**SQLConnect**

Altibase CLI와 DB와의 연결을 설정한다.

연결 핸들 상태, 트랜잭션 상태, 그리고 오류 정보를 포함한 DB 연결에 관한 모든 정보를 참조한다.

구 문

SQLRETURN SQLConnect (

SQLHDBC dbc,

SQLCHAR \* db,

SQLSMALLINT dbLength,

SQLCHAR \* usr,

SQLSMALLINT usrLength,

SQLCHAR \* pwd,

SQLSMALLINT pwdLength );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHDBC | dbc | 입력 | 연결 핸들 |
| SQLCHAR \* | db | 입력 | 호스트 IP |
| SQLSMALLINT | dbLength | 입력 | \*db의 길이 |
| SQLCHAR \* | usr | 입력 | 사용자 식별자 |
| SQLSMALLINT | usrLength | 입력 | \*usr의 길이 |
| SQLCHAR \* | pwd | 입력 | 인증 문자열 (암호) |
| SQLSMALLINT | pwdLength | 입력 | \*pwd의 길이 |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

입력 길이 인수(dbLength, usrLength, pwdLength)는 연관된 자료의 실제 길이로 설정될 수 있다. 여기에는 연관된 자료가 널로 종료된다는 것을 표시하기 위한 SQL\_NTS나 널 종료 문자가 포함되지 않는 길이 값을 설정할 수 있다.

\* 이 함수를 호출하기전에 SQLAllocConnect()를 호출해야 한다.

SQLAllocStmt()를 호출하기 전에 이 함수를 호출해야 한다.

서버 연결시 필요한 정보(인자로 넘기는 IP, 사용자명, 암호를 제외한)는 SQLSetConnectAttr()를 이용해 설정할 수 있다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 08001 | server에 연결할 수 없음 | Altibase CLI가 DB에 연결을 설정할 수 없음 |
| 08002 | 연결 이름이 사용 중 | 해당 dbc는 이미 DB에 연결 되 있음 |
| 08S01 | 통신 회선 장애 (데이터 송수신 실패) | Altibase CLI와 DB간에 함수 처리가 완료되기 전에 통신 회선 실패 |
| HY000 | 일반 오류 | Character set가 존재하지 않음 |
| HY001 | 메모리 할당 오류 | Altibase CLI가 함수를 실행하고 완료하기 위해 요구된 메모리를 할당할 수 없음 |

**SQLDriverConnect**

SQLDriverConnect()는 SQLConnect()를 대체할 수 있다.

이 함수는 SQLConnect()의 세 개의 인자 (DSN, 사용자 ID, 비밀 번호) 보다 더 많은 정보를 요구하는 연결 스트링을 지원한다.

SQLDriverConnect()는 다음과 같은 연결 속성을 제공한다.

호스트 IP 또는 호스트 이름, 하나 이상의 사용자 ID, 하나 이상의 암호, 연결 방법, 포트 번호, NLS\_USE, BATCH, TIMEOUT 설정 등의

연결 스트링을 사용하여 연결을 설정한다.

구 문

SQLRETURN SQLDriverConnect (

SQLHDBC dbc,

SQLPOINTER windowHandle,

SQLCHAR \* InConnectionString,

SQLSMALLINT length1,

SQLCHAR \* OutConnectionString,

SQLSMALLINT bufferLength,

SQLSMALLINT \* strLength2Ptr,

SQLSMALLINT DriverCompletion );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHDBC | dbc | 입력 | 연결 핸들 |
| SQLPOINTER | windowHandle | 입력 | 사용하지 않음 |
| SQLCHAR\* | InConnectionString | 입력 | 완전 연결 스트링  부분 연결 스트링  빈 문자열아래 설명 참조 |
| SQLSMALLINT | length1 | 입력 | \*InConnectionString의 길이 |
| SQLCHAR \* | OutConnectionString | 출력 | 사용하지 않음 |
| SQLSMALLINT | bufferLength | 입력 | 사용하지 않음 |
| SQLSMALLINT \* | strLength2Ptr | 출력 | 사용하지 않음 |
| SQLSMALLINT | DriverCompletion | 입력 | 사용하지 않음 |

결과값

SQL\_SUCCESS   
SQL\_SUCCESS\_WITH\_INFO   
SQL\_NO\_DATA\_FOUND  
SQL\_INVALID\_HANDLE   
SQL\_ERROR

설 명

이 연결 스트링은 연결을 완료하기 위해 필요한 하나 이상의 값을 전달하기 위해 사용된다.

연결 스트링의 내용과 DriverCompletion의 값은 연결 방법을 결정한다.

각 키워드는 다음과 같은 속성을 갖는다.

DSN : 호스트 IP 또는 호스트 이름

UID : 권한 부여 이름 (사용자 ID)

PWD : 권한 부여 이름에 대응하는 암호. 사용자 ID에 대한 암호가 없으면 아무것도 지정되지 않는다.

