인터페이스/연동

ODBC 설정방법

(Linux/Unix 계열)

TMAXTibero

TIITI060

안내서 정보
안내서 제목: ODBC_설정방법(Linux)
발행일: 2025-12-15
소프트웨어 버전: 7FS02PS04
안내서 버전: 1.1.0

제, 개정 이력

안내서 버전	개정일자	개정 사유 및 내용	비고
1.0.0	2024.02.28	최초 제정	
1.1.0	2025.12.15.	기존 자료 최신화	

Linux/Unix 환경에서 ODBC 환경 구성

1. 개요

본 문서에서는 Linux/Unix 환경에서 ODBC 를 이용하여 Tibero 를 사용하기 위한 환경을 구성하는 방법에 대해서 설명한다.
Windows 와 달리 Linux/Unix 의 경우는 ODBC Driver Manager 가 기본설치 되어 있지 않은 경우가 많아 별도의 설치 작업이 필요하다.
기본적인 ODBC 설정방법에 대한 가이드를 함께 제공하도록 한다.

1.1 테스트 환경

본 테스트는 Linux 64bit 환경에서 이루어졌으며 ODBC Driver Manager 로 unixODBC 를 사용하여 진행되었다.
기본 Shell 로 bash 를 사용하였고, 기본 Shell 변경에 따른 설정방법의 변경은 본 문서에서 논외로 하도록 한다.
본 테스트환경은 Tibero 가 설치되지 않은 환경이므로 TB_HOME 등의 Tibero 관련 환경변수가 설정되지 않은 상태에서 진행되었다.
타 Driver Manager 의 경우는 설치 방법 정도에 차이가 있을 뿐 사용방법은 유사하므로 관련 문서를 참조하기 바란다.

2. ODBC Driver Manager 설치

2.1 unixODBC 소스코드 다운로드

http://www.unixodbc.org/를 방문하여 최신 버전을 다운로드 한다.

2.2 unixODBC 설치

2.2.1 2.1에서 다운로드 한 소스코드의 압축을 풀어준다.

```
[tibero@cwcordbp01:PS04:/home/tibero]tar -xzvf unixODBC-2.3.14.tar.gz
unixODBC-2.3.14/
unixODBC-2.3.14/depcomp
unixODBC-2.3.14/exe/
unixODBC-2.3.14/exe/isql.c
unixODBC-2.3.14/exe/COPYING-- 중    략 --
```

2.2.2 configure를 실행하여 설치를 위한 환경설정을 진행한다.

```
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]./configure --prefix=/home/tibero/unixodbc
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
```

```
checking whether make sets $(MAKE)... yes-- 중    략 --
configure: creating ./config.status
config.status: creating Makefile
config.status: creating config.h
config.status: executing depfiles commands
config.status: executing libtool commands
checking are we setting library version ... no
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]
```

2.2.3 소스코드를 컴파일 한다.

```
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]                    make
make  all-recursive
make[1]: Entering directory '/home/tibero/unixODBC-2.3.14'
Making all in extras
make[2]: Entering directory '/home/tibero/unixODBC-2.3.14/extras'
/bin/sh ../libtool  --tag=CC   --mode=compile gcc -DHAVE_CONFIG_H -I. -I..  -I../include   -g -O2 -pthread -MT strcasecmp.lo
-MD -MP -MF .deps/strcasecmp.Tpo -c -o strcasecmp.lo strcasecmp.c
-- 중    략 --
Making all in samples
make[2]: Entering directory '/home/tibero/unixODBC-2.3.14/samples'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/tibero/unixODBC-2.3.14/samples'
make[2]: Entering directory '/home/tibero/unixODBC-2.3.14'
make[2]: Leaving directory '/home/tibero/unixODBC-2.3.14'
make[1]: Leaving directory '/home/tibero/unixODBC-2.3.14'
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]
```

2.2.4 2.2.2에서 prefix로 지정한 위치에 설치한다.

```
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]make install
Making install in extras
make[1]: Entering directory '/home/tibero/unixODBC-2.3.14/extras'
make[2]: Entering directory '/home/tibero/unixODBC-2.3.14/extras'
-- 중    략 --
cp unixodbc_conf.h /home/tibero/unixodbc/include/unixodbc_conf.h
make[2]: Leaving directory `/home/tibero/unixodbc-2.3.9'
make[1]: Leaving directory `/home/tibero/unixodbc-2.3.9'
[tibero@cwcordbp01:PS04:/home/tibero/unixODBC-2.3.14]
```

2.2.4 Linux 이외의 Unix 에서의 설치

Unix 에서도 source compile 을 통한 설치 절차는 Linux 와 동일하다.

