|  |  |
| --- | --- |
|  | **iSQL C API 설명서** |
|  |  |
|  | **iSQL C API에 대해서 설명한다.** |

**주 의 사 항**

|  |
| --- |
| 1. **C API에서 포인터로 전달되는 parameter들은 footprint 최적화 및 성능을 고려하여 API내부에서 NULL check를 하지 않습니다. 사용자가 API를 호출하기전에 직접 NULL check를 해주어야 합니다.** 2. **매뉴얼에 포함되지 않은 API를 사용하여 발생되는 문제는 사용자의 책임으로 고객지원을 받을 수 없습니다.** |

목차

[소개 6](#_Toc343247500)

[자료구조 7](#_Toc343247501)

[**SQL구문 종류의 선언정보** 7](#_Toc343247502)

[**Field 종류의 선언정보** 7](#_Toc343247503)

[**매핑 표** 8](#_Toc343247504)

[**Result type 선언정보** 9](#_Toc343247505)

[**Option flag 선언정보** 9](#_Toc343247506)

[**iSQL 구조체** 9](#_Toc343247507)

[**iSQL\_BIND 구조체** 11](#_Toc343247508)

[**iSQL\_FIELD** 12](#_Toc343247509)

[**iSQL\_FIELD\_OFFSET** 13](#_Toc343247510)

[**iSQL\_LENGTH** 13](#_Toc343247511)

[**iSQL\_RES** 13](#_Toc343247512)

[**iSQL\_ROW** 15](#_Toc343247513)

[**iSQL\_ROWS** 15](#_Toc343247514)

[**iSQL\_ROW\_OFFSET** 15](#_Toc343247515)

[**iSQL\_STMT** 16](#_Toc343247516)

[SQL API 18](#_Toc343247517)

[Database 운영함수들 20](#_Toc343247518)

[**iSQL\_connect** 20](#_Toc343247519)

[**iSQL\_disconnect** 21](#_Toc343247520)

[**iSQL\_options** 21](#_Toc343247521)

[**iSQL\_table\_export** 22](#_Toc343247522)

[**iSQL\_table\_import** 23](#_Toc343247523)

[SQL용 질의함수들 24](#_Toc343247524)

[**iSQL\_affrected\_rows** 24](#_Toc343247525)

[**iSQL\_fetch\_row** 24](#_Toc343247526)

[**iSQL\_field\_count** 25](#_Toc343247527)

[**iSQL\_free\_result** 26](#_Toc343247528)

[**iSQL\_last\_querytype** 26](#_Toc343247529)

[**iSQL\_nquery** 27](#_Toc343247530)

[**iSQL\_num\_rows** 27](#_Toc343247531)

[**iSQL\_query** 28](#_Toc343247532)

[**iSQL\_store\_result** 29](#_Toc343247533)

[**iSQL\_use\_result** 29](#_Toc343247534)

[**iSQL\_get\_plan\_string** 30](#_Toc343247535)

[Statement을 이용한 질의수행 함수들 31](#_Toc343247536)

[**iSQL\_bind\_param** 31](#_Toc343247537)

[**iSQL\_bind\_result** 32](#_Toc343247538)

[**iSQL\_describe** 32](#_Toc343247539)

[**iSQL\_execute** 33](#_Toc343247540)

[**iSQL\_fetch** 34](#_Toc343247541)

[**iSQL\_nprepare** 35](#_Toc343247542)

[**iSQL\_nprepare2** 35](#_Toc343247543)

[**iSQL\_num\_parameter\_fields** 36](#_Toc343247544)

[**iSQL\_num\_result\_fields** 37](#_Toc343247545)

[**iSQL\_param\_count** 37](#_Toc343247546)

[**iSQL\_prepare** 38](#_Toc343247547)

[**iSQL\_prepare2()** 39](#_Toc343247548)

[**iSQL\_prepare\_result** 40](#_Toc343247549)

[**iSQL\_stmt\_affected\_rows** 40](#_Toc343247550)

[**iSQL\_stmt\_close** 41](#_Toc343247551)

[**iSQL\_stmt\_plan\_string** 42](#_Toc343247552)

[**iSQL\_stmt\_querytype** 42](#_Toc343247553)

[**iSQL\_stmt\_store\_result** 43](#_Toc343247554)

[**iSQL\_stmt\_use\_result** 43](#_Toc343247555)

[Rid 관련 함수들 45](#_Toc343247556)

[**iSQL\_affected\_rid** 45](#_Toc343247557)

[**iSQL\_drop\_rid** 45](#_Toc343247558)

[**iSQL\_get\_rid** 46](#_Toc343247559)

[**iSQL\_update\_rid** 47](#_Toc343247560)

[**iSQL\_desc\_update\_rid** 48](#_Toc343247561)

[결과 Set 관련 함수 50](#_Toc343247562)

[**iSQL\_data\_seek** 50](#_Toc343247563)

[**iSQL\_data\_tell** 51](#_Toc343247564)

[**iSQL\_eof** 51](#_Toc343247565)

[**iSQL\_row\_seek** 52](#_Toc343247566)

[**iSQL\_row\_tell** 53](#_Toc343247567)

[Field 관련 함수들 53](#_Toc343247568)

[**iSQL\_fetch\_field** 53](#_Toc343247569)

[**iSQL\_fetch\_fields** 54](#_Toc343247570)

[**iSQL\_fetch\_field\_direct** 55](#_Toc343247571)

[**iSQL\_fetch\_lengths** 55](#_Toc343247572)

[**iSQL\_field\_seek** 56](#_Toc343247573)

[**iSQL\_field\_tell** 57](#_Toc343247574)

[**iSQL\_get\_fieldtype** 58](#_Toc343247575)

[**iSQL\_list\_fields** 58](#_Toc343247576)

[**iSQL\_num\_fields** 59](#_Toc343247577)

[오류 처리 함수들 60](#_Toc343247578)

[**iSQL\_errno** 60](#_Toc343247579)

[**iSQL\_error** 60](#_Toc343247580)

[**iSQL\_stmt\_errno** 61](#_Toc343247581)

[**iSQL\_stmt\_error** 61](#_Toc343247582)

[Transaction 관련 함수들 63](#_Toc343247583)

[**iSQL\_begin\_transaction** 63](#_Toc343247584)

[**iSQL\_commit** 63](#_Toc343247585)

[**iSQL\_commit\_flush** 64](#_Toc343247586)

[**iSQL\_rollback** 65](#_Toc343247587)

[**iSQL\_rollback\_flush** 65](#_Toc343247588)

[예제 프로그램 67](#_Toc343247589)

[**Table / Index생성** 67](#_Toc343247590)

[**Table 삭제** 68](#_Toc343247591)

[**iSQL\_query를 이용한 Insert** 70](#_Toc343247592)

[**iSQL\_query를 이용한 Select** 71](#_Toc343247593)

[**iSQL\_query를 이용한 Update** 73](#_Toc343247594)

[**iSQL\_query를 이용한 Delete** 75](#_Toc343247595)

[**iSQL\_prepare를 이용한 Insert** 76](#_Toc343247596)

[**iSQL\_prepare를 이용한 Select** 79](#_Toc343247597)

[**결과 Set을 이용하여 재 검색** 81](#_Toc343247598)

**그림 목차**

[[그림 1] Table/Index생성 67](#_Toc343097562)

[[그림 2] Table/Index 삭제 68](#_Toc343097563)

[[그림 3] iSQL\_query를 이용한 Insert 70](#_Toc343097564)

[[그림 4] iSQL\_query를 이용한 Select 72](#_Toc343097565)

[[그림 5] iSQL\_query를 이용한 update 74](#_Toc343097566)

[[그림 6] iSQL\_query를 이용한 delete 75](#_Toc343097567)

[[그림 7] iSQL\_prepare를 이용한 Insert 77](#_Toc343097568)

[[그림 8] iSQL\_prepare를 이용한 Select 79](#_Toc343097569)

[[그림 9] 결과 Set을 이용하여 재 검색 82](#_Toc343097570)

**표 목차**

[[표 1] C type 매핑표 9](#_Toc343097571)

[[표 2] iSQL 멤버변수 설명 11](#_Toc343097572)

[[표 3] iSQL\_BIND Input Bind로 사용되는 경우 11](#_Toc343097573)

[[표 4] iSQL\_BIND Resurt Bind로 사용되는 경우 12](#_Toc343097574)

[[표 5] iSQL\_Field 구조체 13](#_Toc343097575)

[[표 6] iSQL\_RES 변수 설명 15](#_Toc343097576)

[[표 7] iSQL\_STMT 변수 설명 17](#_Toc343097577)

|  |
| --- |
| 소개 |

|  |
| --- |
| OpenML에서 제공하는 SQL C API의 자료구조, 함수들과 각 함수들의 예제에 대하여 설명한다.   * 선언정보 및 자료구조 * DB 운영 함수들 * SQL용 질의 함수들 * Statement을 이용한 질의 실행 함수들 * RID 관련 함수들 * 결과 set 관련 함수들 * Field 관련 함수 * 오류 처리 함수 * 트랜잭션 함수 * 예제 및 설명 |
|  |

|  |
| --- |
| 자료구조 |

|  |
| --- |
| **SQL구문 종류의 선언정보** |
|  |
| |  | | --- | | typedef enum {  SQL\_STMT\_NONE = 0,  SQL\_STMT\_SELECT,  SQL\_STMT\_INSERT,  SQL\_STMT\_UPDATE,  SQL\_STMT\_DELETE,  SQL\_STMT\_UPSERT,  SQL\_STMT\_DESCRIBE,  SQL\_STMT\_CREATE,  SQL\_STMT\_RENAME,  SQL\_STMT\_ALTER,  SQL\_STMT\_TRUNCATE,  SQL\_STMT\_ADMIN,  SQL\_STMT\_DROP,  SQL\_STMT\_COMMIT,  SQL\_STMT\_ROLLBACK,  SQL\_STMT\_SET,  SQL\_STMT\_DUMMY = MDB\_INT\_MAX  } isql\_stmt\_type; | |

|  |
| --- |
| **Field 종류의 선언정보** |

|  |  |
| --- | --- |
| |  | | --- | | typedef enum {  SQL\_DATA\_NONE= 0,  SQL\_DATA\_TINYINT= DT\_TINYINT,  SQL\_DATA\_SMALLINT= DT\_SMALLINT,  SQL\_DATA\_INT= DT\_INTEGER,  SQL\_DATA\_BIGINT= DT\_BIGINT,  SQL\_DATA\_FLOAT= DT\_FLOAT,  SQL\_DATA\_DOUBLE= DT\_DOUBLE,  SQL\_DATA\_DECIMAL= DT\_DECIMAL,  SQL\_DATA\_CHAR = DT\_CHAR,  SQL\_DATA\_VARCHAR = DT\_VARCHAR,  SQL\_DATA\_NCHAR= DT\_NCHAR,  SQL\_DATA\_NVARCHAR= DT\_NVARCHAR,  SQL\_DATA\_BYTE = DT\_BYTE,  SQL\_DATA\_VARBYTE = DT\_VARBYTE,  SQL\_DATA\_TIMESTAMP= DT\_TIMESTAMP,  SQL\_DATA\_DATETIME = DT\_DATETIME,  SQL\_DATA\_DATE = DT\_DATE,  SQL\_DATA\_TIME = DT\_TIME ,  SQL\_DATA\_OID= DT\_OID,  SQL\_DATA\_DUMMY = MDB\_INT\_MAX  } isql\_data\_type; | |

|  |
| --- |
| **매핑 표** |

|  |
| --- |
| 아래의 표는 DB Data Type을 C type으로 바인딩할 경우의 매핑 표이다. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type매크로명** | **C Type** | | **비고** |
| **SQL\_DATA\_TINYINT** | signed char | | 1 byte |
| **SQL\_DATA\_SMALLINT** | signed short int | |  |
| **SQL\_DATA\_INT** | signed int | |  |
| **SQL\_DATA\_BIGINT** | Linux | Singed long long int |  |
| Window 2K, XP | signed LONGLONG |
| Window CE | signed \_\_int64 |
| **SQL\_DATA\_FLOAT** | float | |  |
| **SQL\_DATA\_DOUBLE** | double | |  |
| **SQL\_DATA\_DECIMAL** | double | |  |
| **SQL\_DATA\_CHAR** | unsigned char \* | | 데이터 크기 |
| **SQL\_DATA\_VARCHAR** | unsigned char \* | | 데이터 크기 + 1 |
| **SQL\_DATA\_NCHAR** | wchar \* | | 데이터 크기 |
| **SQL\_DATA\_NVARCHAR** | wchar \* | | 데이터 크기 + 1 |
| **SQL\_DATA\_BYTE** | unsigned char \* | |  |
| **SQL\_DATA\_VARBYTE** | unsigned char \* | |  |
| **SQL\_DATA\_DATETIME**  **SQL\_DATA\_TIMESTAMP**  **SQL\_DATA\_DATE**  **SQL\_DATA\_TIME** | typedef struct {  unsigned short int year;  unsigned short int month;  unsigned short int day;  unsigned short int hour;  unsigned short int minute;  unsigned short int second;  unsigned short int fraction;  } ISQL\_TIME | |  |
| **SQL\_OID** | unsigned long | |  |

