGS);

                if(retval < 0) {
                        fprintf(stderr, "Cannot send request to service ->%s₩n",
tpstrerror(tperrno));
                        tpfree((char *)sendbuf);
                        tpfree((char *)recvbuf);
                        tpend();
                        exit(1);
                }
                retval = fbget(recvbuf, MESSAGE, retdat, 0);
                printf("%s", retdat);
                printf("Continue: Press ANY key. Exit: n > ");
                scanf("%*c%c", &cont);
                if(cont == 'n') {
                        printf("*** END ACCOUNT TRANSFER PROGRAM ***₩n");
                        break;
                }
                fbinit(sendbuf, 1024);
                fbinit(recvbuf, 1024);
        }

        tpfree((char *)sendbuf);
        tpfree((char *)recvbuf);
        tpend();
}
```

2.6 서버 프로그램

<koxasvr.tbc>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tmaxapi.h>
#include <usrinc/tx.h>
#include <sample/fdl/demo_fdl.h>
#include <sqlca.h>

EXEC SQL BEGIN DECLARE SECTION;
        char acct_num[100];
        int  acct_bal;
        int  acct_check;
        int  amount;
        char conn_id[1024];
EXEC SQL END DECLARE SECTION;

KOXASVC(TPSVCINFO *tpsvcinfo) {

        int w_retval = 0, d_retval = 0;
        FBUF *recvbuf = (FBUF *)tpsvcinfo->data;
        FBUF *sendbuf = NULL;
        char ret_msg[1024] = {'₩0',};
        char buf[1024];

        sendbuf = (FBUF *)tpalloc("FIELD", NULL, 0);

        tx_begin();

        fbget(recvbuf, ACCTNUM, buf, 0);
        strcpy(acct_num, buf);

        fbget(recvbuf, AMOUNT, buf, 0);
        amount = atoi(buf);
```

```
	w_retval = WITHDRAW();

	fbget(recvbuf, TARGETNUM, buf, 0);
	strcpy(acct_num, buf);

	d_retval = DEPOSIT();

	if((w_retval * d_retval) == 0) {
		tx_rollback();
		sprintf(ret_msg, "FAILED: PLEASE CHECK YOUR ACCOUNT₩n");
		fbput(sendbuf, MESSAGE, ret_msg, 0);
		tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
	}

	tx_commit();

	sprintf(ret_msg, "SUCCESSFULLY COMPLETED₩n");
	fbput(sendbuf, MESSAGE, ret_msg, 0);

	tpreturn(TPSUCCESS, 0, (char *)sendbuf, 0, TPNOFLAGS);
}

int WITHDRAW() {

	acct_check = 0;

	strcpy(conn_id, "db1");
	EXEC SQL XA SET CONNECTION AT :conn_id;

	EXEC SQL
		SELECT COUNT(*) INTO :acct_check FROM BANK_ACCOUNT1 WHERE
ACCT_NUM = :acct_num;

	if(acct_check == 0) {
		return 0;
	} else {
		EXEC SQL
			SELECT ACCT_BAL INTO :acct_bal FROM BANK_ACCOUNT1 WHERE
ACCT_NUM = :acct_num;
```

```
                if(amount > acct_bal) {
                        return 0;
                } else {
                        EXEC SQL
                                UPDATE BANK_ACCOUNT1 SET ACCT_BAL = ACCT_BAL
- :amount WHERE ACCT_NUM = :acct_num;
                }
        }

        return 1;
}

int DEPOSIT() {

        acct_check = 0;

        strcpy(conn_id, "db2");
        EXEC SQL XA SET CONNECTION AT :conn_id;

        EXEC SQL
                SELECT COUNT(*) INTO :acct_check FROM BANK_ACCOUNT2 WHERE
ACCT_NUM = :acct_num;

        if(acct_check == 0) {
                return 0;
        } else {
                EXEC SQL
                        UPDATE BANK_ACCOUNT2 SET ACCT_BAL = ACCT_BAL + :amount
WHERE ACCT_NUM = :acct_num;
        }

        return 1;
}
```

3. 수행 과정

3.1 환경 설정(sample.m) 적용

tmdown

```
ko@ko-tmax:~/oframe_7/tmax/config$ cfl -i sample.m
CFL is done successfully for node(NODE1)
```

3.2 서비스 테이블(koxasvr_svctab.c) 생성

```
ko@ko-tmax:~/oframe_7/tmax/config$ gst
SVC tables are successfully generated
GST is successfully done
```

```
ko@ko-tmax:~/oframe_7/tmax/svct$ ls
kotmaxsvr_svctab.c  obmjinit_svctab.c  obmjspbk_svctab.c
koxasvr_svctab.c    obmjmsvr_svctab.c  obmtsmgr_svctab.c
obmjhist_svctab.c   obmjschd_svctab.c  ofrcmsvr_svctab.c
```

