****

|  |  |
| --- | --- |
|  | **openML JNI 설명서** |
|  | **제공되는 API에 대한 설명을 기술한다.** |
|  |  |

**목차**

[개요 4](#_Toc338843583)

[JEDB class 5](#_Toc338843584)

[SQL\_DATA\_XXX type 5](#_Toc338843585)

[SQL\_STMT\_XXX query type 5](#_Toc338843586)

[CreateDB1() 6](#_Toc338843587)

[CreateDB2() 7](#_Toc338843588)

[CreateDB3() 7](#_Toc338843589)

[DestroyDB() 8](#_Toc338843590)

[GetConnection() 8](#_Toc338843591)

[Connect() 8](#_Toc338843592)

[Disconnect() 9](#_Toc338843593)

[ReleaseConnection() 9](#_Toc338843594)

[Commit() 9](#_Toc338843595)

[CommitFlush() 10](#_Toc338843596)

[Rollback() 10](#_Toc338843597)

[RollbackFlush() 11](#_Toc338843598)

[QueryCreate() 11](#_Toc338843599)

[QueryDestory() 12](#_Toc338843600)

[QueryBindParam() 12](#_Toc338843601)

[QueryExecute() 13](#_Toc338843602)

[QueryClearRow() 13](#_Toc338843603)

[QueryGetNextRow() 13](#_Toc338843604)

[QueryGetFieldCount() 14](#_Toc338843605)

[QueryGetFieldData() 14](#_Toc338843606)

[QueryIsFieldNull() 15](#_Toc338843607)

[QueryGetFieldName() 15](#_Toc338843608)

[QueryGetFieldType() 16](#_Toc338843609)

[QueryGetFieldLength() 16](#_Toc338843610)

[QueryGetQueryType() 17](#_Toc338843611)

[JNI library 생성 방법 18](#_Toc338843612)

[config.mk 사용 방법 18](#_Toc338843613)

[별도 생성 방법 18](#_Toc338843614)

[openML의 JNI 사용 방법 19](#_Toc338843615)

[openML의 JNI 사용 예제 20](#_Toc338843616)

# 개요

openML에서 지원하는 JNI관련 API를 설명한다.

JNI는 다음과 같이 구성된다.



* JEDB class
  + JEDB class는 JNI를 제공하기 위한 java class
  + JEDB.java에 정의되며 javah를 이용하여 com\_openml\_JEDB.h를 생성
  + com\_openml\_JEDB.c는 com\_openml\_JEDB.h에서 정의된 함수들의 실제 구현 부분
  + ML API를 이용하여 com\_openml\_JEDB.c를 구현

# JEDB class

JEDB class는 JNI를 지원하기 위한 class로 JEDB.java에 정의되어 있다.

Javah를 통해 com\_openml\_JEDB.h를 생성하게 되며 JEDB.h에 정의된 함수 정의를 기준으로 com\_openml\_JEDB.c를 작성하여 libjedb.so를 생성하게 된다.

## SQL\_DATA\_XXX type

* Data type 정의
* openML의 data type 정의가 변경되는 경우 함께 변경됨

|  |
| --- |
| public static final int SQL\_DATA\_NONE = 0;  public static final int SQL\_DATA\_TINYINT = 2;  public static final int SQL\_DATA\_SMALLINT = 3;  public static final int SQL\_DATA\_INT = 4;  public static final int SQL\_DATA\_BIGINT = 6;  public static final int SQL\_DATA\_FLOAT = 7;  public static final int SQL\_DATA\_DOUBLE = 8;  public static final int SQL\_DATA\_DECIMAL = 9;  public static final int SQL\_DATA\_CHAR = 10;  public static final int SQL\_DATA\_NCHAR = 11;  public static final int SQL\_DATA\_VARCHAR = 12;  public static final int SQL\_DATA\_NVARCHAR = 13;  public static final int SQL\_DATA\_BYTE = 14;  public static final int SQL\_DATA\_VARBYTE = 15;  public static final int SQL\_DATA\_TIMESTAMP = 18;  public static final int SQL\_DATA\_DATETIME = 19;  public static final int SQL\_DATA\_DATE = 20;  public static final int SQL\_DATA\_TIME = 21;  public static final int SQL\_DATA\_OID = 23; |

## SQL\_STMT\_XXX query type

* query type을 정의
* openML의 query type 정의가 변경되는 경우 함께 변경됨

|  |
| --- |
| public static final int SQL\_STMT\_NONE = 0;  public static final int SQL\_STMT\_SELECT = 1;  public static final int SQL\_STMT\_INSERT = 2;  public static final int SQL\_STMT\_UPDATE = 3;  public static final int SQL\_STMT\_DELETE = 4;  public static final int SQL\_STMT\_UPSERT = 5;  public static final int SQL\_STMT\_DESCRIBE = 7;  public static final int SQL\_STMT\_CREATE = 8;  public static final int SQL\_STMT\_RENAME = 9;  public static final int SQL\_STMT\_ALTER = 10;  public static final int SQL\_STMT\_TRUNCATE = 11;  public static final int SQL\_STMT\_PRIVILEGE = 12;  public static final int SQL\_STMT\_ADMIN = 13;  public static final int SQL\_STMT\_DROP = 14;  public static final int SQL\_STMT\_COMMIT = 15;  public static final int SQL\_STMT\_ROLLBACK = 16;  public static final int SQL\_STMT\_SET = 17; |