다면 C Compiler / Linker 의 옵션 등이 상이할 수 있으며 이에 맞는 설정이 추가로 필요할 수도 있다.

2.2.4.1 AIX 7.1 에서 64bit로 컴파일하기

Compiler 와 Linker 등에서 64bit 모드를 사용할 수 있도록 설정하는 방법 중 가장 간편한 방법은 OBJECT_MODE=64 를 환경변수로 등록해 준다.

```
$ export OBJECT_MODE=64
$ ./configure --prefix=/home/tibero/unixodbc
$ make
$ make install
```

단, 이 경우 다른 32bit Application 의 동작에 영향을 줄 수 있으므로 주의해서 사용해야 한다.

2.3 unixODBC 환경 설정

Tibero 관련 환경변수 (TB_HOME 등)는 설정되지 않은 채로 진행되었다.
색깔로 표기된 부분은 각기 다른 설정파일 내에서 반드시 동일하게 설정하도록 한다.

2.3.1 OS 환경변수 설정

환경변수 설정을 위해 .bash_profile 에 아래 내용을 추가한다.

```
####### UNIXODBC SETTING ######
export UNIXODBC_HOME=/home/tibero/unixodbc
export LD_LIBRARY_PATH=$UNIXODBC_HOME/lib:$LD_LIBRARY_PATH
export PATH=$UNIXODBC_HOME/bin:$PATH
export ODBCINI=/home/tibero/odbc_config/odbc.ini
export ODBCSYSINI=/home/tibero/odbc_config
```

ODBCINI 환경변수를 이용하여 사용하려고 하는 odbc.ini 파일명의 절대경로를 명시적으로 정의한다.
(Default : $UNIXODBC_HOME/etc/odbc.ini)
ODBCSYSINI 환경변수를 이용하여 사용하려고 하는 odbcinst.ini 파일이 존재하는 디렉토리를 명시적으로 정의한다.
UnixODBC 설치시 별도의 옵션을 지정하지 않을 경우 $UNIXODBC_HOME/etc 를 default 로 사용한다.

* 간혹 Driver Manager 가 사용자가 기대하는 위치가 아닌 다른 위치의 odbc.ini 파일을 참조하는 경우 Data source name not found and no default driver specified 와 같은 에러가 발생할 수 있다. 이 경우 **ODBCINI** 환경변수를 통해 명시적으로 사용할 odbc.ini 파일을 지정하면 문제를 해결할 수 있다.

가능하면 ODBCINI 와 ODBCSYSINI 환경변수를 설정하여 사용하는 것을 권고한다.

2.3.2 Tibero ODBC Driver 파일 복사

Tibero ODBC Driver 인 libtbodbc.so 파일을 특정 Directory 에 복사하고 read/execute 권한을 부여한다.
본 가이드에서는 $UNIXODBC_HOME/lib 에 복사해 두었다.
([tibero@cwcordbp01:PS04:/Tibero_Engine/tibero7.2.4/client/lib]cp -p libtbodbc.so $UNIXODBC_HOME/lib)
Tibero ODBC Driver 는 현재 플랫폼과 동일한 Tibero Server 에서 복사해서 사용하면 된다.
($TB_HOME/client/lib/libtbodbc.so 파일)

2.3.3 ODBC Driver 설정 (odbcinst.ini)

ODBC Driver 대한 설정을 위해 $ODBCSYSINI/odbcinst.ini 파일을 작성한다.

```
[ODBC]
Trace=yes
TraceFile=/home/tibero/unixodbc/log/traceFile.log
[Tibero 7 ODBC driver]
Description     = Tibero ODBC driver for Tibero 7
Driver          = /home/tibero/unixodbc/lib/libtbodbc.so
```

ODBC Application 의 동작을 추적하려면 [ODBC] 섹션을 설정하여 trace 파일을 남길 수 있다.
[ODBC] 섹션의 설정은 Driver 를 사용하는 모든 ODBC Application 에 공통으로 적용된다.