CONNTYPE : 연결 방법 (1 : TCP/IP, 2 : UNIX DOMAIN, 3 : IPC)

PORT\_NO : 연결 포트 번호

NLS\_USE : 사용언어 지정 (US7ASCII : 영어, KO16KSC5601 : 한국어)

BATCH : 일괄 처리 (batch processing) 모드로 연결을 생성

TIMEOUT : 서버 연결 시도 시 기다리는 시간. 기본값은 3초이다.

CONNECTION\_TIMEOUT : 네트워크 불안정 시 select() 또는 poll() 에서 발생할 수 있는 blocking을 방지하기 위한 속성이다.

DATE\_FORMAT : 날짜 형식. 기본 날짜 형식은 YYYY/MM/DD HH:MI:SS이다.

InConnectionString : DSN=192.168.1.11;UID=SYS;PWD=MANAGER;CONNTYPE=1;NLS\_USE=KO16KSC5601;PORT\_NO=20202;BATCH=ON;TIMEOUT=5;CONNECTION\_TIMEOUT=10

\* Batch Processing

연결이 생성된 후에 연결 속성을 변경하여 batch processing을 작동시키거나 정지 시킬 수 있다.

SQLSetConnectAttr(dbc, SQL\_ATTR\_BATCH, (void\*)SQL\_BATCH\_ON, 0);

SQLSetConnectAttr(dbc, SQL\_ATTR\_BATCH, (void\*)SQL\_BATCH\_OFF, 0);

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 08001 | server에 연결할 수 없음 | Altibase CLI가 DB에 연결을 설정할 수 없음 |
| 08002 | 연결 이름이 사용 중 | 해당 dbc는 이미 DB에 연결 되 있음 |
| 08S01 | 통신 회선 장애 (데이터 송수신 실패) | Altibase CLI와 DB간에 함수 처리가 완료되기 전에 통신 회선 실패 |
| HY000 | 일반 오류 | Character set가 존재하지 않음 |
| HY001 | 메모리 할당 오류 | Altibase CLI가 함수를 실행하고 완료하기 위해 요구된 메모리를 할당할 수 없음 |

관련함수

[SQLAllocHandle](http://data.altibase.com/pdf/a3/htm/cli/ch03s03.html)

[SQLConnect](http://data.altibase.com/pdf/a3/htm/cli/ch03s09.html)

[SQLDisconnect](http://data.altibase.com/pdf/a3/htm/cli/ch03s12.html)

[SQLFreeHandle](http://data.altibase.com/pdf/a3/htm/cli/ch03s22.html)

[SQLSetConnectAttr](http://data.altibase.com/pdf/a3/htm/cli/ch03s34.html)

sprintf(connStr, "DSN=127.0.0.1;UID=%s;PWD=%s;CONNTYPE=%d;NLS\_USE=%s", /\* ;PORT\_NO=20300", \*/

USERNAME, PASSWD, 2, NLS);

/\* Connection을 형성 \*/

if (SQLDriverConnect( dbc, NULL, connStr, SQL\_NTS,

NULL, 0, NULL,

SQL\_DRIVER\_NOPROMPT ) != SQL\_SUCCESS)

{

execute\_err(dbc, SQL\_NULL\_HSTMT, "SQLDriverConnect");

return SQL\_ERROR;

}

**SQLExecDirect**

만약 SQL 문에 매개변수가 존재한다면 매개변수 마커들의 현재 값을 사용하여 주어진 SQL 문장을 직접 실행한다.

SQL 문은 한 번만 실행될 수 있으며 DB로 보내는 가장 빠른 방법이다.

구 문

SQLRETURN SQLExecDirect (

SQLHSTMT stmt,

SQLCHAR \* sql,

SQLINTEGER sqlLength );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHSTMT | stmt | 입력 | 명령문 핸들 |
| SQLCHAR \* | sql | 입력 | 실행 될 SQL 문 |
| SQLINTEGER | sqlLength | 입력 | \*sql의 길이 |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_NO\_DATA\_FOUND

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

SQL문 스트링에는 매개변수 마커가 포함될 수 있다. 매개변수 마커는 "?" 문자로 표시되며, SQLExecDirect()가 호출될 때 애플리케이션 변수 값이 대체되어야 할 명령문의 위치를 지정한다. SQLBindParameter()는 애플리케이션 변수를 각 매개변수 마커에 바인드(연관)시켜서, 자료가 전송될 때 자료 변환이 수행되어야 하는지를 표시한다. 모든 매개변수는 SQLExecDirect()를 호출하기 전에 바인드되어야 한다.

SQL 문이 SELECT 문일 때, 서버로부터 받은 결과 집합에서 행을 검색하려면, SQLExecDirect()가 성공적으로 리턴된 후 SQLBindCol() 에 의해 버퍼를 바인딩한 후, SQLFetch()를 호출하여서, 바인딩 된 버퍼를 참조하면 된다.