[표 ] C type 매핑표

|  |
| --- |
| **Result type 선언정보** |

|  |
| --- |
| typedef enum {  SQL\_RES\_BINARY = 0,  SQL\_RES\_STRING,  SQL\_RES\_DUMMY = MDB\_INT\_MAX  } isql\_res\_type; |

|  |
| --- |
| **Option flag 선언정보** |

|  |
| --- |
| typedef enum {  OPT\_AUTOCOMMIT = 0x0001,  OPT\_TIME = 0x0002,  OPT\_HEADING = 0x0004,  OPT\_FEEDBACK = 0x0008,  OPT\_RECONNECT = 0x0010,  OPT\_PLAN\_ON = 0x0020,  OPT\_PLAN\_OFF = 0x0040,  OPT\_PLAN\_ONLY = 0x0080,  OPT\_PLAN\_DUMMY = MDB\_INT\_MAX  } isql\_option; |
|  |

|  |
| --- |
| **iSQL 구조체** |

|  |
| --- |
| typedef struct tag\_isql {  int status;  char dbname[256];  unsigned short int port;  unsigned short int free\_me;  int handle;  int server\_handle;  int flags;  iSQL\_FIELD \*fields;  unsigned int field\_count;  unsigned int affected\_rows;  unsigned long affected\_rid;  int last\_errno;  char \*last\_error;  isql\_stmt\_type last\_querytype;  iSQL\_DATA \*data;  int clientType;  } iSQL; |

|  |  |
| --- | --- |
| **변수명** | **설 명** |
| **status** | isql사용 상태표시. isql.h에 선언. iSQL\_STAT\_XXXX… |
| **dbname** | 연결할 DB명.  MOBILE\_LITE\_CONFIG 환경변수를 사용하지 않는 경우에는 전체경로를 지정해야 한다. |
| **port** | 연결할 DB의 Port. |
| **free\_me** | iSQL\_init을 NULL로 호출한 경우 내부적으로 iSQL구조체에 대한 메모리를 할당하여 반환하기 때문에 free\_me 값을 확인하여 iSQL\_disconnect시에 free할 수 있도록 한다.  free\_me = 1 : 내부적으로 iSQL 구조체 메모리할당 iSQL\_init NULL 호출  free\_me = 0 : 내부적으로 iSQL 구조체 메모리할당하지 않음 |
| **server\_handle** | OpenML에서는 사용되지 않고, MobileDBS에서만 사용. |
| **handle** | DB연결 식별자.  DB API사용 시 연결 식별자로 사용한다. |
| **flags** | 설정된 options의 조합. (isql\_option 참고) |
| **fields** | Field정보 결과 값. (schema 정보, selet list 정보) |
| **field\_count** | 최근에 수행된 질의문의 컬럼 수. |
| **affected\_rows** | 최근에 수행된 질의문의 레코드 개수. |
| **affected\_rid** | insert된 레코드의 rid. |
| **last\_errno** | 마지막으로 발생된 오류번호. |
| **last\_querytype** | 최근에 수행된 질의의 유형 |
| **data** | 결과 셋의 저장 구조체. |
| **clientType** | OpenML에서는 사용되지 않음. |

[표 ] iSQL 멤버변수 설명

|  |
| --- |
| **iSQL\_BIND 구조체** |

|  |
| --- |
| typedef struct {  unsigned long int \*length;  int \*is\_null;  isql\_data\_type buffer\_type;  unsigned long int buffer\_length;  char \*buffer;  } iSQL\_BIND;  Input bind parameter로 사용될 때와 result bind parameter로 사용될 때의 의미가 다름을 인지해야 한다. |

|  |  |
| --- | --- |
| **변수명** | **설 명** |
| **length** | 저장할 데이터의 바이트 단위 길이.  이 값은 일반적으로 내부에서 자동으로 설정되지만, BYTE 또는 VARBYTE인 경우에는 꼭 사용자가 설정해야 한다. |
| **is\_null** | 데이터의 NULL여부.  데이터가 NULL인 경우 1이고 NULL이 아닌 경우 0으로 설정한다 |
| **buffer\_type** | 데이터 type  설정하지 않은 경우에, 해당 Table의 Schema를 참고하여 설정한다.  (주의: 조건절에 1+? 과 같이 사용하는 경우 Parameter는 number type을 선언해야하며, 문자열 type을 선언하게 되면 에러가 발생할 수 있다.) |
|
|
| **buffer\_length** | buffer의 바이트 단위 길이. |
| **buffer** | 저장 데이터 |

[표 ] iSQL\_BIND Input Bind로 사용되는 경우

|  |  |
| --- | --- |
| **변수명** | **설 명** |
| **length** | 읽어온 데이터의 바이트 단위 길이. |
| **is\_null** | 읽어온 데이터의 NULL여부  데이터가 NULL인 경우 1이고 NULL이 아닌 경우 0이다. |
| **buffer\_type** | 읽어온 데이터 type |
| **buffer\_length** | buffer의 바이트 단위 길이. |
| **buffer** | 읽어온 데이터 |

[표 ] iSQL\_BIND Resurt Bind로 사용되는 경우

|  |
| --- |
| **iSQL\_FIELD** |

|  |
| --- |
| typedef struct {  char name[FIELD\_NAME\_LENG];  char base\_name[FIELD\_NAME\_LENG];  char table[REL\_NAME\_LENG];  char def[MAX\_FIELD\_VALUE\_SIZE];  int type;  unsigned int length;  unsigned int max\_length;  unsigned int flags;  unsigned int decimals;  unsigned int buffer\_length;  } iSQL\_FIELD; |

|  |  |
| --- | --- |
| **변수명** | **설 명** |
| **name** | field명으로 select시 alias가 있을 경우 alias명을 나타내고, 없을 경우 base\_name과 같다. |
| **base\_name** | Schema 정의시에 지정된 field명  (수식이 포함되어 있는 경우에는 빈 문자열이 설정된다.) |
| **table** | Field가 속해 있는 Table명. |
| **def** | Field의 Default 값 |
| **type** | Field의 Type 형태(isql\_data\_type참조) |
| **length** | FieldType이 가질 수 있는 최대 문자열 길이.   1. tinyint인 경우 표현 가능한 범위는 -128~127이므로 length는 4로 설정한다. 2. decimal(5,2)인 경우 표현 가능 범위는 5자리이고 소수점과 ‘-‘표시를 포함해 length는 7로 설정한다. |
| **max\_length** | length와 동일함. |
| **flags** | Field에 설정된 다음과 같은 flag 값.   1. iSQL\_FIELD\_FLAG\_PRI : primary key 2. iSQL\_FIELD\_FLAG\_NULL : NULL 허용필드 3. iSQL\_FIELD\_FLAG\_AUTO : autoincrement 필드 |
| **decimals** | DECIMAL인 경우 해당 Column의 소수점 이하 자릿수 (scale값)  이 값은 DECIMAL인 경우에만 유효하며, 다른 type에 대해서는 모두 0으로 설정된다  예1) DECIMAL(5,2)인 경우, 이 값은 2로 설정 |
| **buffer\_length** | iSQL\_prepare/iSQL\_execute API사용시에 실제 데이터의 길이정보를 byte단위로 표현한다. |

[표 ] iSQL\_Field 구조체

|  |
| --- |
| **iSQL\_FIELD\_OFFSET** |

|  |
| --- |
| typedef unsigned long int iSQL\_FIELD\_OFFSET;  iSQL\_FIELD\_OFFSET은 질의 결과에서 한 Record의 Column 위치의 pointer를 가지고 있다.  iSQL\_fetch\_row()를 수행한 후에 사용된다. |

|  |
| --- |
| **iSQL\_LENGTH** |

|  |
| --- |
| typedef unsigned long int \*iSQL\_LENGTH;  iSQL\_LENGTH는 unsigned long int의 포인터로 iSQL\_ROW를 사용하는 경우, Column의 문자열 길이를 나타낸다. (iSQL\_ROW값이 “34”라면, iSQL\_LENGTH는 2이다.) |

|  |
| --- |
| **iSQL\_RES** |

|  |
| --- |
| typedef struct {  unsigned char dont\_touch;  struct tag\_isql \*isql;  unsigned long int row\_count;  unsigned int field\_count;  unsigned int current\_field;  iSQL\_FIELD \*fields;  iSQL\_DATA \*data;  iSQL\_ROWS \*data\_cursor;  long int data\_offset;  iSQL\_LENGTH lengths;  iSQL\_ROW row;  iSQL\_ROW current\_row;  OID current\_rid;  int rid\_included;  int eof;  int is\_partial\_result;  DB\_BOOL bFirst;  } iSQL\_RES; |

|  |  |
| --- | --- |
| 변수명 | 설 명 |
| dont\_touch | 내부 사용변수 |
| isql | iSQL api에서 사용하는 연결 식별자 |
| row\_count | 결과 set의 Row Count |
| field\_count | 결과 set의 Field Count |
| current\_field | 엔진 내부에서 사용 |
| fields | 결과 Set의 Field 정보 |
| data | 결과 Set |
| data\_cursor | 내부 사용변수 |
| data\_offset | 내부 사용변수 |
| lengths | 내부 사용변수 |
| row | 내부 사용변수 |
| current\_row | 내부 사용변수 |
| current\_rid | 현재 row의 RID 값 |
| rid\_included | 사용 안됨 |
| eof | 내부적으로 사용하는 변수로 1이면 레코드의 끝을 나타내며, 0이면 레코드가 남아있음. |
| is\_partial\_result | result set 생성시 use\_result / store\_result 인지 구분하는 변수  0 = store\_result / 1 = use\_result |
| is\_seek | 사용 안됨 |
| stmt | iSQL\_RES와 연관된 statement 구조체 |
| bFirst | 사용 안됨 |