각 Driver 에 대한 설정은 [Tibero 6 ODBC driver] 와 같이 Driver 별로 섹션을 정의해서 사용한다.
여기에서 Driver 는 DB 벤더에서 제공하는 ODBC Driver 의 절대 경로를 지정해 준다.
Driver 섹션은 여러 개를 설정할 수 있다.
odbcinst.ini 파일이 있는 위치를 ODBCSYSINI 환경변수를 이용하여 명시적으로 정의해 주는 것을 권고한다.
파일명이 아니라 odbcinst.ini 파일이 위치하는 디렉토리를 지정해 준다.

2.3.4 Data Source 설정 (odbc.ini)

```
[tbodbc]
Driver = Tibero 7 ODBC driver
SERVER = 192.168.153.129
PORT = 19845
DATABASE = PS04
USER=tibero
PASSWORD=tmax
```

Driver 는 2.3.3 에서 설정한 이름과 동일하여야 한다. (또는 드라이버 파일의 절대 경로를 적어준다)
USER, PASSWORD 는 기존에 DB 에 설정한 USER 와 PASSWORD 를 넣어주면 해당 유저로 접속한다.
(ex : Driver = /home/tibero/unixodbc/lib/libtbodbc.so)
Data Source 설정에서 Tibero Server 에 접속하기 위해서 아래와 같이 SERVER, PORT, DATABASE 정보가 **한 세트로 존재**하여야 한다.
SERVER : Tibero Server 의 IP 주소
PORT : Tibero Server 의 Listener Port
DATABASE : Tibero Server 의 DB_NAME (DATABASE 대신 DB 로 정의해도 된다.)

아래 로그인 정보는 ODBC Application 에서 SQLConnect 함수의 ID, PW 로 전달된다.
USER : DB 접속을 위한 사용자 계정 (USER 대신 UID 로 정의해도 된다.)
PASSWORD : DB 접속을 위한 사용자 암호 (PASSWORD 대신 PWD 로 정의해도 된다.)

odbc.ini 파일은 아래 순서대로 존재 유무를 확인하고 파일이 존재하면 그 안에서 Data Source 를 찾게 된다.

A. ODBCINI 환경변수로 지정된 파일
B. $HOME/.odbc.ini 파일 : OS User 의 HOME 디렉토리 하위에 히든 파일로 존재 (파일명 앞에 . 이 있음)

C. /etc/odbc.ini 파일

따라서 2.3.1 에서 설명한 것처럼 가능하면 ODBCINI 환경변수를 설정하여 사용할 odbc.ini 파일을 지정할 것을 권고한다.

A 는 명시적으로 User Data Source 의 위치를 지정한 것이고, **B** 는 특별한 설정이 없으면 참조하는 User Data Source 이다.

2.3.5 ODBC 설정 확인

odbcinst 명령을 통해서 현재 ODBC 환경정보를 확인할 수 있다.

```
[tibero@cwcordbp01:PS04:/home/tibero]odbcinst -j  # 현재 설정 정보를 확인할 수 있다.
unixODBC 2.3.14
DRIVERS............: /home/tibero/odbc_config/odbcinst.ini
SYSTEM DATA SOURCES: /home/tibero/odbc_config/odbc.ini
FILE DATA SOURCES..: /home/tibero/odbc_config/ODBCDataSources
USER DATA SOURCES..: /home/tibero/odbc_config/odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8
[tibero@cwcordbp01:PS04:/home/tibero] odbcinst -q -d     # odbcinst.ini 에 정의된 드라이버 정보를 확인할 수 있다.
[Tibero 7 ODBC driver]
[tibero@cwcordbp01:PS04:/home/tibero] odbcinst -q -s   # odbc.ini 에 정의된 Data Source 의 정보를 확인할 수 있다.
[tbodbc]
```

2.4 ODBC 환경 동작 테스트

2.4.1 isql 툴을 사용한 테스트

```
[tibero@cwcordbp01:PS04:/home/tibero/unixodbc/bin]  isql tbodbc
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| echo [string]                         |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> select sysdate from dual;
```

```
+--------------------+
| SYSDATE            |
+--------------------+
| 2025/12/14 15:52:57|
+--------------------+
SQLRowCount returns 1
1 rows fetched
SQL> quit
```

