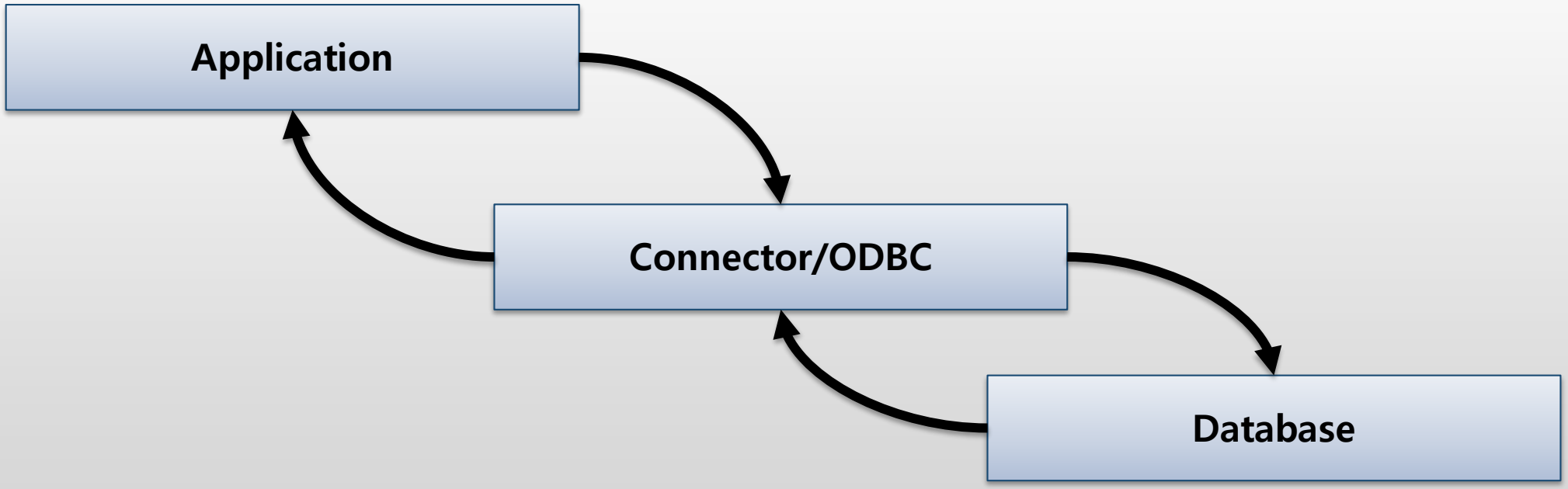
ODBC, JDBC를 이용한 데이터베이스 프로그래밍

ODBC(Open DataBase Connectivity)

- ODBC는 마이크로소프트가 만든 데이터베이스에 접근하기 위한 소프트웨어의 표준 규격임

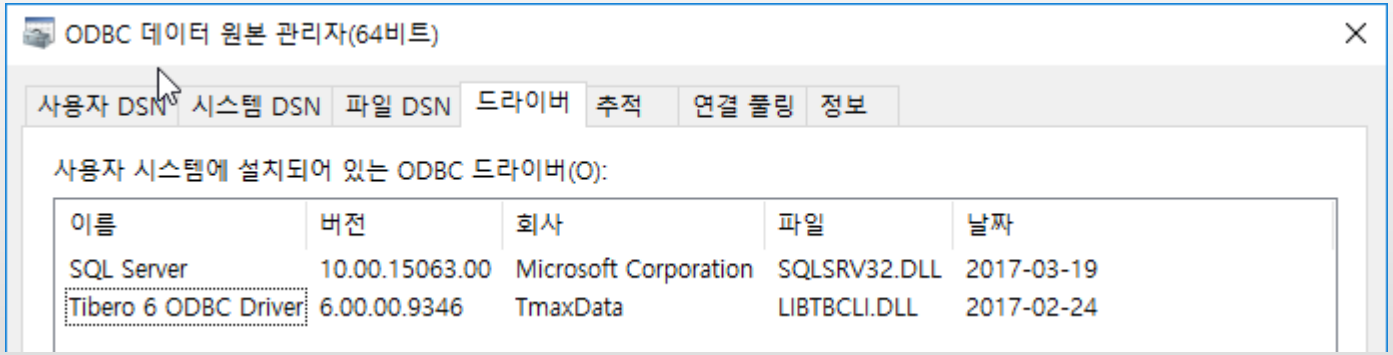
- ODBC는 동일한 API를 사용하여 여러 종류의 데이터베이스(MySQL, Oracle, Tibero,등)에 접근, 관리할 수 있도록 기능을 제공함