[표 ] iSQL\_RES 변수 설명

|  |
| --- |
| **iSQL\_ROW** |

|  |
| --- |
| typedef char \*\*iSQL\_ROW;  iSQL\_ROW는 이중 문자열 포인터로 질의 결과에서, 한 Record의 Column 포인터를 가지고 있다. (iSQL\_fetch\_row()에서만 사용된다.) |

|  |
| --- |
| **iSQL\_ROWS** |

|  |
| --- |
| typedef struct tag\_isql\_rows {  OID rid;  int offset;  iSQL\_ROW data;  struct tag\_isql\_rows \*next;  } iSQL\_ROWS;  iSQL\_ROW는 이중 문자열 포인터로 질의 결과에서 한 Record의 Column 포인터를 가지고 있다. (iSQL\_fetch\_row()에서만 사용된다.) |

|  |
| --- |
| **iSQL\_ROW\_OFFSET** |

|  |
| --- |
| typedef iSQL\_ROWS \*iSQL\_ROW\_OFFSET;  iSQL\_ROW는 이중 문자열 포인터로 질의 결과에서 한 Record의 Column 포인터를 가지고 있다. (iSQL\_fetch\_row()에서만 사용된다.) |

|  |
| --- |
| **iSQL\_STMT** |

|  |
| --- |
| typedef struct {  unsigned int stmt\_id;  isql\_stmt\_type querytype;  isql\_res\_type resulttype;  struct tag\_isql \*isql;  unsigned int affected\_rows;  unsigned long affected\_rid;  unsigned int field\_count;  unsigned int param\_count;  iSQL\_RES \*res;  iSQL\_BIND \*pbind;  iSQL\_BIND \*rbind;  iSQL\_FIELD \*fields;  iSQL\_FIELD \*pfields;  DB\_BOOL \*is\_set\_bind\_param;  int query\_timeout;  char \*bind\_space;  int \*indicator\_space;  unsigned long int \*length\_space;  } iSQL\_STMT; |

|  |  |
| --- | --- |
| 변수명 | 설 명 |
| stmt\_id | stmt의 id |
| querytype | 쿼리 유형 |
| resulttype | 결과 유형 |
| isql | iSQL api에서 사용하는 연결 식별자 |
| affected\_rows | 실행된 결과 수 |
| affected\_rid | insert 수행 시 insert에 성공한 rid를 저장 |
| field\_count | select 문장시 결과 set의 field count |
| param\_count | 질의문의 ‘?’ Parameter 개수 |
| res | 결과set |
| pbind | Parameter bind 정보 |
| rbind | Result bind 정보 |
| fields | Fields정보 |
| pfields | Parameter bind된 Field 정보 |
| is\_set\_bind\_param | 내부 사용 변수 |
| query\_timeout | query 수행 제한 시간. 설정된 시간 내에 끝나지 않으면 query 수행을 중단 시킴 |
| bind\_space | 내부 사용변수 |
| indicator\_space | 내부 사용변수 |
| length\_space | 내부 사용변수 |

[표 ] iSQL\_STMT 변수 설명

|  |
| --- |
| SQL API |

|  |  |
| --- | --- |
| **함수 구분** | **함 수** |
| **Database운영함수** | iSQL\_connect() |
| iSQL\_disconnect() |
| iSQL\_options() |
| **SQL용 질의함수들** | iSQL\_affected\_rows() |
| iSQL\_fetch\_row() |
| iSQL\_field\_count() |
| iSQL\_free\_result() |
| iSQL\_last\_querytype() |
| iSQL\_nquery() |
| iSQL\_num\_rows() |
| iSQL\_query() |
| iSQL\_store\_result() |
| iSQL\_use\_result() |
| iSQL\_get\_plan\_string() |
| **Statement을 이용한**  **질의수행 함수들** | iSQL\_bind\_param() |
| iSQL\_bind\_result() |
| iSQL\_describe() |
| iSQL\_execute() |
| iSQL\_fetch() |
| iSQL\_nprepare() |
| iSQL\_nprepare2() |
| iSQL\_num\_parameter\_fields() |
| iSQL\_num\_result\_fields() |
| iSQL\_param\_count() |
| iSQL\_prepare() |
| iSQL\_prepare2() |
| iSQL\_prepare\_result() |
| iSQL\_stmt\_affected\_rows() |
| iSQL\_stmt\_close() |
| iSQL\_stmt\_plan\_string() |
| iSQL\_stmt\_querytype() |
| iSQL\_stmt\_store\_result() |
| iSQL\_stmt\_use\_result() |
| **Rid관련 함수들** | iSQL\_affected\_rid() |
| iSQL\_drop\_rid() |
| iSQL\_get\_rid() |
| iSQL\_update\_rid() |
| iSQL\_desc\_update\_rid() |
| **결과Set관련 함수들** | iSQL\_data\_seek() |
| iSQL\_data\_tell() |
| iSQL\_eof() |
| iSQL\_row\_seek() |
| iSQL\_row\_tell() |
| **Field관련 함수들** | iSQL\_fetch\_field() |
| iSQL\_fetch\_fields() |
| iSQL\_fetch\_field\_direct() |
| iSQL\_fetch\_lengths() |
| iSQL\_field\_seek() |
| iSQL\_field\_tell() |
| iSQL\_get\_fieldtype() |
| iSQL\_list\_fields() |
| iSQL\_num\_fields() |
| **오류 처리 함수들** | iSQL\_errno() |
| iSQL\_error() |
| iSQL\_stmt\_errno() |
| iSQL\_stmt\_error() |
| **Transaction관련 함수들** | iSQL\_begin\_transaction() |
| iSQL\_commit() |
| iSQL\_commit\_flush() |
| iSQL\_rollback() |
| iSQL\_rollback\_flush() |

|  |
| --- |
| Database 운영함수들 |

|  |
| --- |
| **iSQL\_connect** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_connect()** | | | | |
|  | void \*iSQL\_connect (iSQL \*isql, char \*dbhost, char \*dbname, char \*dbuser, char \*dbpasswd); | | | |
|  | **Arguments** | | | |
|  |  | isql | OUT | DB연결 정보를 저장하는 구조체 |
|  |  | dbhost | IN | DB에 연결하기 위한 IP address |
|  |  | dbname | IN | 연결에 사용할 DB명 (경로 + DB명) |
|  |  | dbuser | IN | 사용자 ID |
|  |  | dbpasswd | IN | 사용자 Password |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(isql 핸들값) | |
|  | **Description** | | | |
|  | OpenML를 사용하기 위해 연결을 시작하는 함수이다.  (다른 함수를 호출하기 전에 반드시 호출되어야 하는 함수)   * dbhost: “127.0.0.1” or NULL (openML에서는 의미 없음) * dbname: database의 전체 경로명을 지정 (database명을 사용할 경우에는 내부적으로 지정된 기본경로명을 사용한다.) * dbuser: 임의 사용 (내부적으로 무시) * dbpasswd: 임의 사용 (내부적으로 무시) | | | |
|  | **Sample** | | | |
|  | iSQL isql;  if (**iSQL\_connect**(&isql, “127.0.0.1”, “/db/mldb”, “mmdb”, “mmdb”) == NULL) {  printf (“연결 오류 \n”);  } else {  printf(“연결 성공 \n”);  } | | | |

|  |
| --- |
| **iSQL\_disconnect** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iSQL\_disconnect() | | | | |
|  | void iSQL\_disconnect (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | DB명에 해당되는 핸들값 |
|  | **Description** | | | |
|  | OpenML DBMS와 연결을 종료하는 함수이다.  함수의 호출없이 응용프로그램이 종료된 경우 재연결을 수행할 때 데이터 복구 작업이 호출되어 연결 시간이 지연될 수 있다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;    if (iSQL\_connect(&isql, “127.0.0.1”, “/db/mldb”, “mmdb”, “mmdb”) == NULL) {  /\* 오류 처리 \*/  }  **iSQL\_disconnect**(&isql)**;** | | | |

|  |
| --- |
| **iSQL\_options** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iSQL\_options() | | | | |
|  | int iSQL\_options(iSQL \*isql, const isql\_option opt, const void \*optval, int optlen); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | opt | IN | 옵션 사항 |
|  |  |  |  | OPT\_AUTO\_COMMIT : auto-commit |
|  |  |  |  | OPT\_PLAN\_ON : execute시 query plan 설정 |
|  |  |  |  | OPT\_PLAN\_OFF : execute시 query plan 해제 |
|  |  |  |  | OPT\_PLAN\_ONLY : execute시 질의문은 수행하지 않고 query plan만 추출 |
|  |  | optval | IN | opt에 해당되는 설정값  0 : 옵션 적용  1 : 옵션 적용 해제 |
|  |  | optlen | IN | optval의 크기값  sizeof(int) |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | OpenML DBMS에게 선택 사항을 설정하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  int onoff = 1;  …..  if (**iSQL\_options**(&isql, OPT\_AUTOCOMMIT, (const void \*)&onoff, sizeof(int)) != 0) {  printf (“설정 오류 \n”);  } else {  printf(“auto commit 설정 성공 \n”);  }  …. | | | |

|  |
| --- |
| **iSQL\_table\_export** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_table\_export()** | | | | |
|  | int iSQL\_table\_export (iSQL \*isql, char \*filename, char \*tablename, int f\_append); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | filename | IN | DB 데이터를 저장하고자하는 파일명 |
|  |  | tablename | IN | 저장할 테이블명 |
|  |  | f\_append | IN | 기존 파일에 append할 것인지 새롭게 생성할 것인지 설정 |
|  |  |  |  | 0 : 새롭게 생성  1 : 기존 파일에 append |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 table명의 데이터를 DB로부터 추출해서 파일로 저장하는 함수이다.  저장된 파일은 byte또는 varbyte를 저장하기위해 ascii file이 아닌 binary file로 저장되며, 어떤 table이 추출되었는지에 대한 정보를 저장하고 있다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  …..  if (**iSQL\_table\_export**(&isql, “mldb\_sample.dat”, “mldb\_sample”, 0) != 0) {  printf (“export 오류 \n”);  } else {  printf(“export 성공 \n”);  }  …. | | | |

|  |
| --- |
| **iSQL\_table\_import** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_table\_import()** | | | | |
|  | int iSQL\_table\_import (iSQL \*isql, char \*filename); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | filename | IN | DB에 데이터를 import하기 위한 원시데이터가 저장되어 있는 파일명 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_table\_export()에 의해 생성된 원시 데이터 파일을 DB로 import하는 함수이다.  원시데이터 파일안에는 어떤 table들이 추출되어 있는 정보가 있어, 명시적으로 import될 table을 지정할 필요가 없다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  …..  if (**iSQL\_table\_import**(&isql, “mldb\_sample.dat”) != 0) {  printf (“import 오류 \n”);  } else {  printf(“import 성공 \n”);  }  …. | | | |

|  |
| --- |
| SQL용 질의함수들 |

|  |
| --- |
| **iSQL\_affrected\_rows** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **iSQL\_affected\_rows()** | | | | | |
|  | unsigned int iSQL\_affected\_rows(iSQL \*isql); | | | | |
|  | **Arguments** | | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 | |
|  | **Return Values** | | | | |
|  |  | insert, update, delete 인 경우 | | | 적용된 row 개수 |
|  |  | select 인 경우 | | | fetch된 row 개수 |
|  |  | upsert인 경우 | | | 0 : 존재 / 1 : insert / 2 : update |
|  | **Description** | | | | |
|  | 질의문 (insert, update, delete, select)에 의해 수행된 row 개수를 얻어오는 함수이다.  upsert 구문인경우 다른 구문과 다르게 0인 경우 존재, 1인 경우 insert, 2인 경우 update를 표시한다. | | | | |
|  | **Sample** | | | | |
|  | iSQL isql;  …..  if (iSQL\_query(&isql, “insert into mldb\_sample values(1, 2)”) == 0) {  printf(“affected row : %d\n”, **iSQL\_affected\_rows**(&isql));  } else {  printf(“질의 실패\n”);  }  …. | | | | |