SQL문이 DELETE 또는 UPDATE 문일 경우, 그 결과 영향을 받은 레코드 수를 검색하려면, SQLRowCount()를 사용하면 된다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 08003 |  | stmt가 연결 되지 않은 상태거나 연결이 끊어진 상태 |
| 08S01 | 통신 장애 | 데이터 송수신 실패 |
| HY000 | 일반 오류 |  |
| HY001 | 메모리 할당 오류 | 명시된 핸들에 대한 메모리 할당 실패 |
| HY009 | 유효하지 않은 인자 (null pointer) 사용 | \*sql이 null pointer 임 |

관련함수

[SQLBindCol](http://data.altibase.com/pdf/a3/htm/cli/ch03s05.html)

[SQLEndTran](http://data.altibase.com/pdf/a3/htm/cli/ch03s14.html)

[SQLExecute](http://data.altibase.com/pdf/a3/htm/cli/ch03s17.html)

[SQLFetch](http://data.altibase.com/pdf/a3/htm/cli/ch03s18.html)

[SQLGetData](http://data.altibase.com/pdf/a3/htm/cli/ch03s25.html)

[SQLPrepare](http://data.altibase.com/pdf/a3/htm/cli/ch03s29.html)

[SQLSetStmtAttr](http://data.altibase.com/pdf/a3/htm/cli/ch03s36.html)

예 제

sprintf(query,"SELECT \* FROM DEMO\_EX1");

if (SQLExecDirect(stmt,query, SQL\_NTS) != SQL\_SUCCESS)

{

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

**SQLBindCol**

모든 데이터 타입에 대해 애플리케이션 변수들을 (application data buffers) 결과 집합의 열들에 바인드한다.

SQLBindCol binds application data buffers to columns in the result set.

구 문

SQLRETURN SQLBindCol (

SQLHSTMT stmt,

SQLSMALLINT col,

SQLSMALLINT cType,

SQLPOINTER value,

SQLINTEGER max,

SQLINTEGER \* valueLength );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHSTMT | stmt | 입력 | 명령문 핸들 |
| SQLSMALLINT | col | 입력 | 바인드할 결과 집합에서의 column의 순서. 1부터 시작 |
| SQLSMALLINT | cType | 입/출력(유예중) | \*value 버퍼의 C 데이터 타입 식별자 (부록 참고). SQLFetch()를 이용하여 DB에서 데이터를 검색할 때 Altibase CLI는 데이터를 이 타입으로 변환한다. |
| SQLPOINTER | value | 출력 | 데이터를 저장할 버퍼의 포인터.  SQLFetch()는 이 버퍼에 데이터를 반환한다.value가 null pointer 면 Altibase CLI는 결과 집합 column에 대한 data buffer를 언바인드한다. 애플리케이션은 SQL\_UNBIND option을 이용해 SQLFreeStmt()를 호출함으로써 모든 column들을 언바인드한다. 그러나 value 인자가 null pointer 일지라도 valueLength 인자가 유효한 값이면 애플리케이션은 여전히 length 버퍼 바운드를 갖는다. |
| SQLINTEGER | max | 입력 | 버퍼의 최대 크기 (byte 단위). \*value에 character data를 반환할 때 \*value 인자는 null-termination character를 위한 공간을 포함해야만 한다. 그렇지 않으면, Altibase CLI는 데이터를 자른다. 고정 길이 data (integer 또는 data structure 등)를 반환할 경우는 CLI가 max를 무시하기 때문에 충분한 버퍼 크기를 할당해야만 한다. 그렇지 않으면 CLI는 버퍼의 끝을 지나 저장한다. |
| SQLINTEGER \* | valueLength | 입/출력(유예중) | NULL이거나 버퍼에 저장된 데이터의 길이를 저장할 포인터SQLFetch()는 이 버퍼에 다음 값들을 반환할 수 있다:데이터 길이, SQL\_NULL\_DATA |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

이 함수에 대해 포인터 value와 valueLength는 유예중인 출력 변수로 (SQLFetch()가 호출될 때까지 이 포인터들이 가리키는 기억장치 위치가 갱신되지 않음), 이 포인터들에 의해 참조된 위치는 SQLFetch()가 호출될 때까지 유효한 상태로 남아 있어야 한다.

SQLBindCol()은 모든 데이터 타입에 대해 애플리케이션 변수들을 (기억장치버퍼) 결과 집합의 열들에 바인드한다. SQLFetch()가 호출되면 데이터가 DBMS에서 애플리케이션으로 전송된다.

애플리케이션은 결과 집합의 각 열에 대해 한 번씩 SQLBindCol()을 호출한다. SQLFetch()가 호출되면, 각 바인드된 열의 데이터는 value나 valueLength 포인터에 의해서 할당된 위치에 저장된다.

애플리케이션은 SQLDescribeCol()이나 또는 SQLColAttribute()를 호출함으로서 열의 데이터 타입이나 길이 등과 같은 속성들을 질의할 수 있다. 이러한 정보는 정확한 데이터 타입을 명시하거나 데이터를 다른 데이터 타입으로 변환하는데 사용할 수 있다.

열들은 1 부터 시작해서 왼쪽부터 오른쪽으로 연속적인 번호에 의해 식별된다. 결과 집합의 열의 개수는 SQLNumResultCols()나 또는 fieldIdentifier 인자에 SQL\_DESC\_COUNT를 설정해서 SQLColAttribute()를 호출함으로서 결정될 수 있다.