Tibero ODBC 드라이버 확인

- **Tibero는 설치시에 ODBC Connector를 함께 설치함**
  - ❖ 실습에서 설치시 ODBC 64비트를 선택하였음

- **제어판-관리 도구-ODBC 데이터 원본 관리자(64비트)-드라이버에서 설치된 ODBC 드라이버를 확인 가능함**

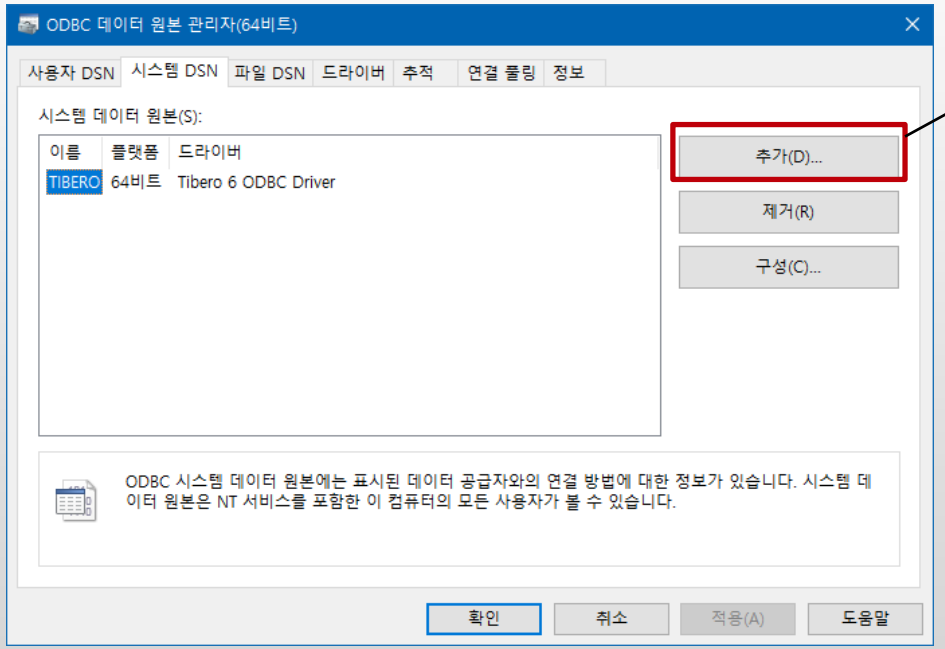


Tibero 6 ODBC Driver 확인

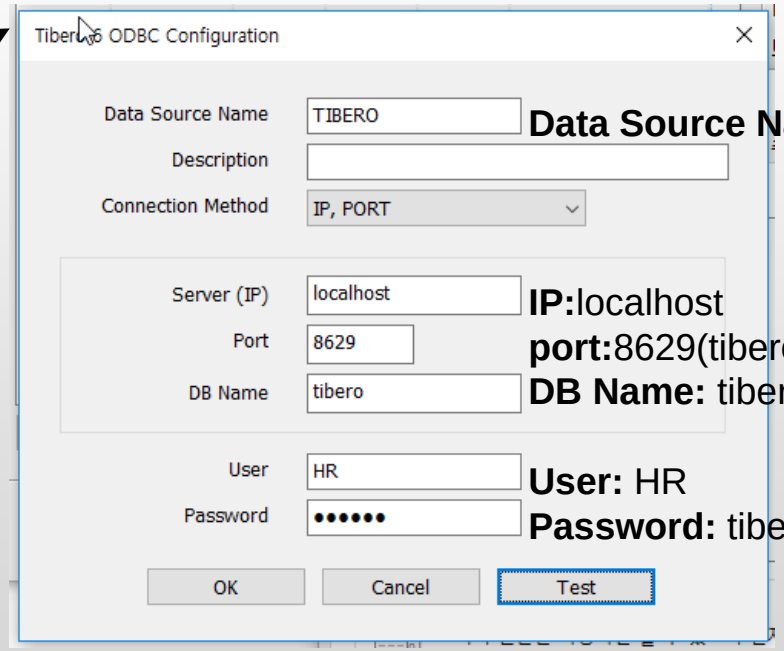
Tibero ODBC Configuration

- ODBC를 이용해 데이터베이스를 접근하기 위해서는 ODBC Configuration에 데이터베이스 접속 정보를 입력해야함

- 제어판-관리 도구-ODBC 데이터 원본(64비트)-시스템 DSN에 데이터베이스 접속 정보를 추가함 (접속 정보는 Tibero Admin에서 접속하는 정보와 동일함)