|  |
| --- |
| **iSQL\_fetch\_row** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch\_row()** | | | | |
|  | iSQL\_ROW iSQL\_fetch\_row(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류발생 or 더 이상 가져올 record가 없을 경우 | |
|  |  | Otherwise | 다음 row의 iSQL\_ROW구조체 반환 | |
|  | **Description** | | | |
|  | SQL 질의 결과(iSQL\_RES 구조체)에서 다음 row를 fetch하는 함수이다. iSQL\_use\_result를 사용한 경우, 8Kbyte단위로 Data를 가져오게 된다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_ROW row;  ……..  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  res = iSQL\_store\_result(&isql);  if (res != NULL) {  while ((row = **iSQL\_fetch\_row**(res)) != NULL) {  …….  }  }  iSQL\_free\_result(res); | | | |

|  |
| --- |
| **iSQL\_field\_count** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_field\_count** | | | | |
|  | unsigned int iSQL\_field\_count(iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당하는 핸들값 |
|  | **Return Values** | | | |
|  |  |  | Column 개수 | |
|  | **Description** | | | |
|  | 가장 최근수행된 검색 질의에 의해 추출된 컬럼 수를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;   ……  if (iSQL\_query(&isql, L“select \* from mldb\_sample”) == 0) {  printf(“%s\n”,**iSQL\_field\_count**(&isql));  }else{ /\* 오류처리 \*/ }  ….  } | | | |

|  |
| --- |
| **iSQL\_free\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_free\_result** | | | | |
|  | void iSQL\_free\_result(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  |  |  | |
|  | **Description** | | | |
|  | iSQL\_RES 구조체에 할당된 메모리를 제거하는 함수이다.  메모리 누수를 없애기 위해서는 iSQL\_RES 구조체를 생성하는 함수를 사용한 후에 반드시 iSQL\_free\_result를 호출해야 한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  ……  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  res = iSQL\_store\_result(&isql);  …….  **iSQL\_free\_result**(res);  } | | | |

|  |
| --- |
| **iSQL\_last\_querytype** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_last\_querytype** | | | | |
|  | int iSQL\_last\_querytype (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 알 수 없는 Query Type | |
|  |  | Otherwise | 마지막 수행된 질의 type (isql\_stmt\_type 참조) | |
|  | **Description** | | | |
|  | 마지막에 수행된 질의 type을 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  if(iSQL\_query(&isql,”select \* from mldb\_sample”)==0){  printf(“%d\n”,**iSQL\_last\_querytype**(&isql));  }  …. | | | |

|  |
| --- |
| **iSQL\_nquery** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_nquery()** | | | | |
|  | int iSQL\_nquery(iSQL \*isql, DB\_WCHAR \*wQuery); | | | |
|  | **Arguments** | | | |
|  |  | Isql | IN | db명에 해당하는 핸들값 |
|  |  | wQuery | IN | UCS2 (UTF16)형태의 SQL질의문 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 유니코드(UCS2) 문자열 SQL질의문을 수행하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  …….   if (**iSQL\_nquery**(&isql, L“select \* from mldb\_sample”) == 0) {  res = iSQL\_store\_result(&isql);  ….  iSQL\_free\_result(res);  } | | | |

|  |
| --- |
| **iSQL\_num\_rows** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_num\_rows()** | | | | |
|  | unsigned int iSQL\_num\_rows(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | Res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | Otherwise | row개수 | |
|  | **Description** | | | |
|  | 수행된 검색 질의에 의해 추출된 record의 개수를 얻어오는 함수이다.  iSQL\_store\_result() 또는 iSQL\_stmt\_store\_result()를 수행한 후에는 전체 record수를 얻어오나, iSQL\_use\_result() 수행 후에는 현재까지 fetch된 record 개수를 얻어온다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  ……..  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  res = iSQL\_store\_result(&isql);  printf(“row count : %d\n”, **iSQL\_num\_rows**(res));  iSQL\_free\_result(res);  }  …. | | | |

|  |
| --- |
| **iSQL\_query** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_query()** | | | | |
|  | int iSQL\_query(iSQL \*isql,char \*query); | | | |
|  | **Arguments** | | | |
|  |  | Isql | IN | db명에 해당되는 핸들값 |
|  |  | query | IN | Ascii 문자열 SQL 질의문 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | Ascii 문자열 SQL 질의문을 수행하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  if (**iSQL\_query**(&isql, “select \* from mldb\_sample”) == 0) {  res = iSQL\_store\_result(&isql);  ….  iSQL\_free\_result(res);  } | | | |

|  |
| --- |
| **iSQL\_store\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_store\_result** | | | | |
|  | iSQL\_RES \*iSQL\_store\_result(iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | Isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 결과 구조체의 포인트 값 | |
|  | **Description** | | | |
|  | iSQL\_query()를 통해 수행된 SQL 질의문의 모든 결과값을 한꺼번에 지역 버퍼에 저장하는 함수이다.  이 함수에 의해 사용된 결과값(iSQL\_RES \*)은 iSQL\_free\_result()에 의해 반드시 free되어야 한다*.* | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  …..  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  res = **iSQL\_store\_result**(&isql);  ….  iSQL\_free\_result(res);  } | | | |

|  |
| --- |
| **iSQL\_use\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_use\_reslut()** | | | | |
|  | iSQL\_RES \*iSQL\_use\_result(iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | Isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 결과 구조체의 포인트 값 | |
|  | **Description** | | | |
|  | iSQL\_query()를 통해 수행된 SQL 질의문의 결과값 중 8Kbyte에 저장가능한 Record개수만큼만 지역버퍼에 가져온다.  나머지 Record들은 iSQL\_fetch\_row()사용하여 8kbyte씩 가져오게 된다.  이 함수에 의해 사용된 결과값(iSQL\_RES \*)는 iSQL\_free\_result()에 의해 반드시 free되어야 한다. (Library type에서 사용할 필요가 없음.) | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  …..  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  res = **iSQL\_use\_result**(&isql);  ….  iSQL\_free\_result(res);  } | | | |

|  |
| --- |
| **iSQL\_get\_plan\_string** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_get\_plan\_string()** | | | | |
|  | char \*iSQL\_get\_plan\_string(iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | Otherwise | Query Plan 문자열 | |
|  | **Description** | | | |
|  | 수행된 검색 질의에 대한 query plan을 추출해 주는 함수이다.  iSQL\_option()에서 OPT\_PLAN\_ON, OPT\_PLAN\_ONLY가 설정되어 있는 경우만 query plan 문자열을 반환한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  …..  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf("plan:%s\n"**,iSQL\_get\_plan\_string**(&isql));  ….  iSQL\_free\_result(res);  } | | | |

|  |
| --- |
| Statement을 이용한 질의수행 함수들 |

|  |
| --- |
| **iSQL\_bind\_param** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_bind\_param()** | | | | |
|  | int iSQL\_bind\_param(iSQL\_STMT \*stmt,iSQL\_BIND \*bind); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  |  | bind | IN | parameter를 바인딩할 구조체 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_prepare()에 의해 생성된statement에Parameter를 바인딩하는 함수이다.   * Parameter는 iSQL\_prepare()함수의 인자인 SQL 문장에서 ? 로 표현된 것을 말한다. * iSQL\_describe()를 이용할 경우에는 param bind buffer에 대해 고민을 하지 않아도 된다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND bind;  int is\_null;  char data[32];  char \*query = "select \* from mldb\_samplewhere id = ?";  ….  if (!(stmt = iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  bind.buffer\_type = SQL\_DATA\_INT;  bind.is\_null = &is\_null;  bind.buffer = data;  bind.buffer\_length = sizeof(int);  if (**iSQL\_bind\_param**(stmt, bind)) { /\* 오류처리 \*/ }  else { /\* 정상처리 \*/ }  …. | | | |

|  |
| --- |
| **iSQL\_bind\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_bind\_result()** | | | | |
|  | int iSQL\_bind\_result (iSQL\_STMT \*stmt,iSQL\_bind \*bind); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  |  | bind | IN | parameter를 바인딩할 구조체 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_prepare()에 의해 생성된statement에 결과 값을 바인딩하는 함수이다.   * iSQL\_describe()를 이용할 경우에는 result bind buffer에 대해 고민을 하지 않아도 된다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_BIND bind[3];  iSQL\_STMT \*stmt;  int i, is\_null[3];  unsigned long length[3];  char data[3][32];  char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt = iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  for (i = 0; i < sizeof(bind)/sizeof(bind[0]); i++)  {  bind[i].buffer\_type= SQL\_DATA\_CHAR;  bind[i].is\_null= &is\_null[i];  bind[i].length= &length[i];  bind[i].buffer = data[i];  bind[i].buffer\_length = sizeof(data[i]);  }  if (**iSQL\_bind\_result**(stmt, bind)){ /\* 오류처리 \*/ }  …… | | | |

|  |
| --- |
| **iSQL\_describe** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_describe()** | | | | |
|  | int iSQL\_describe(iSQL\_STMT \*stmt,iSQL\_BIND \*\*bind\_param,iSQL\_BIND \*\*bind\_res) | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  |  | bind\_param | IN | Parameter를 바인딩하는 구조체 |
|  |  | bind\_res | IN | Result를 바인딩하는 구조체 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 statement에 대한 정보를 분석하여 Parameter를 바인딩을 위한 buufer와 결과 값 바인딩을 위한 buffer를 미리 할당하는 함수이다.   * iSQL\_stmt\_close()를 이용하여 반드시 메모리를 free해야한다. | | | |
|  | **Sample** | | | |
|  | iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  int ret;  char \*query = "select \* from mldb\_sample limit 10";  …..  if(!(stmt=iSQL\_prepare(isql, query, strlen(query)))){ /\* 오류처리 \*/ }  ret = **iSQL\_describe**(stmt, &bindParam, &bindRes);  …… | | | |

|  |
| --- |
| **iSQL\_execute** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_execute()** | | | | |
|  | int iSQL\_execute(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_prepare()에 준비된 질의문을 수행하는 함수이다.  iSQL\_prepare()과 iSQL\_describe()에 의해 준비된 질의문의 Parameter 값을 변경하면서 반복 수행 가능하다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!**iSQL\_execute**(stmt)) { /\* 오류처리 \*/ }  else { /\* 정상처리 \*/ }  … | | | |

|  |
| --- |
| **iSQL\_fetch** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch()** | | | | |
|  | int iSQL\_fetch(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 stmt로부터 현재 record 값을 바인딩된 버퍼로 가져오는 함수이다.  iSQL\_prepare()와 iSQL\_execute()를 사용한 경우에만 사용 가능하다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  int ret;  char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  while (1)  {  ret = **iSQL\_fetch**(stmt)**;**  if (ret == iSQL\_NO\_DATA) break;  else if (ret < 0) { /\* 오류처리 \*/ }  }  ….. | | | |

|  |
| --- |
| **iSQL\_nprepare** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_nprepare()** | | | | |
|  | iSQL\_STMT iSQL\_nprepare(iSQL \*isql, DB\_WCHAR \*wQuery, unsigned long int length); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | wQuery | IN | 유니코드(UCS2) 문자열 질의문 |
|  |  | length | IN | 유니코드(UCS2) 질의문의 문자열 길이 값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(statement의 핸들값) | |
|  | **Description** | | | |
|  | 유니코드(UCS2)형태의 질의문을 실행하기 위해 준비하는 함수이다.  Return된 iSQL\_stmt 구조체는 사용후 반드시 iSQL\_close()를 통해서 할당된 메모리를 free해야 한다. 최대 stmt핸들 개수는 32767개이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  DB\_WCHAR \*wQuery = "select \* from mldb\_sample where nchar\_field = ?"; // 실질적으로는 UCS2 string 이여야 함.  …  if (!(stmt= **iSQL\_nprepare**(&isql, wQuery, wcslen(wQuery)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  …… | | | |