애플리케이션은 모든 열을 바인드 하지 않을 수도 있고, 어떠한 열과도 바인드 하지 않을 수 있다. 언바인드된 열의 데이터는 (언바인드된 열들만) SQLFetch()가 호출된 후에 SQLGetData()를 사용해서 검색될 수 있다. SQLBindCol()은 SQLGetData() 보다 더 효율적이다.

\* 이 호출에 의해 지정된 버퍼에 자료를 얻기 위해, SQLFetch() 전에 SQLBindCol()을 호출해야 한다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 07009 | 유효하지 않은 결과 집합에서의 column 번호 | col 인수의 값이 결과 집합에서의 최대 column 수를 초과함 |
| HY000 | 일반 오류 |  |
| HY001 | 메모리 할당 오류 | 명시된 핸들에 대한 메모리 할당 실패 |
| HY003 | Application buffer type이 유효하지 않음 | cType 인자의 값이 유효하지 않음 |

관련함수

[SQLDescribeCol](http://data.altibase.com/pdf/a3/htm/cli/ch03s10.html)

[SQLFetch](http://data.altibase.com/pdf/a3/htm/cli/ch03s18.html)

[SQLFreeStmt](http://data.altibase.com/pdf/a3/htm/cli/ch03s23.html)

[SQLGetData](http://data.altibase.com/pdf/a3/htm/cli/ch03s25.html)

[SQLNumResultCols](http://data.altibase.com/pdf/a3/htm/cli/ch03s28.html)

sprintf(query,"SELECT id,name,age FROM DEMO\_EX2 WHERE id=?");

if (SQLPrepare(stmt,query, SQL\_NTS) != SQL\_SUCCESS)

{

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

if (SQLBindParameter(stmt, 1, SQL\_PARAM\_INPUT,

SQL\_C\_CHAR, SQL\_CHAR,

8, 0,

id\_in, sizeof(id\_in), NULL) != SQL\_SUCCESS)

{

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

if (SQLBindCol(stmt, 1, SQL\_C\_CHAR,

id, sizeof(id), NULL) != SQL\_SUCCESS)

{

printf("SQLBindCol error!!!\n");

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

if (SQLBindCol(stmt, 2, SQL\_C\_CHAR,

name, sizeof(name), NULL) != SQL\_SUCCESS)

{

printf("SQLBindCol error!!!\n");

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

if (SQLBindCol(stmt, 3, SQL\_C\_SLONG,

&age, 0, NULL) != SQL\_SUCCESS)

{

printf("SQLBindCol error!!!\n");

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

printf("id\tName\tAge\tbirth\t\sex\tetc\n");

printf("=====================================================================\n");

for ( i=1; i<=3; i++ )

{

sprintf(id\_in, "%d0000000", i);

if ( SQLExecute(stmt) != SQL\_SUCCESS )

{

execute\_err(dbc, stmt, "SQLExecute : ");

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

if ( (rc = SQLFetch(stmt)) != SQL\_NO\_DATA && rc == SQL\_SUCCESS )

{

printf("%-10s%-20s%-5d%4d/%02d/%02d %02d:%02d:%02d\t%-2d\t",

id, name, age, birth.year, birth.month, birth.day,

birth.hour, birth.minute, birth.second, sex);

if (etc\_ind == SQL\_NULL\_DATA)

{

printf("NULL\n");

}

else

{

printf("%.3f\n", etc);

}

}

else

{

execute\_err(dbc, stmt, query);

break;

}

}

**SQLSetConnectAttr**

특정 연결에 대한 연결 속성값을 설정한다.

구 문

SQLRETURN SQLSetConnectAttr (

SQLHDBC dbc,

SQLINTEGER Attribute,

SQLPOINTER ValuePtr,

SQLINTEGER StringLength );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHDBC | dbc | 입력 | 연결 핸들 |
| SQLINTEGER | Attribute | 입력 | 설정할 속성 |
| SQLPOINTER | ValuePtr | 입력 | 속성에 해당하는 값의 포인터  Attribute 값에 따라, ValuePtr은 32-bit unsigned integer 값이거나, null-terminated character string을 가리키는 포인터. 현재 아래에 설명된 속성값만 지원한다.  SQL\_ATTR\_AUTOCOMMIT  SQL\_ATTR\_BATCH  SQL\_ATTR\_CONNECTION\_TIMEOUT  SQL\_ATTR\_IDN\_LANG  SQL\_ATTR\_PORT  SQL\_ATTR\_TXN\_ISOLATION |
| SQLINTEGER | StringLength | 입력 | ValuePtr이 character string 일 때 StringLength는 문자열의 길이  또는 SQL\_NTS. 32-bit Integer 일 때는 무시한다. |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

애플리케이션은 연결이 할당되거나 해제되는 어느 시점에서나 SQLSetConnectAttr()을 호출할 수 있다. 애플리케이션에서 성공적으로 설정된 모든 연결과 명령문 속성들은 현재 연결에서 SQLFreeHandle()이 호출될 때까지 보존된다.