Tibero 6 ODBC 추가



Data Source Name(ODBC_NAME): TIBERO

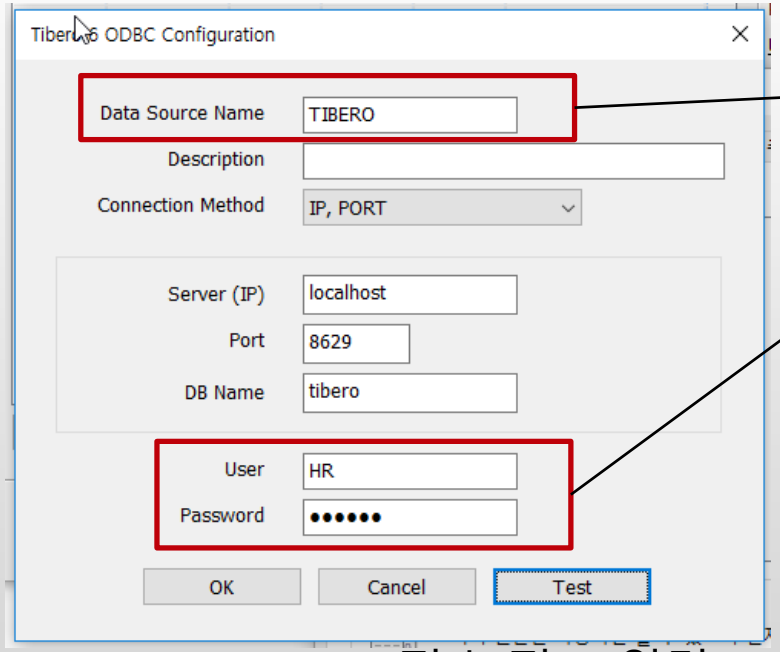
IP:localhost
port:8629(tibero default port)
DB Name: tibero

User: HR
Password: tibero

Tibero 6 접속 정보를 추가

ODBC를 이용한 Tibero 데이터베이스 접속 방법

- **C++에서는 시스템 DSN에 등록한 ODBC정보를 통해 데이터베이스에 접근하기 위한 API 를 제공하고 있음**
- **ODBC접속을 위한 API를 활용하여 DBConnect, DBDisConnect함수를 작성함**
  - ODBC정보는 앞에서 추가한 정보를 의미함
  - 해당 예제: odbc_test_driver.cpp



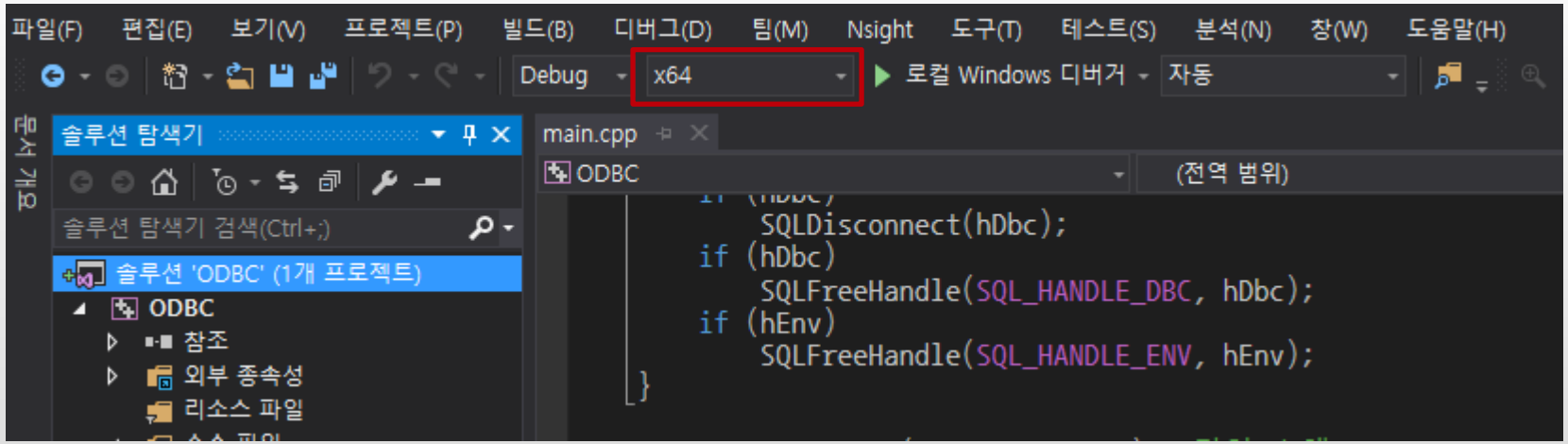
Tibero 6 ODBC 접속 정보 입력

```
#include <Windows.h>
#include <iostream>
#include <sql.h>
#include <sqlext.h>
using namespace std;


//ODBC conn
SQLHENV hEnv;
SQLHDBC hDbc;
SQLHSTMT hStmt;

//접속 정보
SQLCHAR *ODBC_Name = (SQLCHAR*)"TIBERO";
SQLCHAR *ODBC_ID = (SQLCHAR*)"HR";
SQLCHAR *ODBC_PW = (SQLCHAR*)"tibero";

bool DBConnect();
void DBDisConnect();
bool DBExecuteSQL(SQLCHAR* query);

bool DBConnect() //데이터베이스 연결
{
    if (SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hEnv) != SQL_SUCCESS)
        return false;
    if (SQLSetEnvAttr(hEnv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, SQL_IS_INTEGER) != SQL_SUCCESS)
        return false;
    if (SQLAllocHandle(SQL_HANDLE_DBC, hEnv, &hDbc) != SQL_SUCCESS)
        return false;
    if (SQLConnectA(hDbc, ODBC_Name, SQL_NTS, ODBC_ID, SQL_NTS, ODBC_PW, SQL_NTS) != SQL_SUCCESS)
        return false;
    if (SQLAllocHandle(SQL_HANDLE_STMT, hDbc, &hStmt) != SQL_SUCCESS)
        return false;
    return true;
}

void DBDisConnect() //데이터베이스 연결 해제
{
    if (hStmt)
        SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
    if (hDbc)
        SQLDisconnect(hDbc);
    if (hDbc)
        SQLFreeHandle(SQL_HANDLE_DBC, hDbc);
    if (hEnv)
        SQLFreeHandle(SQL_HANDLE_ENV, hEnv);
}

bool DBExecuteSQL(SQLCHAR* query) //질의 수행
{
    if (SQLExecDirectA(hStmt, query, SQL_NTS) != SQL_SUCCESS)
        return false;
    return true;
}
```