2.4.2 간단한 ODBC Sample 프로그램을 사용한 테스트

아래와 같은 Sample Program 을 test.c 파일로 작성한다.

```
#include <stdio.h>
#include <stdlib.h>
#include <sqlext.h>
#include <sql.h>

#define COL_LEN 30

#define _DEBUG_

int main(int argc, char* argv[])
{
  SQLRETURN rc = SQL_SUCCESS;
  SQLHANDLE henv, hdbc, hstmt;
  SQLSMALLINT wResultCols;
  int Pos;
  SQLCHAR ColName[1024];
  SQLSMALLINT     ColLen;
  SQLSMALLINT ColType=0;
  SQLULEN Precision=0;
  SQLSMALLINT Scale=0;
  SQLSMALLINT NullOk;

 SQLCHAR *sql = (SQLCHAR *)"select to_char(sysdate,'yyyy/mm/dd hh24:mi:ss') as datetime,'ODBC TEST Application' as
name from dual;";
  char buf[128];
  SQLCHAR DATETIME[COL_LEN], NAME[COL_LEN];
  SQLLEN dtlen = 0, namelen = 0;

  /* Memory allocation for ODBC Connection */
  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
  if(rc==SQL_SUCCESS || rc ==SQL_SUCCESS_WITH_INFO)
    printf("Memory allocation for ODBC Env. (SQLAllocHandle) : Success₩n");
```

```
else
  printf("Memory allocation for ODBC Env. (SQLAllocHandle) : Failed₩n");

rc = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
if(rc==SQL_SUCCESS || rc ==SQL_SUCCESS_WITH_INFO)
  printf("ODBC Attributes asignment (SQLSetEnvAttr) : Success₩n");
else
  printf("ODBC Attributes asignment (SQLSetEnvAttr) : Failed₩n");

rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
if(rc==SQL_SUCCESS || rc ==SQL_SUCCESS_WITH_INFO)
  printf("Memory allocation for Connection (SQLAllocHandle) : Success₩n");
else
  printf("Memory allocation for Connection (SQLAllocHandle) : Failed₩n");

/* Tibero Connect */
rc = SQLConnect(hdbc,
      (SQLCHAR *)"tbodbc", SQL_NTS,       // odbc.ini 에 등록된 Data Source 명과 동일하게 설정
      NULL, 0,
      NULL, 0);

if (rc != SQL_SUCCESS || rc == SQL_SUCCESS_WITH_INFO) {
  fprintf(stderr, "Connection failed!!![%d]₩n", rc);
  exit(1);
}

/* Memory allocation for Statements */
SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
printf("Query: %s₩n", sql);

/* Execute query */
rc = SQLExecDirect(hstmt, sql, SQL_NTS);
if (rc != SQL_SUCCESS) {
  fprintf(stderr, "SQLExecDirect failed!!!₩n");
  exit(1);
}

/* Bind for result */
SQLBindCol(hstmt, 1, SQL_C_CHAR, DATETIME, COL_LEN, &dtlen);
SQLBindCol(hstmt, 2, SQL_C_CHAR, NAME, COL_LEN, &namelen);

printf( "₩n₩t[ Column Attribute ]₩n" );
SQLNumResultCols( hstmt, &wResultCols);
printf("wResultCols : %d₩n", wResultCols);
```

```
  printf("Pos  ColName            ColType ( Precision, Scale)₩n");
  for(Pos = 1; Pos <= wResultCols; Pos++)
  {
    rc = SQLDescribeCol(hstmt,
          Pos,
          ColName,
          sizeof(ColName),
          (SQLSMALLINT*)&ColLen,
          (SQLSMALLINT*)&ColType,
          (SQLULEN*)&Precision,
          (SQLSMALLINT*)&Scale,
          (SQLSMALLINT*)&NullOk);
    printf("%3d. %-15s : %-3d (%ld, %d) %d₩n", Pos, ColName, ColType, Precision, Scale, ColLen);
  }

  /* Fetch resultset */
  while(SQLFetch(hstmt) != SQL_NO_DATA) {
    printf("₩nResult : %s₩t%s₩n", DATETIME,NAME);
  }
  printf("₩nEND Program₩n");

  /* Deallocate handles and disconnect */
  SQLFreeStmt(hstmt, SQL_DROP);
  SQLDisconnect(hdbc);
  SQLFreeConnect(hdbc);
  SQLFreeEnv(henv);
  return 0;
}
```