연결 속성

| Attribute | 내용 |
| --- | --- |
| SQL\_ATTR\_AUTOCOMMIT | 연결에 대한 반영 작동을 설정하는 32비트 값.  SQL\_AUTOCOMMIT\_ON: 각각의 SQL문은 실행 시 자동으로 반영.  SQL\_AUTOCOMMIT\_OFF: 각 SQL문을 실행 시 자동으로 반영하지 않는다. |
| SQL\_ATTR\_BATCH | Commit 되지 않은 INSERT, UPDATE, DELETE 문은 같은 트랜잭션에서만 읽을 수 있다는 점을 이용하여 SELECT나 Commit이 발생할 때까지 서버로의 전송을 지연  SQL\_BATCH\_ON: 배치 모드를 활성화  SQL\_BATCH\_OFF: 배치 모드를 비활성화 |
| SQL\_ATTR\_CONNECTION\_TIMEOUT | 네트워크 불안정 시 select() 또는 poll() 에서 발생할 수 있는 blocking을  방지하기 위한 timeout 값 설정 |
| SQL\_ATTR\_IDN\_LANG | Null-terminated 문자열 |
| SQL\_ATTR\_PORT | Server port 설정 (32-bit Integer) |
| SQL\_ATTR\_TXN\_ISOLATION | dbc에 의해 참조되는 현재 연결에 대한 트랜잭션 분리 레벨을 설정하는 32 비트값. |
| ALTIBASE\_DATE\_FORMAT | 날짜 형식 설정. 기본값으로 YYYY/MM/DD HH:MI:SS을 지원한다. |

\* 일괄 처리 (Batch Processing)

연결 속성을 변경하여 일괄 처리를 작동시키거나 정지 시킬 수 있다.

(e.g.) SQLSetConnectAttr(dbc, SQL\_ATTR\_BATCH, (void\*)SQL\_BATCH\_ON, 0);

SQLSetConnectAttr(dbc, SQL\_ATTR\_BATCH, (void\*)SQL\_BATCH\_OFF, 0);

아래 그림은 batch processing을 간략하게 도식화 한 것이다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 08S01 | 통신 회선 장애 (데이터 송수신 실패) | Altibase CLI와 DB간에 함수 처리가 완료되기 전에 통신 회선 실패 |
| HY000 | 일반 오류 |  |

Attribute가 명령문 속성일 때, SQLSetConnectAttr()은 SQLSetStmtAttr()에 의해 반환된 어떠한 SQLSTATE도 반환할 수 있다.

관련함수

[SQLAllocHandle](http://data.altibase.com/pdf/a3/htm/cli/ch03s03.html)

예 제

< $ALTIBASE\_HOME/sample/SQLCLI/demo\_tran1.cpp 참고 >

if (SQLSetConnectAttr(dbc, SQL\_ATTR\_AUTOCOMMIT,

(void\*)SQL\_AUTOCOMMIT\_OFF, 0) != SQL\_SUCCESS)

{

execute\_err(dbc, SQL\_NULL\_HSTMT, "Autocommit OFF");

return SQL\_ERROR;

}

**SQLAllocHandle**

환경, 연결, 그리고 명령문 핸들을 위한 메모리를 할당하고 초기화 한다.

구 문

SQLRETURN SQLAllocHandle (

SQLSMALLINT HandleType,

SQLHANDLE InputHandle,

SQLHANDLE \* OutputHandlePtr );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQL  SMALLINT | Handle  Type | 입력 | 할당 될 핸들 타입으로 다음 세 값 중에 하나이다. SQL\_HANDLE\_ENV,  SQL\_HANDLE\_DBC, SQL\_HANDLE\_STMT |
| SQL  HANDLE | Input  Handle | 입력 | 입력 핸들 HandleType이 SQL\_HANDLE\_ENV이면 InputHandle은  SQL\_NULL\_HANDLE이고 SQL\_HANDLE\_DBC이면 환경 핸들 이어야 하고  SQL\_HANDLE\_STMT이면 연결 핸들 이어야 한다. |
| SQL  HANDLE \* | Output  Handle  Ptr | 출력 | 할당 받은 핸들의 포인터 |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

SQLAllocHandle()은 다음 단락에서 묘사될 환경, 연결, 그리고 명령문 핸들을 할당한다.

이 함수는 SQLAllocEnv(), SQLAllocConnect() 및 SQLAllocStmt() 함수들을 대치한다.

환경 핸들을 요구하기 위하여 애플리케이션은 HandleType이 SQL\_HANDLE\_ENV이고, InputHandle이 SQL\_NULL\_HANDLE인 SQLAllocHandle()을 호출한다.

연결 핸들을 요구 하기 위하여 애플리케이션은 HandleType이 SQL\_HANDLE\_DBC인 SQLAllocHandle()을 호출하고, 그때의 InputHandle은 유효한 환경 핸들이어야 한다.

명령문 핸들을 요구하기 위하여 애플리케이션은 HandleType이 SQL\_HANDLE\_STMT인 SQLAllocHandle()을 호출하고, 그때의 InputHandle은 유효한 연결 핸들이어야 한다.