ODBC를 이용한 Tibero 데이터베이스 접속 방법

- **실습에서 등록한 ODBC는 64비트 환경에서 동작하므로, 빌드 옵션을 수정하여 어플리케이션을 64비트로 빌드하도록 한다.**
  - 32비트 ODBC환경에서는 x86으로 설정한다.
  - 이 설정을 진행하지 않으면 ODBC접속 정보를 찾지 못함

x86 -> x64



ODBC-질의 방법1

■ 질의 방법1: 최적화 플랜 작성과 질의 수행을 동시에 수행하는 방법

- ❖ SQLExecDirectA함수는 주어진 질의를 수행하는 ODBC기반의 질의 수행 함수임
- ❖ 관리되고 있는 Statement핸들러를 통해 질의를 수행하는 DBExecuteSQL함수를 작성함

```
bool DBExecuteSQL(SQLCHAR* query) //질의 수행
{
    if (SQLExecDirectA(hStmt, query, SQL_NTS) != SQL_SUCCESS)
        return false;
    return true;
}
```

```
DBExecuteSQL((SQLCHAR*)"insert into jobs_copy values('CP_EN','Computer Engineering',20000,60000)");
DBExecuteSQL((SQLCHAR*)"insert into jobs_copy values('AT_PT','Artist Painter',10000,80000)");
```

ODBC-질의 방법2

■ 질의 방법2: 최적화 플랜 작성과 질의 수행을 분리하여 수행하는 방법

- ❖ 데이터베이스는 질의를 최적화된 상태로 수행하기 위한 플랜을 작성함
- ❖ 처리해야하는 값만 바뀌는 질의는 한번 작성된 최적화 플랜을 재사용하기 위해 ODBC는 SQLPrepareA, SQLExecute함수를 제공함
- ❖ SQLPrepareA함수는 플랜을 생성하고, SQLExecute함수는 생성된 플랜을 기반으로 질의를 수행함

```
bool DBExecuteSQL()
{
    if (SQLExecute(hStmt) != SQL_SUCCESS)
        return false;
    return true;
}
bool DBPrepare(SQLCHAR* query_for_statement)
{
    if (SQLPrepareA(hStmt, query_for_statement, SQL_NTS) != SQL_SUCCESS)
        return false;
    return true;
}
```

```
DBPrepare((SQLCHAR*)"select * from employees where salary > 10000");
DBExecuteSQL();
DBExecuteSQL();
```

쿼리플랜 생성

생성된 플랜을 재사용하여 질의 수행

파라미터 바인딩

- 생성된 쿼리 플랜에 처리하기 위해 다른 값 입력하면서 재사용하기 위해서는 파라미터 바인딩 과정을 수행해야함
- 쿼리 플랜 작성시에 쿼리에서 값이 변하는 변수를 ?로 표시함
- SQLBindParameter함수는 C++내의 변수와 쿼리의 ?변수를 바인딩 시킴

```
DBPrepare((SQLCHAR*)"insert into jobs_copy values(?,?,?,?)");

InsertJob((SQLCHAR*)"CP_EN", (SQLCHAR*)"Computer Engineer", 20000, 60000);
InsertJob((SQLCHAR*)"AT_PT", (SQLCHAR*)"Artist Painter", 10000, 80000);
```

```
bool InsertJob(SQLCHAR* job_id, SQLCHAR* job_name, SQLINTEGER min_salary, SQLINTEGER max_salary)
{
    SQLBindParameter(hStmt, 1, SQL_PARAM_INPUT, SQL_CHAR,    SQL_CHAR,    0, 0, job_id, 0, NULL);
    SQLBindParameter(hStmt, 2, SQL_PARAM_INPUT, SQL_CHAR,    SQL_CHAR,    0, 0, job_name, 0, NULL);
    SQLBindParameter(hStmt, 3, SQL_PARAM_INPUT, SQL_INTEGER, SQL_INTEGER, 0, 0, &min_salary, 0, NULL);
    SQLBindParameter(hStmt, 4, SQL_PARAM_INPUT, SQL_INTEGER, SQL_INTEGER, 0, 0, &max_salary, 0, NULL);

    if (DBExecuteSQL())
        return true;
    return false;
}
```