아래와 같은 명령어를 사용하여 실행 가능한 바이너리로 컴파일 한다. 컴파일 후 test 라는 바이너리 파일이 생성된다.

```
gcc -I$UNIXODBC_HOME/include test.c -o test -L$UNIXODBC_HOME/lib -lodbc
```

프로그램을 실행하여 동작을 확인한다.

```
[tibero@cwcordbp01:PS04:/home/tibero/unixodbc/bin]./test
Memory allocation for ODBC Env. (SQLAllocHandle) : Success
ODBC Attributes asignment (SQLSetEnvAttr) : Success
Memory allocation for Connection (SQLAllocHandle) : Success
Query: select to_char(sysdate,'yyyy/mm/dd hh24:mi:ss') as datetime,'ODBC TEST Application' as name from dual;

[ Column Attribute ]
wResultCols : 2
Pos  ColName            ColType ( Precision, Scale)
1. DATETIME         : 12   (19, 0) 8
2. NAME             : 1    (21, 0) 4
```

```
Result : 2025/12/14 15:54:51     ODBC TEST Application

END Program
[tibero@cwcordbp01:PS04:/home/tibero/unixodbc/bin]
```

3 ODBC와 Tibero CLI(Call Level Interface, tbCLI)와의 관계

3.1 개요

tbCLI 는 Call-Level Interface(CLI) 명세와 표준을 따르므로, ODBC 와 연동할 수 있다. tbCLI 는 ODBC 3.51 표준에 맞게 구현되어 있어 데이터베이스 추적 로그 등 ODBC 가 가진 기능을 사용할 수 있다.
tbCLI 는 ODBC(Open Database Connectivity) 및 X/Open Call Level Interface Standard 를 기초로 개발되었다. tbCLI 는 ODBC 2.0 의 Level 2 및 ODBC 3.0 의 Level 1 의 모든 조건, 그리고 ODBC 3.0 Level 2 의 대부분의 조건을 만족한다. 따라서 ODBC 나 CLI 를 이용해 작성된 기존의 애플리케이션 프로그램은 tbCLI 환경으로 쉽게 전환될 수 있다.

3.1.1 tbCLI와 ODBC의 접속 우선순위

tbCLI 에서는 두 가지 방법(tbdsn.tbr 과 ODBC 데이터 원본 관리자의 DSN)을 사용하여 데이터베이스 접속정보를 가져올 수 있다.
Windows 계열의 tbCLI 또는 UNIX 계열(Linux 포함)에서 설치한 ODBC 에서 데이터베이스 접속 정보를 가져올 때에는 우선 ODBC 데이터 원본 관리자를 검색한다.
만약 ODBC 데이터 원본 관리자에 해당 정보가 없다면 tbdsn.tbr 파일을 검색하여 데이터 베이스 접속 정보를 찾는다.
tbsql 또한 tbCLI 를 사용하므로 위에서 언급한 방법으로 데이터베이스 접속정보를 가져온다.

3.2 환경설정

3.2.1 tbdsn.tbr 정의

tbCLI 에서 사용할 Data Source(Alias)가 정의되어 있는 tbdsn.tbr 을 찾기 위해 다음 중 하나의 방법으로 환경설정을 해야 한다.

3.2.1.1 TB_DSN_FILE 환경변수

TB_DSN_FILE 환경변수에 tbdsn.tbr 파일의 절대경로를 등록한다.
ex) TB_DSN_FILE=/home/tibero/TbClntCfg/tbdsn.tbr

3.2.1.2 TB_HOME 환경변수

3.2.1.1 과 같이 TB_DSN_FILE 환경변수가 정의되어 있지 않으면 TB_HOME 환경변수를 찾게 된다.
tbCLI 는 TB_HOME 환경변수 하위의 client/config 디렉토리에서 tbdsn.tbr 파일을 찾게 된다.
따라서 TB_HOME 을 지정한 경우는 TB_HOME 디렉토리 하위에 client/config/tbdsn.tbr 과 같은 디렉토리 구조를 가지고 tbdsn.tbr 파일이 존재하여야 한다.