한 애플리케이션에 의해 한 번에 한 개 이상의 환경, 연결, 그리고 명령문 핸들들이 할당 될 수 있는데, 한 프로세스 내의 다른 쓰레드 상에서 같은 환경, 연결, 또는 명령문 핸들은 동시에 사용할 수 없다.

환경 핸들 할당

환경 핸들은 연결 핸들이 유효하다든지 활성화 되 있다든지 하는 전역적인 정보를 제공한다.

환경 핸들을 요구하기위해 애플리케이션은 Handletype이 SQL\_HANDLE\_ENV이고 InputHandle이 SQL\_NULL\_HANDLE인 SQLAllocHandle()을 호출한다. Altibase CLI는 환경 정보에 필요한 메모리를 할당하고, \*OutputHandle에 관련 핸들의 값을 돌려준다. 애플리케이션은 \*OutputHandle 값을 환경 핸들을 필요로하는 후속 호출들에 전달한다.

연결 핸들 할당

연결 핸들은 연결 핸들상에서 명령문 핸들이 유효한가, 트랜잭션이 현재 열려있는가와 같은 정보를 제공한다.

연결 핸들을 요구하기위해, 애플리케이션은 HandleType이 SQL\_HANDLE\_DBC인 SQLAllocHandle()을 호출한다. InputHandle 인자는 SQLAllocHandle()을 호출함으로서 반환된 환경 핸들로 설정된다. Altibase CLI는 연결 정보에 필요한 메모리를 할당하고, \*OutputHandle에 관련 핸들의 값을 돌려준다. 애플리케이션은 \*OutputHandle 값을 연결 핸들을 필요로하는 후속 호출들에 전달한다.

명령문 핸들 할당

명령문 핸들은 SQL문 처리에 대한 오류 메시지 그리고 상태 정보등과 같은 명령문 정보를 제공한다.

명령문 핸들을 요구하기위해, 애플리케이션은 DB와 연결하고, SQL 문을 보내기 전에 SQLAllocHandle()을 호출한다. 이러한 호출에서 HandleType은 SQL\_HANDLE\_STMT로 설정되야만 하고 InputHandle 인자는 SQLAllocHandle()을 호출함으로서 반환된 연결 핸들로 설정되야만 한다. Altibase CLI는 명령문 정보에 필요한 메모리를 할당하고, 명시된 연결로 명령문 핸들과 연결하고, \*OutputHandle에 관련 핸들의 값을 돌려준다. 애플리케이션은 \*OutputHandle 값을 명령문 핸들을 필요로하는 후속 호출들에 전달한다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| HY000 | 일반 오류 |  |
| HY001 | 메모리 할당 오류 | 명시된 핸들을 위한 메모리 할당에 실패 함. |
| HY009 | 유효하지 않은 인자 (null pointer) 사용 | OutputHandlePtr가 null pointer 임. |

관련함수

[SQLExecDirect](http://data.altibase.com/pdf/a3/htm/cli/ch03s16.html)

[SQLExecute](http://data.altibase.com/pdf/a3/htm/cli/ch03s16.html)

[SQLFreeHandle](http://data.altibase.com/pdf/a3/htm/cli/ch03s22.html)

[SQLPrepare](http://data.altibase.com/pdf/a3/htm/cli/ch03s29.html)

[SQLSetConnectAttr](http://data.altibase.com/pdf/a3/htm/cli/ch03s34.html)

[SQLSetEnvAttr](http://data.altibase.com/pdf/a3/htm/cli/ch03s35.html)

[SQLSetStmtAttr](http://data.altibase.com/pdf/a3/htm/cli/ch03s36.html)

예 제

< $ALTIBASE\_HOME/sample/SQLCLI/demo\_meta1.cpp 참고 >

if (SQLAllocHandle(SQL\_HANDLE\_ENV, SQL\_NULL\_HENV, &env) != SQL\_SUCCESS)

{

printf("SQLAllocEnv error!!\n");

return SQL\_ERROR;

}

if (SQLAllocHandle(SQL\_HANDLE\_DBC, env, &dbc) != SQL\_SUCCESS)

{

printf("SQLAllocConnect error!!\n");

return SQL\_ERROR;

}

**SQLRowCount**

UPDATE, INSERT 또는 DELETE 문에 의해 영향을 받은 행들의 수를 반환한다.

구 문

SQLRETURN SQLRowCount (

SQLHSTMT stmt,

SQLINTEGER \* row );

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHSTMT | stmt | 입력 | 명령문 핸들 |
| SQLINTEGER \* | row | 출력 | 행의 개수를 저장할 포인터 |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

설 명

입력 명령문 핸들에 의해 참조된 마지막으로 실행 된 명령문이 UPDATE, INSERT 또는 DELETE문이 아니거나 성공적으로 실행되지 않았을 경우 이 함수는 \*row의 값을 -1로 설정한다.

그러나 행들을 수정, 삭제하거나 또는 삽입을 수행한 후 영향받은 행이 없거나 SELECT 문을 수행했을 때는 \*row의 값을 0으로 설정한다.