컬럼 번호

C++ 데이터 타입

SQL데이터 타입

바인딩 변수 주소

ODBC-Insert, Delete, Update 실습

■ 1. Insert, Delete, Update질의 실습

❖ Jobs테이블을 복사하여 Jobs_copy테이블을 생성하시오.

❖ 직업 테이블 Jobs_copy 에 새로운 직업 Computer Engineer에 관한 정보를 INSERT하는 코드를 공부하고, 수행하여 결과를 확인하시오.

- JOB_ID('CP_EN'), JOB_TITLE('Computer Engineer'), MIN_SALARY(20000), MAX_SALARY(60000)

```
if (DBExecuteSQL((SQLCHAR*)"insert into jobs values('CP_EN', 'Computer Engineer', 20000, 60000)") == false)
```

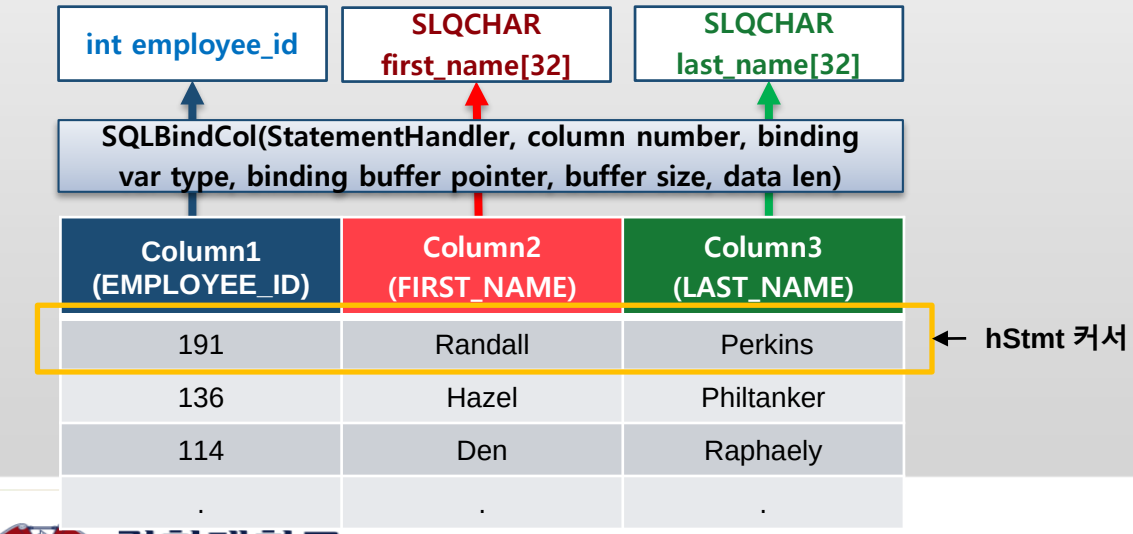
❖ 동일한 방법으로 직업 테이블 Jobs_copy안의 Computer Engineer의 직업을 DELETE하는 코드를 공부하고, 수행하여 결과를 확인하시오.

```
if (DBExecuteSQL((SQLCHAR*)"delete from jobs where job_title='Computer Engineer'") == false)
```

❖ Jobs_copy테이블에서 JOB_TITLE에 'Marketing'글자가 있는 직업의 MAX_SALARY를 500증가시키기 위한 질의를 작성하고, 그 결과를 확인하시오.