|  |
| --- |
| **iSQL\_nprepare2** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_nprepare2()** | | | | |
|  | int iSQL\_nprepare2(iSQL \*isql, DB\_WCHAR \*wQuery, unsigned long int length, isql\_res\_type resulttype); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | wQuery | IN | 유니코드 문자열 질의문 |
|  |  | length | IN | 유니코드 질의문의 분자열 길이 값 |
|  |  | resulttype | IN | 결과 반환 type  SQL\_RES\_BINARY : field type에 맞추어 반환  SQL\_RES\_STRING : 모든 field의 결과를 문자열로 반환 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(statement핸들값) | |
|  | **Description** | | | |
|  | iSQL\_nprepare() 함수와 동일하지만, 두 가지의 결과값 형태 (SQL\_RES\_BINARY 또는 SQL\_RES\_STRING)로 반환할 수 있게 하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  DB\_WCHAR \*wQuery = "select \* from mldb\_sample where nchar\_field = ?"; // 실질적으로는 UCS2 string 이여야 함.  …  if (!(stmt= **iSQL\_nprepare2**(&isql, wQuery, wcslen(wQuery),SQL\_RES\_STRING))) {/\*오류처리\*/}  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  …. | | | |

|  |
| --- |
| **iSQL\_num\_parameter\_fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_num\_parameter\_fields()** | | | | |
|  | int iSQL\_num\_parameter\_fields(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 statement로부터 Parameter로 바인딩된 Field의 개수를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  int cnt;  char \*query = "select \* from mldb\_sample where id = ? limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  cnt= **iSQL\_num\_parameter\_fields**(stmt); | | | |

|  |
| --- |
| **iSQL\_num\_result\_fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_num\_result\_fields()** | | | | |
|  | int iSQL\_num\_result\_fields(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 statement의결과 Field의 개수를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  int cnt;  char \*query = "select \* from mldb\_sample where id = ? limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  cnt= **iSQL\_num\_result\_fields**(stmt); | | | |

|  |
| --- |
| **iSQL\_param\_count** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_param\_count()** | | | | |
|  | unsigned int iSQL\_param\_count (iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | Otherwise | Parameter의 개수 | |
|  | **Description** | | | |
|  | 해당 statement의 Parameter 개수를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  int cnt;  char \*query = "select \* from mldb\_sample where id = ? limit ?";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  cnt= **iSQL\_ paran\_count**(stmt);  ….. | | | |

|  |
| --- |
| **iSQL\_prepare** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_prepare()** | | | | |
|  | iSQL\_STMT \*iSQL\_prepare(iSQL \*sql, char \*query, unsigned long int length); | | | |
|  | **Arguments** | | | |
|  |  | sql | IN | db명에 해당되는 핸들값 |
|  |  | query | IN | 질의문 |
|  |  | length | IN | 질의문에 대한 byte 길이값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(statement 핸들값) | |
|  | **Description** | | | |
|  | Ascii 형태의 질의문을 실행하기 위해 준비하는 함수이다.  Return된 iSQL\_stmt 구조체는 사용후 반드시 iSQL\_close()를 통해서 할당된 메모리를 free해야 한다. 최대 stmt핸들 개수는 32767개이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= **iSQL\_prepare**(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  …. | | | |

|  |
| --- |
| **iSQL\_prepare2()** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_prepare2()** | | | | |
|  | iSQL\_STMT \*iSQL\_prepare2(iSQL \*sql, char \*query, unsigned long int length, isql\_res\_type resulttype); | | | |
|  | **Arguments** | | | |
|  |  | sql | IN | db명에 해당되는 핸들값 |
|  |  | query | IN | 질의문 |
|  |  | length | IN | 질의문에 대한 byte 길이값 |
|  |  | resulttype | IN | 결과 반환 type  SQL\_RES\_BINARY : field type에 맞추어 반환  SQL\_RES\_STRING : 모든 field의 결과를 문자열로 반환 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(statement 핸들값) | |
|  | **Description** | | | |
|  | iSQL\_prepare() 함수와 동일하지만, 두 가지 형태의 결과값으로 (SQL\_RES\_BINARY 또는 SQL\_RES\_STRING)로 반환할 수 있게 하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= **iSQL\_prepare2**(&isql, query, strlen(query), SQL\_RES\_STRING))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류 처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  ….. | | | |

|  |
| --- |
| **iSQL\_prepare\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_prepare\_result()** | | | | |
|  | iSQL\_RES \*iSQL\_prepare\_result(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | iSQL\_RES 포인터 값 | |
|  | **Description** | | | |
|  | Stmtement의 field에 대한 메타정보(Field의 정보)를 결과 형태로 얻어오는 함수이다.  return된 iSQL\_RES pointer를 가지고 field관련 함수만 사용할 수 있다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  iSQL\_RES \*result;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!(result= **iSQL\_prepare\_result**(stmt))) { /\* 오류처리 \*/ }  if (iSQL\_num\_fields(result) != 3) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  ….. | | | |

|  |
| --- |
| **iSQL\_stmt\_affected\_rows** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_affected\_rows()** | | | | |
|  | unsigned int iSQL\_stmt\_affected\_rows(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | Otherwise | Row 개수 | |
|  | **Description** | | | |
|  | 마지막으로 수행한 statement에 의해 변경된 row 개수를 얻어오는 함수이다.  Select인 경우에는 fetch된 row 개수를 반환한다. 이 함수는 iSQL\_execute() 후에 바로 수행해야 한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  char \*query = "update mmdb set name = 'UPDATE' where id = 1";  ….  if(!(stmt=iSQL\_prepare(&isql,query,strlen(query)))){/\*오류처리\*/}  printf("%u record(s) is updated successfully\n", **iSQL\_stmt\_affected\_rows**(stmt));  ….. | | | |

|  |
| --- |
| **iSQL\_stmt\_close** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_close()** | | | | |
|  | int iSQL\_stmt\_close(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | Otherwise | 오류 | |
|  | **Description** | | | |
|  | 해당 statement를 close하는 함수이다.  이 함수는 statement에 할당된 메모리를 free하고, 가져오지 않은 결과값이 있으면 모두 취소한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  ….  If (**iSQL\_stmt\_close**(stmt) < 0) { /\* 오류처리 \*/ } | | | |

|  |
| --- |
| **iSQL\_stmt\_plan\_string** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_plan\_string()** | | | | |
|  | char iSQL\_stmt\_plan\_string(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 statement 대한query plan을 추출해 주는 함수이다.  iSQL\_option()에서 OPT\_PLAN\_ON, OPT\_PLAN\_ONLY가 설정되어 있는 경우만 query plan 문자열을 반환한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (iSQL\_options(&isql, OPT\_PLAN\_ON, (const void \*)&onoff, sizeof(int)) != 0 {/\*오류처리\*/}  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  **iSQL\_stmt\_plan\_string**(stmt); | | | |

|  |
| --- |
| **iSQL\_stmt\_querytype** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_querytype()** | | | | |
|  | int iSQL\_stmt\_querytype(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 마지막으로 실행 준비된 statement문의 질의 type을 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (iSQL\_options(&isql, OPT\_PLAN\_ON, (const void \*)&onoff, sizeof(int)) != 0 {/\*오류처리\*/}  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  printf(“%d\n”, **iSQL\_stmt\_querytype**(stmt)); | | | |

|  |
| --- |
| **iSQL\_stmt\_store\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_store\_result()** | | | | |
|  | int iSQL\_stmt\_store\_result(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_store\_result()처럼, 해당 statement의 결과값 전체를 지역 버퍼로 저장하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  if (**iSQL\_stmt\_store\_result**(stmt) < 0) { /\* 오류처리 \*/ }  … | | | |

|  |
| --- |
| **iSQL\_stmt\_use\_result** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_use\_result()** | | | | |
|  | int iSQL\_stmt\_use\_result(iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | Stmt | IN | statement의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_use\_result()처럼, 해당 statement의 결과중 8Kbyte에 해당하는 Record개수만 지역 버퍼로 저장하는 함수이다. 필요한 경우, 나머지 Record들은 iSQL\_fetch\_row()사용하여 8kbyte씩 가져오게 된다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  if (**iSQL\_stmt\_use\_result**(stmt) < 0) { /\* 오류처리 \*/ }  … | | | |

|  |
| --- |
| Rid 관련 함수들 |

|  |
| --- |
| **iSQL\_affected\_rid** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_affected\_rid()** | | | | |
|  | unsigned long iSQL\_affected\_rid(iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | Insert 질의가 수행되지 않음  (insert 실패 또는 수행된 질의가 insert 질의가 아님) | |
|  |  | < 0 | 성공(insert 한 rid) | |
|  | **Description** | | | |
|  | data insert 후에 호출하면 insert 한 rid를 반환하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  OID rid;  char \*query = "insert into tbl values(‘test’)";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  …  rid= **iSQL\_affected\_rid** (&isql);  ….. | | | |

|  |
| --- |
| **iSQL\_drop\_rid** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_drop\_rid()** | | | | |
|  | int iSQL\_drop\_rid(iSQL \*isql, OID rid); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | rid | IN | 레코드 ID |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | rid에 해당하는 레코드를 삭제하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  int ret;  OID rid;  Char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  while (1)  {  ret = iSQL\_fetch(stmt);  if (ret == iSQL\_NO\_DATA) break;  iSQL\_get\_rid(stmt, &rid);  **iSQL\_drop\_rid**(&isql, rid);  }  ….. | | | |

|  |
| --- |
| **iSQL\_get\_rid** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_get\_rid()** | | | | |
|  | int iSQL\_get\_rid(iSQL\_STMT \*stmt, OID \*rid); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | db명에 해당되는 핸들값 |
|  |  | rid | IN | 레코드 ID |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 statement의 결과 값에서 현재 커서가 가리키고 있는 rid 값을 얻는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes;  int ret;  OID rid;  Char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  while (1)  {  ret = iSQL\_fetch(stmt);  if (ret == iSQL\_NO\_DATA) break;  **iSQL\_get\_rid**(stmt, &rid);  }  … | | | |

|  |
| --- |
| **iSQL\_update\_rid** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_update\_rid()** | | | | |
|  | int iSQL\_update\_rid(iSQL \*isql, OID rid, char \*fieldname, iSQL\_BIND \*fieldvalue); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | rid | IN | 레코드 ID |
|  |  | fieldname | IN | 파일 명 |
|  |  | fieldvalue | IN | Field 값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 rid의 레코드 중 하나의 field 값을 변경하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes, updateParam;  int ret;  OID rid;  Char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  while (1)  {  ret = iSQL\_fetch(stmt);  if (ret == iSQL\_NO\_DATA) break;  iSQL\_get\_rid(stmt, &rid);  …..  **iSQL\_update\_rid**(&isql, rid, “name”, &updateParam);  }  ….. | | | |

|  |
| --- |
| **iSQL\_desc\_update\_rid** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_desc\_update\_rid()** | | | | |
|  | int iSQL\_desc\_update\_rid(iSQL \*isql, OID rid, int numfields, iSQL\_FIELD \*fielddesc, iSQL\_BIND \*fieldvalue); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  |  | rid | IN | 레코드 ID |
|  |  | numfields | IN | Field 개수 |
|  |  | fielddesc | IN | Field 설명 |
|  |  | fieldvalue | IN | Field 값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 해당 rid의 레코드 중 동시에 여러 field 값을 변경하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  iSQL\_BIND \*bindParam, \*bindRes, updateParam;  iSQL\_FIELD updateField;  int ret;  OID rid;  Char \*query = "select \* from mldb\_sample limit 10";  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) { /\* 오류처리 \*/ }  if ( iSQL\_describe(stmt, &bindParam, &bindRes)< 0) { /\* 오류처리 \*/ }  if (!iSQL\_execute(stmt)) { /\* 오류처리 \*/ }  while (1)  {  ret = iSQL\_fetch(stmt);  if (ret == iSQL\_NO\_DATA) break;  iSQL\_get\_rid(stmt, &rid);  …..  **iSQL\_desc\_update\_rid**(&isql, rid, 3, &updateField, &updateParam);  }  ….. | | | |