## CreateDB1()

* dbname에 해당하는 DB를 생성하는 메소드

|  |
| --- |
| public native static int CreateDB1 (String dbname); |

* String dbname : DB name
* Return
  + 0 <= : success
  + 0 > : fail, error number

## CreateDB2()

* dbname에 해당하는 DB를 생성하는 메소드
* DB name 및 DB segment 인자로 받음

|  |
| --- |
| public native static int CreateDB2 (String dbname, int dbseg\_num); |

* String dbname : DB name
* int dbseg\_num : DB segment 개수
* Return
  + 0 <= : success
  + 0 > : fail, error number

## CreateDB3()

* dbname에 해당하는 DB를 생성하는 메소드
* DB name 및 DB segment, index segment, temp segment를 인자로 받음

|  |
| --- |
| public native static int CreateDB3 (String dbname, int dbseg\_num,  int idxseg\_num, int tmpseg\_num); |

* String dbname : DB name
* int dbseg\_num : DB segment 개수
* int idxseg\_num : INDEX segment 개수
* int tmpseg\_num : TEMP segment 개수
* Return
  + 0 <= : success
  + 0 > : fail, error number

## DestroyDB()

* dbname에 해당하는 DB를 제거하는 함수
* dbname은 DB 경로가 함께 포함되어야 함

|  |
| --- |
| public native static int DestoryDB (String dbname); |

## GetConnection()

* Connection을 위한 id 값을 얻어오는 메소드
* 최대 connection은 EDB\_MAX\_CONNECTION으로 제한

|  |
| --- |
| public native static int GetConnection (); |

* Return
  + 0 <= : success, connection id
  + 0 > : fail, error number

## Connect()

* dbname에 해당하는 DB에 connect하기 위한 메소드

|  |
| --- |
| public native static int Connect (int connid, String dbname); |

* int connid : connection id
* char \*dbname : 접속할 DB name
* Return
  + 0 <= : success
  + 0 > : fail, error number

## Disconnect()

* dbname에 해당하는 DB를 disconnect하기 위한 메소드

|  |
| --- |
| public native static void Disconnect (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## ReleaseConnection()

* Connecton id에 해당하는 connection을 해제하는 메소드

|  |
| --- |
| public native static void ReleaseConnection (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## Commit()

* 해당 connection을 commit하는 메소드

|  |
| --- |
| public native static void Commit (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## CommitFlush()

* 해당 connection을 commit flush하는 메소드

|  |
| --- |
| public native static void CommitFlush (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## Rollback()

* 해당 connection을 rollback하는 메소드

|  |
| --- |
| public native static void Rollback (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## RollbackFlush()

* 해당 connection을 rollback flush하는 메소드

|  |
| --- |
| public native static void RollbackFlush (int connid); |

* int connid : connection id
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryCreate()

해당 Query를 이용하여 prepare된 statement 생성하는 메소드

|  |
| --- |
| public native static int QueryCreate (int connid, String query, int utf16); |

* int connid : connection id
* char \*query : query string
* int UTF16
  + 1 = : UTF16 문자 처리
  + 1 != : 일반 문자 처리
* Return
  + 0 <= : statement handle
  + 0 > : fail, error number

## QueryDestory()

* 해당 statement handle의 statement를 초기화하는 메소드

|  |
| --- |
| public native static int QueryDestroy (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryBindParam()

* Parameter binding을 위한 data 및 정보 설정하는 메소드

|  |
| --- |
| public native static int QueryBindParam (int connid, int stmt,  int param\_idx, int is\_null, int type, String buffer, int length); |

* int connid : connection id
* int stmt : statement handle
* int param\_idx : paramter index
* int is\_null : null flag
* int type : parameter type
* String buffer : parameter value
* int length : parameter length
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryExecute()

* Statement handle에 해당하는 prepare된 statement를 수행하는 메소드

|  |
| --- |
| public native static int QueryExecute (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryClearRow()

* Statement의 결과를 저장한 result set을 초기화 하는 메소드

|  |
| --- |
| public native static int QueryClearRow (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryGetNextRow()

* Result set의 현재 위치를 다음 result의 위치로 변경하는 메소드.

|  |
| --- |
| public native static int QueryGetNextRow (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryGetFieldCount()

* Field 개수를 반환하는 메소드

|  |
| --- |
| public native static int QueryGetFieldCount (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 <= : success
  + 0 > : fail, error number

## QueryGetFieldData()

* Field index에 해당하는 result value를 반환하는 메소드

|  |
| --- |
| public native static String QueryGetFieldData (int connid, int stmt,  int field\_idx); |