ODBC-Select 실습

- **데이터베이스에서 수행된 Select질의 결과를 Statement객체에 바인딩된 변수를 통해 질의 결과를 확인할 수 있음**
  - ❖ SQLBindCol함수를 통해 수행된 질의 결과의 컬럼과 변수를 바인딩할 수 있음
    - SQLBindCol(Statement핸들, 컬럼 번호, C++ 데이터 타입, 결과를 받을 버퍼 포인터(바인딩 변수), 버퍼 크기, 버퍼에 저장된 데이터의 길이)
  - ❖ SQLFetch는 수행한 질의의 결과의 커서를 다음 레코드로 이동시킴
- **주어진 예제는 EMPLOYEES테이블에서 EMPLOYEE_ID, FIRST_NAME, LAST_NAME을 가져오는 질의를 수행하고, C++내의 바인딩된 변수를 통해 결과를 출력하는 예제임**




```
if (DBExecuteSQL((SQLCHAR*)"SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME FROM EMPLOYEES") == false)
{
    cout << "쿼리 에러" << endl;
    return -2;
}
PrintResult();
```

```
void PrintResult()//Select문으로 가져온 데이터 접근하는 방법
{
    int employee_id;
    SQLCHAR first_name[32];
    SQLCHAR last_name[32];

    SQLLEN i_employee_id, i_first_name, i_last_name;

    //select 질의 수행 결과를 저장할 변수를 지정한다.(Column Number는 1부터 시작한다)
    SQLBindCol(hStmt, 1, SQL_INTEGER, &employee_id, sizeof(employee_id), &i_employee_id);
    SQLBindCol(hStmt, 2, SQL_CHAR, first_name, sizeof(first_name), &i_first_name);
    SQLBindCol(hStmt, 3, SQL_CHAR, last_name, sizeof(last_name), &i_last_name);

    //질의 수행결과를 hStmt로 가져온다. 이때, 바인딩된 변수에 값이 할당된다.
    while (SQLFetch(hStmt) != SQL_NO_DATA)
    {
        cout << employee_id << ' ' << first_name << ' ' << last_name << endl;
    }

    //hStmt 커서 해제
    if (hStmt)
        SQLCloseCursor(hStmt);
}
```

ODBC-Select 실습

- **2. 아래 2-4번에서 요구되는 테이블을 식별하고, 각 테이블을 복사한 테이블을 생성하시오.**
  - ❖ 아래 실습은 복사된 테이블에서 진행하시오.

- **3. 각 나라에 속한 사원들의 급여의 총 합을 기준으로 나라 이름을 정렬하여 출력하시오.**

- **4. 급여(SALARY)가 10,000이상인 사원들의 종합 급여가 가장 높은 부서 2곳의 부서 이름(DEPARTMENT_NAME)과 속한 나라의 이름(CONTRY_NAME)을 출력하시오.**

■ 5. ODBC를 이용한 사원 관리 프로그램을 작성하시오

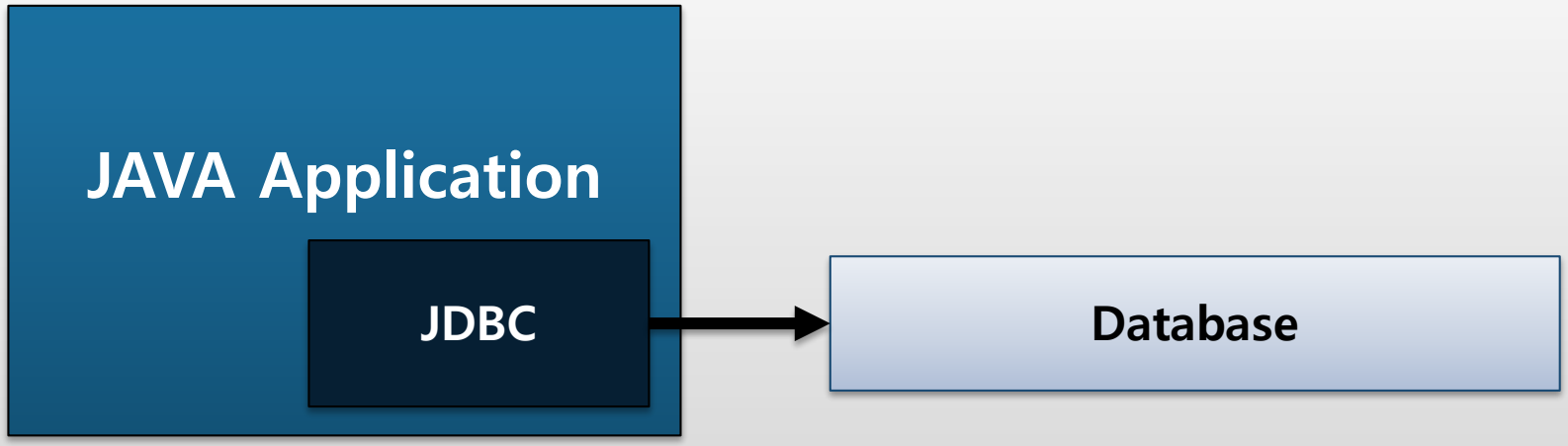
❖ 사원 관리 프로그램은 다음과 같은 기능을 포함한다.