3.3 서비스 테이블(koxasvr_svctab.c) 컴파일(오브젝트 파일 생성)

cc -c -I$TMAXDIR/svct/koxasvr_svctab.c

3.4 필드 버퍼(demo.f) 컴파일

```
ko@ko-tmax:~/oframe_7/tmax/sample/fdl$ fdlc -c -i demo.f
FDLC is successfully done
ko@ko-tmax:~/oframe_7/tmax/sample/fdl$ ls
demo.f  demo_fdl.h  tmax.fdl
```

3.5 TMS 컴파일

make -f tms_tbr.mk all

3.6 클라이언트 프로그램 컴파일

```
ko@ko-tmax:~/oframe_7/tmax/sample/client$ cc -O -I$TMAXDIR -c bankclixa.c
cc -O -I$TMAXDIR -L$TMAXDIR/lib -o transfer bankclixa.o -lcli -lnsl
```

3.7 서버 프로그램 프리 컴파일 및 컴파일

```
tbpc INCLUDE=$(TMAXDIR) INCLUDE=$(TB_HOME)/client/include koxasvr.tbc
cc -c -I$(TB_HOME)/client/include -I$(TMAXDIR) koxasvr.c -L$(TMAXDIR)/lib -lsvr -ltbs -L$(TB_HOME)/client/lib -ltbertl -ltbcli -ltbxa
cc -o koxasvr koxasvr.o $(TMAXDIR)/lib/sdl.o koxasvr_svctab.o -L$(TMAXDIR)/lib -lsvr -ltbs -L$(TB_HOME)/client/lib -ltbertl -ltbcli -ltbxa

cp koxasvr $(TMAXDIR)/appbin
tmboot
```

4. 실행 결과

4.1 Tmax 서버

```
ko@ko-tmax:~/oframe_7/tmax/sample/server$ tmadmin
--- Welcome to Tmax Admin (Type "quit" to leave) ---

$$1 NODE1 (tmadm): si
------------------------------------------------------------------------
  clh   svrname    (svri)   status     count   qcount   qpcount  emcount
------------------------------------------------------------------------
    0   kotmaxsvr  (   4)      RDY         1        0         0        0
    0   koxasvr    (   5)      RDY         0        0         0        0
```

4.2 클라이언트 프로그램

```
ko@ko-tmax:~/oframe_7/tmax/sample/client$ ./transfer
*** START ACCOUNT TRANSFER PROGRAM ***
[YOUR] Account Number > 11111
[TARGET] Account Number > 22222
Please Input Amount > 500
SUCCESSFULLY COMPLETED
Continue: Press ANY key. Exit: n > n
*** END ACCOUNT TRANSFER PROGRAM ***
```

4.3 데이터베이스

```
SQL> select * from bank_account1;

ACCT_NUM     ACCT_BAL
---------- ----------
11111            9500

1 row selected.

SQL> select * from bank_account2;

ACCT_NUM     ACCT_BAL
---------- ----------
22222             500

1 row selected.
```

5. 이슈 사항

- Tmax Programming Guide(MultipleRM)에는 MultipleRM 서버 프로그램 컴파일 시 -lnodb 옵션을 사용해 $TMAXDIR/lib 경로에 있는 libnodb.so 라이브러리를 참조해야 한다고 되어 있다. 하지만, 실제로 이 라이브러리를 참조할 경우 XA 서버 프로그램 컴파일을 위해 필요한 옵션인 -ltbxa를 통한 libtbxa.so 라이브러리 참조가 정상적으로 되지 않는다. 샘플 Makefile을 통해 -lnodb 옵션 대신 -ltbs 옵션을 사용함으로써 해결하였지만 원인은 불분명하다.

Makefile

Makefile의 TMAXLIBS에 반드시 **-lnodb**를 포함해야 한다. 다음은 64bit Linux에서 MultipleRM 서버 프로그램을 컴파일하기 위한 Makefile의 예이다.

- XA 트랜잭션 제어 구문이 포함된 프로그램 소스 내에 ESQL DB 트랜잭션 제어 구문이 있을 경우 트랜잭션에 대한 처리는 되지만 Tibero 내부적으로 다음과 같은 에러가 발생한다.

```
EXEC SQL COMMIT WORK RELEASE;
```

<$TB_HOME/instance/tb_oframe7/log/tracelog/trace.log>

```
07/21 14:49:22.593885 [FRM] 82(82) tbsvr_er:066 THROW. ec=ERROR_SESS_TCS_NOT_ALLOWED(-12014) [ Transaction control state
ments are not allowed in XA transactions.   ] (csr_id:4294967295) [tbsvr_tcs.c:100:tbsvr_commit_internal]
```

- tbsql 을 통해 데이터베이스 내용을 수정하는 UPDATE 쿼리를 직접 수행한 후 tbsql 을 종료(quit) 하지 않고, XA 서버 프로그램을 이용할 경우 타임아웃이 발생한다.