Cascading delete operation (부모 테이블에 임의의 행이 삭제되면 그것과 관련된 자식 테이블의 모든 행도 삭제됨)과 같은 SQL문에 의해 영향을 받는 임의의 테이블의 행들은 개수에 포함되지 않는다.

이 함수를 호출하기전에 SQLExecute() 또는 SQLExecDirect()를 호출해야만 한다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| HY000 | 일반 오류 |  |

관련함수

[SQLExecDirect](http://data.altibase.com/pdf/a3/htm/cli/ch03s16.html)

[SQLExecute](http://data.altibase.com/pdf/a3/htm/cli/ch03s17.html)

if (SQLExecute(stmt) != SQL\_SUCCESS)

{

execute\_err(dbc, stmt, query);

SQLFreeStmt(stmt, SQL\_DROP);

return SQL\_ERROR;

}

SQLRowCount(stmt, &row\_count);

**SQLFetch**

결과 집합의 다음 행으로 커서를 진행시키고 바인드된 열을 가져온다.

SQLFetch()를 사용하여 사용자가 SQLBindCol()에 명시된 변수들에 데이터를 직접 받거나 SQLGetData()를 호출하여 페치 후에 열을 개별적으로 수신할 수 있다. 열이 바인드될 때 변환이 지정된 경우 SQLFetch()가 호출될 때 데이터 변환도 발생 한다.

구 문

SQLRETURN SQLFetch (

SQLHSTMT stmt);

인 자

| 자료유형 | 인자 | 사용 | 설명 |
| --- | --- | --- | --- |
| SQLHSTMT | stmt | 입력 | 명령문 핸들 |

결과값

SQL\_SUCCESS

SQL\_SUCCESS\_WITH\_INFO

SQL\_INVALID\_HANDLE

SQL\_ERROR

SQL\_NO\_DATA\_FOUND

설 명

stmt에서 가장 최근에 실행된 명령문이 SELECT 문인 경우에만 호출될 수 있다.

SQLBindCol()과 바인드된 애플리케이션 변수 개수는 결과 집합의 열 수를 초과해서는 안된다. 그렇지 않으면 SQLFetch()가 실패한다.

커서의 위치

결과 집합이 생성될 때, 커서는 결과 집합의 시작 전에 위치한다. SQLFetch()는 결과 집합에서 바로 다음 행집합(rowset)을 반환한다. 예를 들어, 결과 집합이 전체 100개의 행들을 가지고 있을 때, 행집합의 크기가 5이면, 최초 SQLFetch()의 결과 행집합은 1 부터 5이다. 또한, 현재 행집합 52행부터 56행에 대한 결과 집합이라면, 그 다음 SQLFetch()에 의해 반환되는 새로운 행집합은 57행부터 61행, 반환 코드는 SQL\_SUCCESS, 그리고 인출된 행들의 개수는 5개가 될것이다. 다음 테이블은 행집합을 보여주고 SQLFetch()에서 반환된 코드를 반환한다.

| 현재행집합 | 반환 코드 | 새로운 행집합 | 페치된 행의 수 |
| --- | --- | --- | --- |
| 시작 전 | SQL\_SUCCESS | 1 to 5 | 5 |
| 1 to 5 | SQL\_SUCCESS | 6 to 10 | 5 |
| 52 to 56 | SQL\_SUCCESS | 57 to 61 | 5 |
| 91 to 95 | SQL\_SUCCESS | 96 to 100 | 5 |
| 93 to 97 | SQL\_SUCCESS | 98 to 100. 행 4와 5에는 SQL\_ROW\_NOROW이 설정됨 | 3 |
| 96 to 100 | SQL\_NO\_DATA | 없음 | 0 |
| 99 to 100 | SQL\_NO\_DATA | 없음 | 0 |
| After end | SQL\_NO\_DATA | 없음 | 0 |

SQLFetch()는 반환 후에, 현재 행은 행집합의 첫 번째 행이다.

바인드된 열에 데이터 반환

SQLFecth()는 각 행을 반환하면서 각 바인드된 열에 데이터를 위치시킨다. SQLBindCol()이 호출되었지만 바인드된 열이 없으면, SQLFetch()는 애플리케이션에 자료를 리턴하지는 않지만 커서를 진행시킨다. 이 경우 SQLFetch()가 다음 행으로 커서를 진행시키면 바인드되지 않은 열의 자료가 삭제 된다.

각 바인드된 열에 대해 SQLFetch()는 다음을 한다.

데이터가 NULL이면 길이/지시자 버퍼에 SQL\_NULL\_DATA를 설정하고 다음 column으로 진행한다.

SQLBindCol()의 type 인자에 명시된 타입의 데이터를 변환한다.