|  |
| --- |
| 결과 Set 관련 함수 |

|  |
| --- |
| **iSQL\_data\_seek** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_data\_seek()** | | | | |
|  | long int iSQL\_data\_seek(iSQL\_RES \*res,long int offset,int whence); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | offset | IN | 이동하려는 row 개수 |
|  |  | whence | IN | 지정 위치 (처음, 현재, 끝)  iSQL\_DATA\_SEEK\_START  iSQL\_DATA\_SEEK\_CURRENT  iSQL\_DATA\_SEEK\_END |
|  | **Return Values** | | | |
|  |  | < 0 | 오류 | |
|  |  | Otherwise | 성공(위치 값) | |
|  | **Description** | | | |
|  | SQL 질의 수행에 의해 생성된 결과 set에서 지정위치(whence)를 기준으로 offset만큼을 이동하여 현재 커서의 위치를 설정하는 함수이다.  이 함수는 모든 결과를 지역 버퍼에 저장되어야만 사용 가능하므로 반드시 iSQL\_store\_result() 또는 iSQL\_stmt\_store\_result() 함수가 호출된 후에 사용 가능하다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) { /\* 오류 처리 \*/ }  res = iSQL\_store\_result(&isql);  …  **iSQL\_data\_seek**(res, 2, iSQL\_DATA\_SEEK\_START);  … | | | |

|  |
| --- |
| **iSQL\_data\_tell** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_data\_tell()** | | | | |
|  | long int SQL\_data\_tell(iSQL\_RES \*res) | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | -1 | 오류 | |
|  |  | Otherwise | 결과 set에서 현재 가르키고 있는 위치 | |
|  | **Description** | | | |
|  | SQL 질의 수행에 의해 생성된 결과 set에서 현재위치값을 구하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  int record\_offset;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) { /\* 오류 처리 \*/ }  res = iSQL\_store\_result(&isql);  …  record\_offset =**iSQL\_data\_tell**(res);  … | | | |

|  |
| --- |
| **iSQL\_eof** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_eof()** | | | | |
|  | int iSQL\_eof(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 비정상적인 종료 | |
|  |  | 1 | 정상적인 종료 | |
|  | **Description** | | | |
|  | SQL 질의 수행 후 결과의 모든 row를 다 읽었는 지를 검사하는 함수이다.  이 함수는 iSQL\_fetch\_row()의 return값이 NULL인 경우, 더 이상의 row가 없어서 NULL이 return된 것인지, 아니면 오류에 의한 것인지를 검사할 때 사용한다. | | | |
|  | **Sample** | | | |
|  | iSQL\_RES \*res;  iSQL\_ROW row;  …..  if ((row = iSQL\_fetch\_row(res)) == NULL) {  if (**iSQL\_eof**(res) == 0) printf(“비정상적인 종료 \n”);  else printf(“row를 다 읽었음 \n”);  }  …. | | | |

|  |
| --- |
| **iSQL\_row\_seek** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_row\_seek()** | | | | |
|  | iSQL\_ROW\_OFFSET iSQL\_row\_seek(iSQL\_RES \*res,iSQL\_ROW\_OFFSET row); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | row | IN | 이동하고자 하는 커서의 offset 값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | iSQL\_ROW\_OFFSET type의 offset 값 | |
|  | **Description** | | | |
|  | 커서를 주어진 offset 위치으로 이동시키는 함수이다.  이 함수는 모든 결과를 지역 버퍼에 저장되어야만 사용 가능하므로 반드시 iSQL\_store\_result() 또는 iSQL\_stmt\_store\_result() 함수가 호출된 후에 사용 가능하다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_ROW\_OFFSET offset;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) { /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  offset = iSQL\_row\_tell(res);  …  **iSQL\_row\_seek**(res, offset);  …. | | | |

|  |
| --- |
| **iSQL\_row\_tell** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_row\_tell()** | | | | |
|  | iSQL\_ROW\_OFFSET iSQL\_row\_tell(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 현재 커서의 위치 | |
|  | **Description** | | | |
|  | 결과의 현재 커서 위치를 얻어오는 함수이다.  이 함수는 모든 결과를 지역 버퍼에 저장되어야만 사용 가능하므로 반드시 iSQL\_store\_result() 또는 iSQL\_stmt\_store\_result() 함수가 호출된 후에 사용 가능하다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_ROW\_OFFSET offset;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) { /\* 오류 처리 \*/ }  res = iSQL\_store\_result(&isql);  offset = **iSQL\_row\_tell**(res);  …  iSQL\_row\_seek(res, offset);  …. | | | |

|  |
| --- |
| Field 관련 함수들 |

|  |
| --- |
| **iSQL\_fetch\_field** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch\_field()** | | | | |
|  | iSQL\_FIELD \*iSQL\_fetch\_field(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 더 이상의 Field 정보가 없음 | |
|  |  | Otherwise | 성공 (iSQL\_FIELD 구조체 포인터) | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과 값으로부터 한 Field씩 순차적으로 Field 정보를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  field=**iSQL\_fetch\_field**(res);  printf(“name:%s,table:%s\n”,field->name,field->table);  ….. | | | |

|  |
| --- |
| **iSQL\_fetch\_fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch\_fields** | | | | |
|  | iSQL\_FIELD \*iSQL\_fetch\_fields(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | iSQL\_FIELD array 값 | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과 값으로부터 모든 Field의 정보를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  int i;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  field=**iSQL\_fetch\_fields**(res);  for (i=0; i < iSQL\_num\_fields(res); i++) {  printf(“name:%s,table:%s\n”,field[i]->name,field[i]->table);  }  ….. | | | |

|  |
| --- |
| **iSQL\_fetch\_field\_direct** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch\_field\_direct()** | | | | |
|  | iSQL\_FIELD \*iSQL\_fetch\_field\_direct (iSQL\_RES \*res,unsigned int findex); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | findex | IN | 결과에서 Field의 순서값  값은 0 ~ iSQL\_num\_fields(res)-1 |
|  | **Return Values** | | | |
|  |  | NULL | 잘못된 Field 순서값 | |
|  |  | Otherwise | 성공 (iSQL\_FIELD 구조체 포인터) | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과 값으로부터 Field 순서값(findex)에 해당되는 Field의 정보를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  field=**iSQL\_fetch\_field\_direct**(res, 1);  printf(“name:%s,table:%s\n”,field->name,field->table);  ….. | | | |

|  |
| --- |
| **iSQL\_fetch\_lengths** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_fetch\_lengths()** | | | | |
|  | iSQL\_LENGTH iSQL\_fetch\_lengths(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | iSQL\_LENGTH 값 | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과 값에서 레코드의 모든 컬럼에 저장되어 있는 데이터 길이 값을 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_LENGTH lengths;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){/\*오류처리\*/}  res = iSQL\_store\_result(&isql);  iSQL\_fetch\_row(res);  lengths **= iSQL\_fetch\_lengths**(res);  for (i = 0; i < iSQL\_num\_fields(res); i++) {  printf(“field[%d] : %d\n”, i, lengths[i]);  }  … | | | |

|  |
| --- |
| **iSQL\_field\_seek** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_field\_seek** | | | | |
|  | iSQL\_FIELD\_OFFSET iSQL\_field\_seek(iSQL\_RES \*res,iSQL\_FIELD\_OFFSET field\_offset); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | field\_offset | IN | 이동하고자 하는 Field 커서의 offset 값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(isql 핸들값) | |
|  | **Description** | | | |
|  | Field 커서를 주어진 offset으로 이동시키는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  iSQL\_FIELD\_OFFSET field\_offset;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){/\*오류처리\*/}  res = iSQL\_store\_result(&isql);  iSQL\_fetch\_row(res);  iSQL\_fetch\_field(res);  field\_offset = iSQL\_field\_tell(res);  …….  **iSQL\_field\_seek**(res, field\_offset);  ….. | | | |

|  |
| --- |
| **iSQL\_field\_tell** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_field\_tell** | | | | |
|  | iSQL\_FIELD\_OFFSET iSQL\_field\_tell (iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | Otherwise | Field 커서의 현재 offset | |
|  | **Description** | | | |
|  | Field의 커서 현재 위치를 가져오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  iSQL\_FIELD\_OFFSET field\_offset;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  iSQL\_fetch\_row(res);  iSQL\_fetch\_field(res);  field\_offset = **iSQL\_field\_tell**(res);  …….  iSQL\_field\_seek(res, field\_offset);  ….. | | | |

|  |
| --- |
| **iSQL\_get\_fieldtype** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_get\_fieldtype** | | | | |
|  | int iSQL\_get\_fieldtype(iSQL\_RES \*res,const unsigned int findex); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | findex | IN | 결과에서 Field의 순서값  값은 0 ~ iSQL\_num\_fields(res)- |
|  | **Return Values** | | | |
|  |  | 0 | 잘못된 type 또는 잘못된 Field id | |
|  |  | Otherwise | 해당 Field의 type값(enum isql\_data\_type을 참조) | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과값에서 해당 Field(findex)의 Field type을 가지고 오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_FIELD \*field;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  printf(“ data type : %d\n”, **iSQL\_get\_fieldtype**(res, 1));  …. | | | |

|  |
| --- |
| **iSQL\_list\_fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_list\_fields()** | | | | |
|  | iSQL\_RES \*iSQL\_list\_fields(iSQL \*isql,char \*table,char \*column); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  |  | table | IN | 테이블명 |
|  |  | column | IN | wild card(%)를 사용한 Field명 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공 | |
|  | **Description** | | | |
|  | 해당 테이블명에서 wild char(%)를 이용하여 해당하는 컬럼 정보를 얻어오는 함수.  만약 wild char가 NULL이면 해당 테이블의 모든 컬럼 정보를 얻어온다. | | | |
|  | **Sample** | | | |
|  | res = **iSQL\_list\_fields**(&isql, “SYSTABLES”,NULL); | | | |

|  |
| --- |
| **iSQL\_num\_fields** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_num\_fields()** | | | | |
|  | unsigned int iSQL\_num\_fields(iSQL\_RES \*res); | | | |
|  | **Arguments** | | | |
|  |  | res | IN | 결과의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 | |
|  |  | Otherwise | 성공(컬럼 수) | |
|  | **Description** | | | |
|  | 질의문 수행에 의해 생성된 결과값에서 컬럼 개수를 얻어오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_RES \*res;  iSQL\_LENGTH lengths;  …..  if(iSQL\_query(&isql,”select \* from mldb\_sample where id = 1”){ /\* 오류처리 \*/ }  res = iSQL\_store\_result(&isql);  iSQL\_fetch\_row(res);  lengths = iSQL\_fetch\_lenghts(res);  for (i = 0; i <**iSQL\_num\_fields**(res); i++) {  printf(“field[%d] : %d\n”, i, lengths[i]);  }  … | | | |

|  |
| --- |
| 오류 처리 함수들 |

|  |
| --- |
| **iSQL\_errno** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_errno()** | | | | |
|  | int iSQL\_errno (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 오류 없음 (no error) | |
|  |  | Otherwise | 마지막으로 발생된 오류 번호 | |
|  | **Description** | | | |
|  | iSQL\_query() 계열에 의해 마지막으로 발생된 오류 번호를 얻어 오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, **iSQL\_errno**(&isql));  }  …. | | | |

|  |
| --- |
| **iSQL\_error** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_error()** | | | | |
|  | char \*iSQL\_error (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 없음 (no error) | |
|  |  | Otherwise | 마지막으로 발생된 오류에 관한 문자열 | |
|  | **Description** | | | |
|  | iSQL\_query() 계열에 의해 마지막으로 발생된 오류에 관한 문자열을 얻어오는 함수이다.  library code size를 줄이기 위해 error string을 없앤 경우에는 빈 문자열이 넘어오는 경우가 있다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error : %s\n”, **iSQL\_error**(&isql));  }  …. | | | |