- 사원 조회 기능
  - FIRST_NAME을 이용한 사원 조회 기능
  - LAST_NAME을 이용한 사원 조회 기능
  - 이름(FIRST_NAME 혹은 LAST_NAME)에 특정 문자가 포함된 사원 조회 기능
  - 부서의 이름을 기준으로 부서 소속의 사원 조회 기능
  - 특정 부서에서 가장 적은 급여의 사원 3명과 가장 많은 급여의 3명을 조회하는 기능
- 사원 정보 수정(EMPLOYEE_ID를 입력받고, 해당하는 사원의 정보를 수정)
  - 사원의 SALARY의 변경하는 기능
  - 사원의 부서 이동
  - 사원의 핸드폰 번호 수정

JDBC(Java Database Connectivity)

- JDBC는 자바에서 데이터베이스에 접속할 수 있도록 하는 자바 API임

- 자바 어플리케이션에 내장되어 데이터베이스에 접근을 지원함



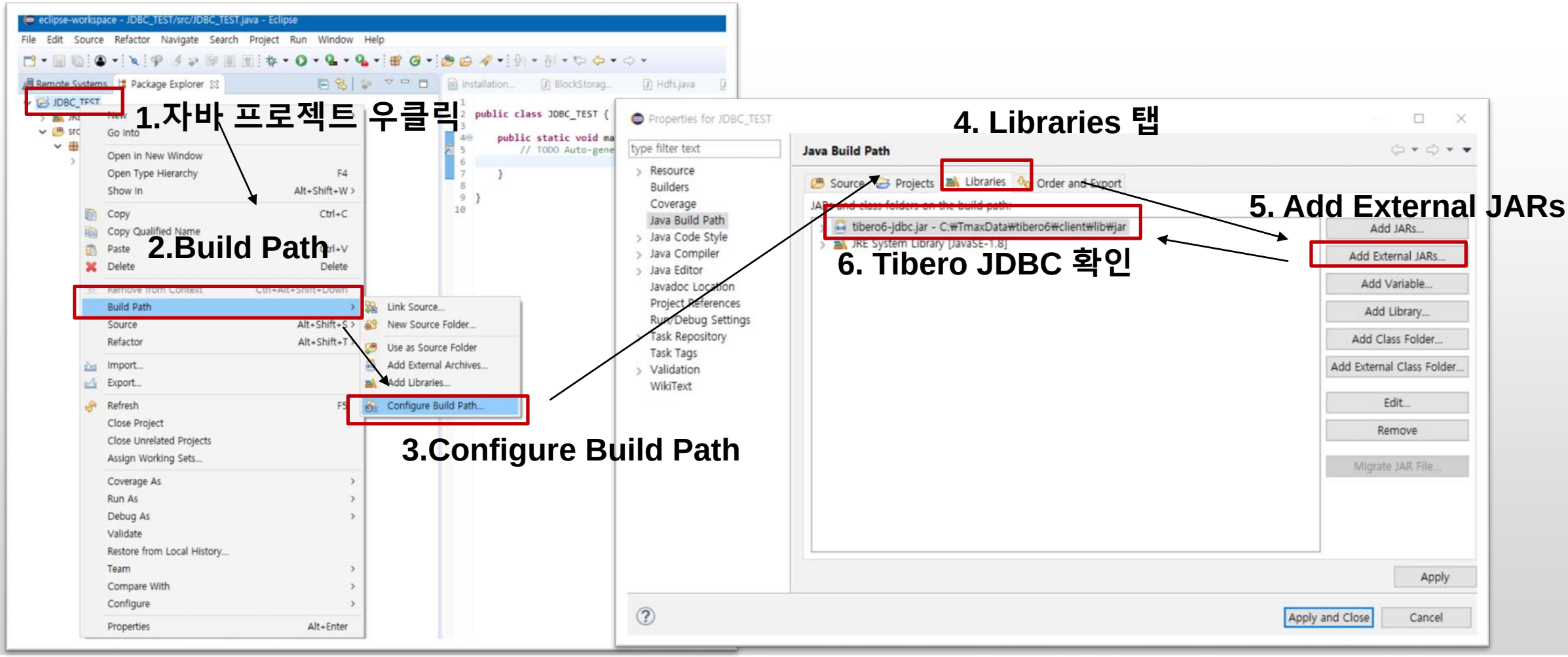
Tibero JDBC 위치

- **JDBC를 이용한 질의를 위해서는 Tibero JDBC를 어플리케이션에 추가하기 위한 작업이 필요함**

- **Tibero는 설치시 JDBC파일을 함께 설치하는데 “$TB_HOME/client/lib/jar” 경로에 위치함**
  - 기본 경로 설치시: C:₩TmaxData₩tibero6₩client₩lib₩jar에 위치함
  - Tibero JDBC 파일: tibero6-jdbc.jar

Tibero JDBC 추가 방법

- Eclipse에서는 Build Path-Configure Build Path-Libraries-Add External JARs에서 Tibero JDBC를 추가할 수 있음