3.2.2 ODBC 설정에서 tbdsn.tbr 참조

다음과 같이 tbdsn.tbr 파일이 정의가 되어 있고, tbdsn.tbr 파일을 찾을 수 있도록 TB_DSN_FILE 또는 TB_HOME 환경변수가 설정되어 있는 경우 ODBC Application 에서 DB 에 접속하는 과정은 다음과 같다.

tbdsn.tbr 파일

```
PS04=(
   (INSTANCE=(HOST=localhost)
            (PORT=19845)
            (DB_NAME=PS04)
   )
)
```

이 때 ODBC 설정(odbc.ini)은 다음과 같이 SID 를 명시하여 tbdsn.tbr 파일에서 찾을 Data Source(alias 명)를 지정할 수 있다.

```
[tbodbc]
Driver = Tibero 7 ODBC driver
SID = PS04
```

Tibero 서버의 위치를 식별할 수 있는 Server, Port, DB 등과 같은 ODBC 설정 대신 SID 만 설정되어 있다.
이 때 SID 는 tbdsn.tbr 파일에 정의한 Data Source(Alias)와 동일하여야 한다.

3.2.3 ODBC Application 에서 tbdsn.tbr 참조

사용자의 ODBC Application 은 **tbodbc** 라는 Data Source Name 을 ODBC 설정(odbc.ini)에서 확인하고, 해당 Data Source 설정에 특별한 DB 접속정보 대신 SID 만 정의되어 있음을 확인한다. 따라서 Tibero ODBC Driver 는 tbdsn.tbr 에서 해당 SID 와 동일한 Data Source 를 찾아서 DB 접속을 시도하게 된다.
3.3 에서는 몇몇 ODBC 설정에 따라 각각의 경우 어떻게 동작하는지 확인해 보도록 한다.

3.3 ODBC 설정에 따른 동작 확인

3.3.1 Case #1 - SID와 DB접속정보(Server, Port, DB)가 동시에 설정되어 있는 경우

odbc.ini 가 아래와 같이 설정되어 있을 경우

```
[tbodbc]
Driver = Tibero 7 ODBC driver
SID = PS04
SERVER = 192.168.153.129
PORT = 19845
DB = PS04
```

결론 : Server, Port, DB 항목을 통해 Tibero Server 에 접속한다.
SID 로 등록된 값이 tbdsn.tbr 에 등록되어 있고, tbdsn.tbr 을 참조할 수 있도록 TB_DSN_FILE 또는 TB_HOME 등이 정의되어 있다고 할 지라도 이를 이용하지 않고 Server, Port, DB 항목을 통해 Tibero Server 에 접속을 하게 된다.
만약 Server, Port, DB 에 정의된 항목이 잘못된 경우 Tibero Server 접속이 실패된다.
즉, Server, Port, DB 항목이 정의되어 있으면 SID 설정은 무시된다.

3.3.2 Case #2 - SID와 DB접속정보(Server, Port, DB)중 일부가 설정되어 있는 경우

odbc.ini 가 아래와 같이 설정되어 있을 경우

```
[tbodbc]
Driver = Tibero 7 ODBC driver
SID = PS04
SERVER = 192.168.153.129
PORT = 19845
#DB = PS04
```

결론 : 접속이 불가능하다.

2.3.4 에서 설명하였듯 Server, Port, DB 는 반드시 한 세트로 정의되어야 의미가 있다.

SID 로 등록된 값으로 접속을 하기 위해서는 SERVER, PORT, DB 모두 주석처리가 되어야 한다.

4. Tibero ODBC / tbCLI의 디버깅

2.3.3 에서 언급하였 듯 unixODBC 의 Driver Manager Level 에서 Trace 설정을 사용할 수 있다. 하지만 Tibero ODBC Driver Level 에서도 Trace 를 사용할 수 있다. Tibero ODBC Driver 의 디버깅을 위해서는 다음과 같은 환경변수를 정의한 뒤 ODBC Application 을 실행하면 디버깅 로그를 확인할 수 있다.

```
TBCLI_LOG_DIR=<로그를 남길 디렉토리>
TBCLI_LOG_LVL=TRACE
```

로그는 tbcli_년월일시분초_PID.log 형태로 남게 된다.

더 이상 로그를 남기지 않으려면 환경변수를 제거하고 ODBC Application 을 재 실행하면 된다.