* int connid : connection id
* int stmt : statement handle
* int field\_idx : field index
* Return
  + Null != : result value
  + Null == : error

## QueryIsFieldNull()

* Field index에 해당하는 Result 의 null flag 값을 return하는 메소드

|  |
| --- |
| public native static int QueryIsFieldNull (int connid, int stmt,  int field\_idx); |

* int connid : connection id
* int stmt : statement handle
* int field\_idx : field index
* Return
  + 1 == : field id null
  + 0 == : field is not null
  + 0 > : fail, error number

## QueryGetFieldName()

* Field index에 해당하는 filed name을 return하는 메소드

|  |
| --- |
| public native static String QueryGetFieldName (int connid, int stmt,  int field\_idx); |

* int connid : connection id
* int stmt : statement handle
* int field\_idx : field index
* Return
  + Null != : field name
  + Null == : error

## QueryGetFieldType()

* Field index에 해당하는 field type을 return하는 메소드

|  |
| --- |
| public native static int QueryGetFieldType (int connid, int stmt,  int field\_idx); |

* int connid : connection id
* int stmt : statement handle
* int field\_idx : field index
* Return
  + 0 < : field type
  + 0 > : fail, error number

## QueryGetFieldLength()

* Field index에 해당하는 field length를 return하는 메소드

|  |
| --- |
| public native static int QueryGetFieldLength (int connid, int stmt,  int field\_idx); |

* int connid : connection id
* int stmt : statement handle
* int field\_idx : field index
* Return
  + 0 <= : field length
  + 0 > : fail, error number

## QueryGetQueryType()

* Statement handle에 해당하는 statement의 query type을 return 하는 메소드

|  |
| --- |
| public native static int QueryGetQueryType (int connid, int stmt); |

* int connid : connection id
* int stmt : statement handle
* Return
  + 0 < : query type
  + 0 > : fail, error number

# JNI library 생성 방법

JNI library 생성 방법에 대하여 기술한다.

## config.mk 사용 방법

1. config.mk의 'JNI=0'을 'JNI=1'로 변경
2. 환경변수 JAVA\_HOME에 설치된 java directory를 설정
3. make 수행
   1. lib/libjedb.so 생성
   2. include/JEDB.java 생성

## 별도 생성 방법

1. 필요한 file list
   1. ConvertStinrg.c
   2. ConvertString.h
   3. mdb\_MLapi.c
   4. mdb\_MLapi.h
   5. JEDB.java
   6. com\_openml\_JEDB.c
      1. com\_openml\_JEDB.c는 JEDB.java를 javah로 수행하여 생성되는 com\_openml\_JEDB.h를 바탕으로 작성되어야 함.
2. Makefile 생성

|  |
| --- |
| CFLAGS = -c -g  DB\_HOME = $(MOBILE\_LITE\_HOME)  INCLUDE = -I. -I$(DB\_HOME)/src/include -I$(JAVA\_HOME)/include -I$(JAVA\_HOME)/include/linux  DBLIBS = -L$(DB\_HOME)/lib $(DB\_HOME)/lib/libopenML.a -lm -ldl -lpthread  CC = gcc -m32 -fPIC -D\_GCC  COMPILE = $(CC) $(CFLAGS) $(INCLUDE)  JAVAC = javac  JAVAH = javah  JNI\_LIB = libjedb.so  JNI\_SRC = com\_openml\_JEDB.c ConvertString.c mdb\_MLapi.c  JNI\_OBJ = $(JNI\_SRC:.c=.o)  all : jni\_compile $(JNI\_LIB)  release : jni\_compile $(JNI\_LIB)  jni\_compile:  $(JAVAC) com/openml/JEDB.java  $(JAVAH) -jni com.openml.JEDB  $(JNI\_LIB) : $(JNI\_OBJ)  $(CC) -shared -Wl -o $@ $(JNI\_OBJ) $(DBLIBS)  .c.o:  $(COMPILE) $<  clean :  rm -rf \*.o \*.so com/openml/JEDB.class com\_openml\_JEDB.h |

1. MOBILE\_LITE\_HOME, JAVA\_HOME 환경 변수 설정
2. make 수행
   1. com\_openml\_JEDB.h 생성
   2. libjedb.so 생성

# openML의 JNI 사용 방법

1. JEDB.java를 작성할 Example.java 코드의 폴더에 복사
2. Example.java 작성 (코드 작성시 JEDB.java에 정의된 함수들을 이용)
3. Example.java 작성시 libjedb.so를 링크 하도록 코드 추가
   1. 예

|  |
| --- |
| public Example  {  static  {  System.loadLibrary("jedb");  }  …  } |

1. JEDB.java와 example.java를 javac를 이용하여 함께 compile
2. 'java Example' 수행

# openML의 JNI 사용 예제

JNI 사용예제는 다음을 참고

* src/jni/com/openml/app/Jsql.java
* OS/android/Jsql/src/com/openml/android/Jsql.java