|  |
| --- |
| **iSQL\_stmt\_errno** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_errno()** | | | | |
|  | int iSQL\_stmt\_errno (iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | stmt의 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 오류 없음 (no error) | |
|  |  | Otherwise | 마지막으로 발생된 오류 번호 | |
|  | **Description** | | | |
|  | 해당 statement에 대해 마지막으로 발생된 오류 번호를 얻어 오는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) {  printf(“error code : %d\n”, **iSQL\_stmt\_errno**(stmt));  }  …. | | | |

|  |
| --- |
| **iSQL\_stmt\_error** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_stmt\_error()** | | | | |
|  | char \*iSQL\_stmt\_error (iSQL\_STMT \*stmt); | | | |
|  | **Arguments** | | | |
|  |  | stmt | IN | stmt의 핸들값 |
|  | **Return Values** | | | |
|  |  | NULL | 오류 없음 (no error) | |
|  |  | Otherwise | 마지막으로 발생된 오류에 관한 문자열 | |
|  | **Description** | | | |
|  | 해당 stmt에 대해, 마지막으로 발생된 오류에 관한 문자열을 얻어오는 함수 이다.  library code size를 줄이기 위해 error string을 없앤 경우에는 빈 문자열이 넘어오는 경우가 있다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  iSQL\_STMT \*stmt;  char \*query = "select \* from mldb\_sample where id = ? limit ?”;  …  if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) {  printf(“error : %s\n”, **iSQL\_stmt\_error**(stmt));  }  …. | | | |

|  |
| --- |
| Transaction 관련 함수들 |

|  |
| --- |
| **iSQL\_begin\_transaction** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_begin\_transcation()** | | | | |
|  | int iSQL\_begin\_transaction (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | 명시적으로 트랜잭션를 시작하는 함수이다.  이 함수는 commit 또는 rollback 후 SQL문장을 수행하기 전에 사용해야 한다.  이 함수를 명시적으로 호출하지 않은 경우에는 시스템 내부적으로 트랜잭션을 시작한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  **iSQL\_begin\_transaction**(&isql);  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, iSQL\_errno(&isql));  }  …. | | | |

|  |
| --- |
| **iSQL\_commit** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_commit()** | | | | |
|  | int iSQL\_commit (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | Transaction내에서 변경된 모든 사항을 영구적으로 반영하는 함수이다.  DDL(Data Definition Language) 질의인 경우 시스템 내부적으로 auto-commit을 수행한다. 또한 검색질의인 경우에도 commit를 수행해서 시스템 내부적으로 할당되어 있는 lock을 풀어주어야 한다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  iSQL\_begin\_transaction(&isql);  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, iSQL\_errno(&isql));  }  **iSQL\_commit**(&isql);  …. | | | |

|  |
| --- |
| **iSQL\_commit\_flush** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_commit\_flush()** | | | | |
|  | int iSQL\_commit\_flush (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_commit()과 동일한 함수로 data cache buffer에 할당되어 있는 memory를 release 하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  iSQL\_begin\_transaction(&isql);  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, iSQL\_errno(&isql));  }  **iSQL\_commit\_flush**(&isql);  …. | | | |

|  |
| --- |
| **iSQL\_rollback** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_rollback()** | | | | |
|  | int iSQL\_rollback (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | Isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | Transaction내에서 변경된 모든 사항을 취소하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  iSQL\_begin\_transaction(&isql);  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, iSQL\_errno(&isql));  }  **iSQL\_rollback**(&isql);  …. | | | |

|  |
| --- |
| **iSQL\_rollback\_flush** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **iSQL\_rollback\_flush()** | | | | |
|  | int iSQL\_rollback\_flush (iSQL \*isql); | | | |
|  | **Arguments** | | | |
|  |  | isql | IN | db명에 해당되는 핸들값 |
|  | **Return Values** | | | |
|  |  | 0 | 성공 | |
|  |  | < 0 | 오류 | |
|  | **Description** | | | |
|  | iSQL\_rollback()과 동일한 함수로 data cache buffer에 할당되어 있는 memory를 release 하는 함수이다. | | | |
|  | **Sample** | | | |
|  | iSQL isql;  ….  iSQL\_begin\_transaction(&isql);  if (iSQL\_query(&isql, “select \* from mldb\_sample”) == 0) {  printf(“error code : %d\n”, iSQL\_errno(&isql));  }  **iSQL\_rollback\_flush**(&isql);  …. | | | |

|  |
| --- |
| 예제 프로그램 |

|  |
| --- |
| **Table / Index생성** |

|  |
| --- |
| Static SQL함수를 사용하여 Table / Index를 생성하는 예제와 함수들을 설명한다. |

[그림 ] Table/Index생성

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를이용하여 OpenML (embedded DBMS)에접속 2. Execute Statement : iSQL\_query를이용하여질의수행    1. CREATE TABLE :사용자가지정한 table을생성    2. CREATE INDEX :사용자가지정한 table의 colum에 index를생성 3. Disconnect Statement : iSQL\_disconnect( ) 함수를이용하여 OpenML와연결해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **char query[BUFSIZ];**  **if (!iSQL\_connect(&isql, "127.0.0.1", "mldb", "mldb", "mldb")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **snprintf(query, sizeof(query), "create table mldb\_sample(id int primary key, name varchar(20), age int)");**  **if (iSQL\_query(&isql, query)) {**  **printf("CREATE(table) failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **snprintf(query, sizeof(query), "create index idx\_msmp on mldb\_sample(name)");**  **if (iSQL\_query(&isql, query))**  **printf("CREATE(index) failed: %s\n", iSQL\_error(&isql));**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **Table 삭제** |

|  |
| --- |
| Static SQL함수를 사용하여 Table / Index를 삭제하는 예제와 함수들을 설명한다. |

[그림 ] Table/Index 삭제

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query를 이용하여 질의 수행    1. DROP TABLE : 사용자가 지정한 table을 제거 (table에 속해 있는 index들은 자동 제거) 3. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **if (!iSQL\_connect(&isql, "127.0.0.1", "mldb", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **if (iSQL\_query(&isql, "drop table mldb\_sample"))**  **printf("DROP failed: %s\n", iSQL\_error(&isql));**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_query를 이용한 Insert** |

|  |
| --- |
| Static SQL 함수를 이용하여 table에 record를 삽입하는 예제이다. |

[그림 ] iSQL\_query를 이용한 Insert

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query를 이용하여 질의 수행    1. INSERT INTO : 존재하는 table에 record 삽입 3. Commit Statement    1. iSQL\_commit( ) : 현재 transaction의 모든 변경사항을 반영    2. iSQL\_rollback( ) : 현재 transaction의 모든 변경사항을 취소 4. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **#define INSERT\_ITEMS 1000**  **int main(void)**  **{**  **iSQL isql;**  **char query[BUFSIZ];**  **int i;**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **for (i = 0; i < INSERT\_ITEMS; i++) {**  **snprintf(query, sizeof(query), "insert into mldb\_sample values (%d,'NAME\_%d',%d)", i, i, i);**  **if (iSQL\_query(&isql, query)) {**  **printf("INSERT failed: %s\n", iSQL\_error(&isql));**  **iSQL\_rollback(&isql);**  **goto DISCONNECT;**  **}**  **}**  **printf("%d record(s) is inserted successfully\n", INSERT\_ITEMS);**  **iSQL\_commit(&isql);**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_query를 이용한 Select** |

|  |
| --- |
| Static SQL 함수를 이용하여 table에 저장된 record를 검색하는 예제이다. |

[그림 ] iSQL\_query를 이용한 Select

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query( ) 함수를 이용하여 질의 수행    1. SELECT FROM : 하나 이상의 table에서 record 검색 3. Result Statement    1. iSQL\_store\_result( ) : SQL 질의문의 모든 결과값을 한꺼번에 지역 버퍼에 저장    2. iSQL\_fetch\_row( ) : SQL 질의의 결과에서 다음 row를 fetch    3. iSQL\_free\_result( ) : 할당된 메모리를 제거 4. Commit Statement    1. iSQL\_commit( ) : 현재 transaction의 모든 변경사항을 반영    2. iSQL\_rollback( ) : 현재 transaction의 모든 변경사항을 취소 5. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **iSQL\_RES \*res;**  **iSQL\_ROW row;**  **int i, j, num\_fields;**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **if (iSQL\_query(&isql, "select \* from mldb\_sample")) {**  **printf("SELECT failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **res = iSQL\_store\_result(&isql);**  **if (res == NULL)**  **printf("failed to store result: %s\n", iSQL\_error(&isql));**  **else {**  **num\_fields = iSQL\_num\_fields(res);**  **for (i = 0; (row = iSQL\_fetch\_row(res)) != NULL; i++) {**  **for (j = 0; j < num\_fields; j++)**  **printf("[%s]", row[j]);**  **printf("\n");**  **}**  **printf("%d record(s) was selected\n", i);**  **if (!iSQL\_eof(res))**  **printf("The last row was not read\n");**  **}**  **iSQL\_free\_result(res);**  **iSQL\_commit(&isql);**  **DISCONNECT:**  **iSQL\_rollback(&isql);**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_query를 이용한 Update** |

|  |
| --- |
| Static SQL 함수를 이용하여 table에 저장된 record를 수정하는 예제이다. |

[그림 ] iSQL\_query를 이용한 update

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query를 이용하여 질의 수행    1. UPDATE SET : 사용자가 지정한 table의 레코드 수정 3. Commit Statement    1. iSQL\_commit( ) : 현재 transaction의 모든 변경사항을 반영    2. iSQL\_rollback( ) : 현재 transaction의 모든 변경사항을 취소 4. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **char query[BUFSIZ];**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **snprintf(query, sizeof(query), "update mldb\_sample set name = 'UPDATED' where id = 99");**  **if (iSQL\_query(&isql, query)) {**  **printf("UPDATE failed: %s\n", iSQL\_error(&isql));**  **iSQL\_rollback(&isql);**  **goto DISCONNECT;**  **}**  **printf("%u record(s) is updated successfully\n", SQL\_affected\_rows(&isql));**  **iSQL\_commit(&isql);**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_query를 이용한 Delete** |

|  |
| --- |
| Static SQL 함수를 이용하여 table에 저장된 record를 삭제하는 예제이다. |

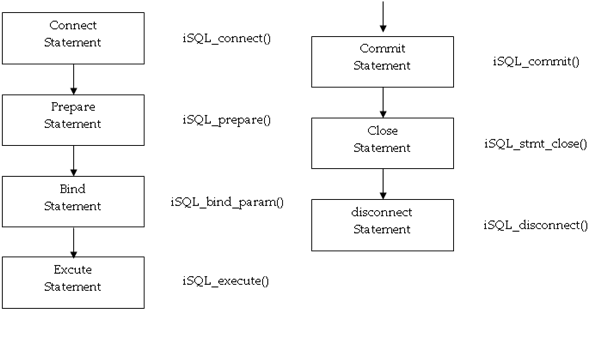
[그림 ] iSQL\_query를 이용한 delete

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query를 이용하여 질의 수행    1. DELETE FROM : 사용자가 지정한 table의 레코드 삭제 3. Commit Statement    1. iSQL\_commit( ) : 현재 transaction의 모든 변경사항을 반영    2. iSQL\_rollback( ) : 현재 transaction의 모든 변경사항을 취소 4. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **char query[BUFSIZ];**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **snprintf(query, sizeof(query), "delete from mldb\_sample where id = 100");**  **if (iSQL\_query(&isql, query)) {**  **printf("DELETE failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **printf("%u record(s) is deleted successfully\n", SQL\_affected\_rows(&isql));**  **iSQL\_commit(&isql);**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_prepare를 이용한 Insert** |