JDBC를 이용한 Tibero 데이터베이스 접속 방법

■ JDBC접속을 위한 정보들을 입력하여 Tibero 데이터베이스에 접속 가능함

```
Connection      conn    = null;    //DB접속
Statement       stmt    = null;    //SQL Statement
ResultSet       rs      = null;    //SQL 실행결과
```

```
//Tibero JDBC Driver
String DB_DRV    = "com.tmax.tibero.jdbc.TbDriver";
//Tibero 연결정보
String DB_IP     = "localhost";      //Tibero IP
String DB_PORT   = "8629";           //Tibero 접속 Port
String DB_SID    = "tibero";         //Tibero SID
String DB_ID     = "HR";          //접속할 유저 ID
String DB_PWD    = "tibero";           //접속할 유저 패스워드
String DB_URL    = "jdbc:tibero:thin:@"+DB_IP+":"+DB_PORT+":"+DB_SID;
```

```
//Tibero 접속
public void connect()
{
    try
    {
        System.out.println("=====================================");
        System.out.println("DB_DRV : " + DB_DRV);
        System.out.println("DB_DRV : " + DB_URL);
        System.out.println("DB_ID  : " + DB_ID);
        System.out.println("DB_PWD : " + DB_PWD);
        System.out.println("-------------------------------------");
        Class.forName(DB_DRV);
        conn = DriverManager.getConnection(DB_URL, DB_ID, DB_PWD);
        System.out.println("Tibero Connect Success");
        System.out.println("=====================================");
        System.out.println("");
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
}
```

```
//Tibero 접속종료
public void disconnect()
{
    try
    {
        if (rs   != null)   rs.close();
        if (stmt != null)   stmt.close();
        if (conn != null)   conn.close();
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
    finally
    {
        if ( rs   != null ) try { rs   .close(); } catch(Exception e) {}
        if ( stmt != null ) try { stmt .close(); } catch(Exception e) {}
        if ( conn != null ) try { conn .close(); } catch(Exception e) {}
    }
}
```

JDBC-Insert, Delete 예제

■ statemen객체의 executeQuery함수를 통해 원하는 질의를 수행할 수 있음

```
//Main 함수
public static void main (String [] args)
{
    JDBC_TEST test = new JDBC_TEST();
    test.connect();
    test.excute();
    test.disconnect();
}
```

1. JDBC객체 생성
2. 데이터베이스 연결
3. 쿼리 수행
4. 데이터베이스 연결 해제

```
Connection      conn    = null;    //DB접속
Statement       stmt    = null;    //SQL Statement
ResultSet       rs      = null;    //SQL 실행결과
```

JDBC관련 변수

```
//SQL 수행
public void excute()
{
    try
    {
        strSQL = "insert into jobs values('CP_EN', 'Computer Engineer',20000,60000)";
        stmt = conn.createStatement();
        rs = stmt.executeQuery(strSQL);
        System.out.println("======================================");
        System.out.println("SQL : " + strSQL);
        System.out.println("--------------------------------------");

    }
    catch (Exception ex)
    {
        ex.printStackTrace();
        disconnect();
    }
}
```

1. Statement 생성
2. 쿼리 수행

JDBC를 이용한 INSERT문

```
//SQL 수행
public void excute()
{
    try
    {
        strSQL = "delete from jobs where job_title='Computer Engineer'";
        stmt = conn.createStatement();
        rs = stmt.executeQuery(strSQL);
        System.out.println("======================================");
        System.out.println("SQL : " + strSQL);
        System.out.println("--------------------------------------");

    }
    catch (Exception ex)
    {
        ex.printStackTrace();
        disconnect();
    }
}
```

JDBC를 이용한 Delete문

JDBC-Select 예제

- **ResultSet객체의 함수(getInt, getString)를 통해 현재 레코드의 Column값을 확인할 수 있음**
  - ❖ next()함수를 통해 다음 레코드 값을 가져옴

```
Connection          conn    = null;    //DB접속
Statement           stmt    = null;    //SQL Statement
ResultSet           rs      = null;    //SQL 실행결과
```

```
//SQL 수행
public void excute()
{
    try
    {
        strSQL = "select employee_id, first_name, last_name from employees";
        stmt = conn.createStatement();
        rs = stmt.executeQuery(strSQL);
        System.out.println("====================================");
        System.out.println("SQL : " + strSQL);
        System.out.println("------------------------------------");
        while ( rs.next() )
        {
            System.out.println("employee_id        : " + rs.getInt(1));
            System.out.println("first_name    : " + rs.getString(2));
            System.out.println("last_name     : " + rs.getString(3));
            System.out.println("====================================");
        }
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
        disconnect();
    }
}
```

```
====================================
employee_id        : 124
first_name    : Kevin
last_name     : Mourgos
====================================
employee_id        : 125
first_name    : Julia
last_name     : Nayer
====================================
employee_id        : 198
first_name    : Donald
last_name     : OConnell
====================================
employee_id        : 153
first_name    : Christopher
last_name     : Olsen
====================================
employee_id        : 132
first_name    : TJ
last_name     : Olson
====================================
employee_id        : 168
first_name    : Lisa
```

JDBC를 이용한 Select문과 질의 결과 출력