만약 데이터가 가변 길이 데이터 형으로 변환됐다면, SQLFetch()는 데이터 길이가 (SQL\_C\_CHAR로 변환 될 경우 null-termination 문자 포함) 데이터 버퍼 길이를 초과했는지를 검사한다. 만약 문자 데이터의 길이가 데이터 버퍼의 길이를 초과하면, SQLFetch()는 null-termination 문자 길이를 뺀 데이터 버퍼 길이에 맞춰 데이터를 자른다. 그렇게해서 null 문자로 데이터를 마무리한다. 만약 이진 데이터 길이가 데이터 버퍼 길이를 초과하면 SQLFetch()는 데이터 버퍼에 맞춰 데이터를 자른다. 데이터 버퍼의 길이는 SQLBindCol()에 length에 명시된다.

데이터 버퍼에 변환된 데이터를 위치시킨다.

길이/지시자 버퍼에 데이터 길이를 위치시킨다. 만약 지시자 포인터와 길이 포인터 둘다 같은 버퍼로 설정됐다면 유효한 데이터에 대해서 버퍼에는 길이가 기록되고 NULL 데이터에 대해서는 SQL\_NULL\_DATA가 기록된다. 만약 바운드 된 길이/지시자 버퍼가 없다면 SQLFetch()는 길이를 반환하지 않는다.

바인드된 데이터 버퍼와 길이/지시자 버퍼의 내용들은 만약 SQLFetch()가 SQL\_SUCCESS나 또는 SQL\_SUCCESS\_WITH\_INFO를 반환하지 않으면 정의되지 않는다. 즉, SQLFetch()의 결과가 SQL\_ERROR 일 경우 유효하지 않은 값이다.

행 상태 배열 (Row Status Array)

행 상태 배열에는 행집합에서의 각 열의 상태가 반환된다. 이러한 배열의 주소는 SQL\_ATTR\_ROW\_STATUS\_PTR 명령문 속성으로서 명시된다. 애플리케이션은 이 배열을 할당하고 배열의 크기는 SQL\_ATTR\_ROW\_ARRAY\_SIZE 명령문 속성에 의해 명시된 원소들의 수이다. 배열의 값들은 SQLFetch()에 의해 설정된다. 만약 SQL\_ATTR\_ROW\_STATUS\_PTR 명령문 속성의 값이 널 포인터이면, SQLFetch()는 행 상태를 반환하지 않는다.

Rows Fetched Buffer

Rows fetched buffer에는 페치된 행들의 수 (행들이 페치되는 동안 발생한 오류 때문에 어떤 데이터도 반환되지 않은 행들까지도 포함) 가 반환된다. 이 버퍼의 주소는 SQL\_ATTR\_ROWS\_FETCHED\_PTR 명령문 속성으로서 명시된다. 이 버퍼는 애플리케이션에 의해 할당되고 SQLFetch()에 의해 설정된다. 만일 SQL\_ATTR\_ROWS\_FETCHED\_PTR 명령문 속성이 널 포인터이면, SQLFetch()는 페치된 행들의 수를 반환하지 않는다.

만약 SQLFetch()가 SQL\_NO\_DATA가 반환될 때를 빼고 (이런 경우 row fetched buffer 값은 0으로 설정됨)

SQL\_SUCCESS나 또는 SQL\_SUCCESS\_WITH\_INFO를 반환하지 않으면 rows fetched buffer의 내용들은 정의되지 않는다.

오류 처리

오류나 경고등의 처리는 행들에게 개별적으로나 또는 모든 함수에 적용할 수 있다.

함수에 대한 오류 또는 경고들

만일 임의의 경고가 모든 함수에 적용된다면 SQLFetch()는 SQL\_SUCCESS\_WITH\_INFO와 적절한 SQLSTATE를 반환한다. 함수에 적용되는 경고 상태 레코드들은 상태 레코드들이 행들에 개별적으로 적용되기 전에 반환된다.

진 단

| SQLSTATE | 설명 | 부연설명 |
| --- | --- | --- |
| 01004 | 자료가 잘림 | 반환될 값의 크기가 주어진 버퍼의 크기보다 큼 |
| 07006 | 제한된 데이터 타입 속성 위반 | 결과 집합 내의 열의 데이터 값은 SQLBindCol()의 cType에서 명시된 데이터 타입으로 변환될 수 없다. |
| 08S01 | 통신 회선 장애 (데이터 송수신 실패) | Altibase CLI와 DB간에 함수 처리가 완료되기 전에 통신 회선 실패 |
| HY000 | 일반 오류 |  |

관련함수

[SQLBindCol](http://data.altibase.com/pdf/a3/htm/cli/ch03s05.html)

[SQLDescribeCol](http://data.altibase.com/pdf/a3/htm/cli/ch03s10.html)

[SQLExecDirect](http://data.altibase.com/pdf/a3/htm/cli/ch03s16.html)

[SQLExecute](http://data.altibase.com/pdf/a3/htm/cli/ch03s17.html)

[SQLFreeStmt](http://data.altibase.com/pdf/a3/htm/cli/ch03s23.html)

[SQLGetData](http://data.altibase.com/pdf/a3/htm/cli/ch03s25.html)

[SQLNumResultCols](http://data.altibase.com/pdf/a3/htm/cli/ch03s28.html)

[SQLPrepare](http://data.altibase.com/pdf/a3/htm/cli/ch03s29.html)