|  |
| --- |
| Dynamic SQL 함수를 이용하여 table에 record를 삽입하는 예제이다. |



[그림 ] iSQL\_prepare를 이용한 Insert

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Prepare Statement : iSQL\_prepare( ) 함수를 이용하여 질의문의 실행 전 준비과정을 수행 3. Bind Statement : iSQL\_bind\_param( ) 함수를 이용하여 해당 statement에 대한 parameter를 바인딩 4. Execute Statement : iSQL\_execute( ) 함수를 이용하여 iSQL\_prepare( ), iSQL\_bind\_param( ) 함수에서 준비된 질의문을 수행 5. Commit Statement : iSQL\_commit( ) 함수를 이용하여 현재 transaction의 모든 변경사항을 반영 6. Close Statement : iSQL\_stmt\_close( ) 함수를 이용하여 생성된statement를 close 7. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include <strings.h>**  **#include "isql.h"**  **#define INSERT\_ITEMS 10**  **char \*query = "insert into mldb\_sample values(?, ?, ?)";**  **int main(int ac, char \*\*av)**  **{**  **iSQL isql;**  **iSQL\_BIND bind[3];**  **iSQL\_STMT \*stmt;**  **char data[3][32];**  **int is\_null[3];**  **int i;**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) {**  **printf("Prepare failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **printf("Total parameters: %u\n", iSQL\_param\_count(stmt));**  **for (i = 0; i < sizeof(data)/sizeof(data[0]); i++) {**  **switch (i) {**  **case 0:**  **bind[i].buffer\_type = SQL\_DATA\_INT;**  **bind[i].is\_null = &is\_null[i];**  **bind[i].buffer = data[i];**  **bind[i].buffer\_length = sizeof(int);**  **break;**  **case 1:**  **bind[i].buffer\_type = SQL\_DATA\_VARCHAR;**  **bind[i].is\_null = &is\_null[i];**  **bind[i].buffer = data[i];**  **bind[i].buffer\_length = sizeof(data[i]);**  **break;**  **case 2:**  **bind[i].buffer\_type = SQL\_DATA\_INT;**  **bind[i].is\_null = &is\_null[i];**  **bind[i].buffer = data[i];**  **bind[i].buffer\_length = sizeof(int);**  **break;**  **default:**  **goto CLOSE\_AND\_DISCONNECT;**  **}**  **}**  **if (iSQL\_bind\_param(stmt, bind) < 0) {**  **printf("Parameter binding failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto CLOSE\_AND\_DISCONNECT;**  **}**  **is\_null[0] = is\_null[1] = 0; is\_null[2] = 1;**  **for (i = 0; i < INSERT\_ITEMS; i++) {**  **\*(int\*)data[0] = i;**  **snprintf(data[1], sizeof(data[1]), "NAME\_%d", i);**  **\*(int\*)data[2] = -i;**  **if (iSQL\_execute(stmt) < 0) {**  **printf("Execute failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto CLOSE\_AND\_DISCONNECT;**  **}**  **}**  **iSQL\_commit(&isql);**  **CLOSE\_AND\_DISCONNECT:**  **if (iSQL\_stmt\_close(stmt) < 0) {**  **printf("Failed to free the statement handle: %s\n", iSQL\_stmt\_error(stmt));**  **return -1;**  **}**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **iSQL\_prepare를 이용한 Select** |

|  |
| --- |
| Dynamic SQL 함수를 이용하여 table에 저장된 record를 읽어오는 예제이다. |

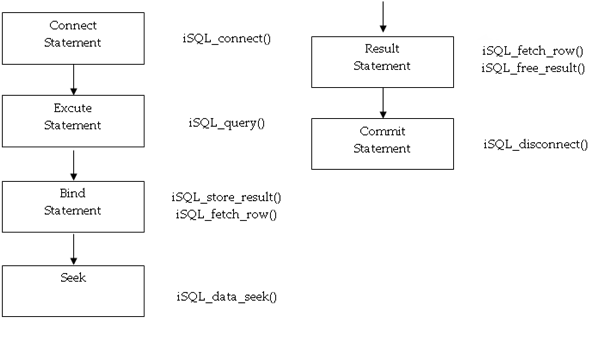
[그림 ] iSQL\_prepare를 이용한 Select

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Prepare Statement    1. iSQL\_prepare( ) : 질의문의 실행 전 준비과정을 수행    2. iSQL\_prepare\_result( ) : statement의 메타 정보를 결과 형태로 얻어오는 함수 3. Execute Statement : iSQL\_execute( ) 함수를 이용하여 질의 수행 4. Bind Statement : iSQL\_bind\_result( ) 함수는 해당 statement에 대한 결과 값을 바인딩 5. Fetch Statement : iSQL\_fetch( ) 함수는 statement로부터 현재 record 값을 바인딩된 변수로 fetch 6. Commit Statement : 검색 질의문에 의해 설정되었던 lock를 해제한다. 7. Close Statement    1. iSQL\_free\_result( ) : iSQL\_RES에 할당된 메모리를 제거    2. iSQL\_stmt\_close( ) : 생성된 statement를 close 8. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include <strings.h>**  **#include "isql.h"**  **char \*query = "select \* from mldb\_sample limit 10";**  **int main(int ac, char \*\*av)**  **{**  **iSQL isql;**  **iSQL\_BIND bind[3];**  **iSQL\_STMT \*stmt;**  **iSQL\_RES \*result;**  **char data[3][32];**  **int is\_null[3];**  **unsigned long length[3];**  **int i, ret;**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **if (!(stmt= iSQL\_prepare(&isql, query, strlen(query)))) {**  **printf("Prepare failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **if (!(result= iSQL\_prepare\_result(stmt))) {**  **printf("Prepare\_result failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto CLOSE\_AND\_DISCONNECT;**  **}**  **printf("Total fields: %u\n", iSQL\_num\_fields(result));**  **if (iSQL\_num\_fields(result) != 3) {**  **printf("Prepare returned invalid field count\n");**  **goto FREE\_AND\_DISCONNECT;**  **}**  **if (iSQL\_execute(stmt) < 0) {**  **printf("Execute failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto FREE\_AND\_DISCONNECT;**  **}**  **if (iSQL\_stmt\_store\_result(stmt) < 0) {**  **printf("Failed to store result: %s\n", iSQL\_stmt\_error(stmt));**  **goto FREE\_AND\_DISCONNECT;**  **}**  **for (i = 0; i < sizeof(bind)/sizeof(bind[0]); i++) {**  **bind[i].buffer\_type= SQL\_DATA\_CHAR;**  **bind[i].is\_null= &is\_null[i];**  **bind[i].length= &length[i];**  **bind[i].buffer = data[i];**  **bind[i].buffer\_length = sizeof(data[i]);**  **}**  **if (iSQL\_bind\_result(stmt, bind) < 0) {**  **printf("Bind\_result failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto FREE\_AND\_DISCONNECT;**  **}**  **while (1) {**  **ret = iSQL\_fetch(stmt);**  **if (ret == iSQL\_NO\_DATA)**  **break;**  **else if (ret < 0) {**  **printf("Fetch failed: %s\n", iSQL\_stmt\_error(stmt));**  **goto FREE\_AND\_DISCONNECT;**  **}**  **for (i = 0; i < sizeof(bind)/sizeof(bind[0]); i++) {**  **if (is\_null[i])**  **printf("[NULL]");**  **else**  **printf("[%s:%ld]", data[i], length[i]);**  **}**  **printf("\n");**  **}**  **iSQL\_commit(&isql);**  **FREE\_AND\_DISCONNECT:**  **iSQL\_free\_result(result);**  **CLOSE\_AND\_DISCONNECT:**  **if (iSQL\_stmt\_close(stmt) < 0) {**  **printf("Failed to free the statement handle: %s\n", iSQL\_stmt\_error(stmt));**  **return -1;**  **}**  **DISCONNECT:**  **iSQL\_rollback(&isql);**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |

|  |
| --- |
| **결과 Set을 이용하여 재 검색** |

|  |
| --- |
| Static SQL 함수를 이용하여 table에 저장된 record를 검색하고 cursor의 위치를 재설정한 후 다시 검색하는 예제이다. |



[그림 ] 결과 Set을 이용하여 재 검색

|  |
| --- |
| 1. Connect Statement : iSQL\_connect( ) 함수를 이용하여 OpenML (embedded DBMS)에 접속 2. Execute Statement : iSQL\_query( ) 함수를 이용하여 질의 수행    1. SELECT FROM : 하나 이상의 table에서 record 검색 3. Result Statement    1. iSQL\_store\_result( ) : SQL 질의문의 모든 결과값을 한꺼번에 지역 버퍼에 저장    2. iSQL\_fetch\_row( ) : SQL 질의의 결과에서 다음 row를 fetch 4. Seek Statement : iSQL\_data\_seek( ) 함수를 이용하여 커서를 이동하여 row의 위치를 재설정 5. Result Statement    1. iSQL\_store\_result( ) : SQL 질의문의 모든 결과값을 한꺼번에 지역 버퍼에 저장    2. iSQL\_fetch\_row( ) : SQL 질의의 결과에서 다음 row를 fetch    3. iSQL\_free\_result( ) : 할당된 메모리를 제거 6. Disconnect Statement : iSQL\_disconnect( ) 함수를 이용하여 OpenML와의 연결 해제 |

|  |  |
| --- | --- |
| **예제** | **#include <stdio.h>**  **#include "isql.h"**  **int main(void)**  **{**  **iSQL isql;**  **iSQL\_RES \*res;**  **iSQL\_ROW row;**  **int i, j, num\_fields;**  **long int offset;**  **if (!iSQL\_connect(&isql, "127.0.0.1", " mldb ", " mldb ", " mldb ")) {**  **printf("Connection failed: %s\n", iSQL\_error(&isql));**  **return -1;**  **}**  **if (iSQL\_query(&isql, "select \* from mldb\_sample limit 10")) {**  **printf("SELECT failed: %s\n", iSQL\_error(&isql));**  **goto DISCONNECT;**  **}**  **res = iSQL\_store\_result(&isql);**  **if (res == NULL)**  **printf("failed to store result: %s\n", iSQL\_error(&isql));**  **else {**  **num\_fields = iSQL\_num\_fields(res);**  **printf("Fetch the first 10 records:\n");**  **for (i = 0; (row = iSQL\_fetch\_row(res)) != NULL; i++) {**  **for (j = 0; j < num\_fields; j++)**  **printf("[%s]", row[j]);**  **printf("\n");**  **}**  **printf("%d record(s) was selected\n", i);**  **if (!iSQL\_eof(res))**  **printf("The last row was not read\n");**  **printf("\n");**    **printf("Go back, and then fetch records;\n");**  **if ((offset = iSQL\_data\_seek(res, -4, iSQL\_DATA\_SEEK\_CURRENT)) == iSQL\_NO\_DATA) {**  **printf("SCROLL failed: invalid offset\n");**  **goto FREE\_AND\_DISCONNECT;**  **}**  **else if (offset < 0) {**  **printf("SCROLL failed: %s\n", iSQL\_error(&isql));**  **goto FREE\_AND\_DISCONNECT;**  **}**  **for (i = 0; (row = iSQL\_fetch\_row(res)) != NULL; i++) {**  **for (j = 0; j < num\_fields; j++)**  **printf("[%s]", row[j]);**  **printf("\n");**  **}**  **printf("%d record(s) was selected\n", i);**  **if (!iSQL\_eof(res))**  **printf("The last row was not read\n");**  **}**  **FREE\_AND\_DISCONNECT:**  **iSQL\_free\_result(res);**  **DISCONNECT:**  **iSQL\_disconnect(&isql);**  **return 0;**  **}** |