Database

35기 이찬규

목차

[1. ODBC API 2](#_Toc535920701)

[2. TSQL 2](#_Toc535920702)

[3. DB 프로그램 분석 3](#_Toc535920703)

# ODBC API

## SQLAllocHandle

SQLAllocHandle은 SQL을 사용하는데 필요한 장치들을 초기화하는 함수이다. 인자값으로 들어가는 HandleType에는 SQL\_HANDLE\_ENV, SQL\_HANDLE\_DBC, SQL\_HANDLE\_STMT 등이 사용되고 초기화를 원하는 자료형의 타입에 맞게 넣어준다. InputHandle은 할당을 해 줄 핸들을 입력한다.

OutputHandle에는 할당을 원하는 데이터의 포인터를 넣어주면 된다.

## SQLExecDirect

SQL을 실행하는 함수로 인자값에는 SQLHSTMT StatementHandle, SQLCHAR \* StatementText, SQLINTEGER TextLength 이 들어간다.

StatementHandle에는 핸들을 넣어주고, StatementText에는 실행을 할 SQL문장을 넣어주면 된다.

TextLength는 StatementText의 크기를 넣어주면 SQL을 StatementText에 해당하는 SQL이 실행이된다.

## SQLBindParameter

실행을 할 SQL 함수가 반환형이 있거나 인자값을 요구하는 경우 프로그램에서 바인딩을 해줘야 하는데 SQLBindParameter가 이 역할을 한다. 함수의 인자값은 다음과 같다.

SQLHSTMT StatementHandle,

SQLUSMALLINT ParameterNumber,

SQLSMALLINT InputOutputType,

SQLSMALLINT ValueType,

SQLSMALLINT ParameterType,

SQLULEN ColumnSize,

SQLSMALLINT DecimalDigits,

SQLPOINTER ParameterValuePtr,

SQLLEN BufferLength,

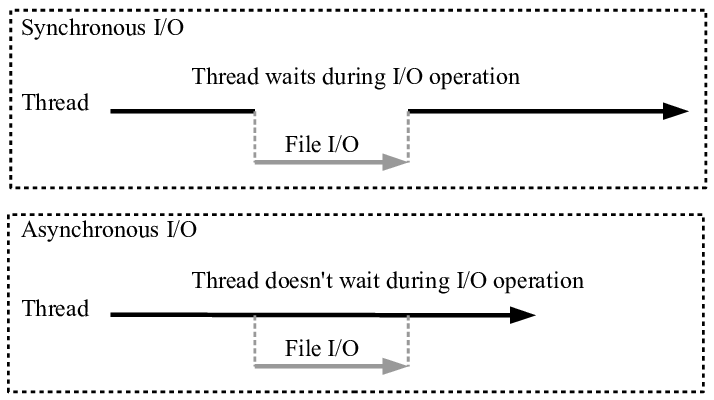
SQLLEN \* StrLen\_or\_IndPtr

StatementHandle에는 입력을 할 핸들을 넣어주고, ParameterNumber 1부터 시작하여서 1씩 증가시켜서 사용한다. InputOutputType은 현재 바인딩을 인자가 반환형인지 인자 값인지를 선택하는 매개변수이다.

ValueType은 매개변수의 C 데이터 형식을 넣어주고, ParameterType에는 매개변수의 SQL데이터 형식을 넣어준다.

ColumnSize는 매개 변수의 크기를 넣어준다. DecimalDigits 열 크기를 받는 인자값이며 보통 0을 쓴다. ParameterValuePtr은 반환을 받을 변수의 포인터를 넣어주면 된다. BufferLength는 ParameterValuePtr의 바이트 버퍼인데 보통 0을 넣어준다. StrLen\_or\_IndPtr은 매개변수 길이에대한 버퍼의 포인터인데 보통 nullptr을 넣어준다.

# TSQL



[그림 2-1] 동기 입출력과 비동기 입출력

동기 입출력은 입출력 함수가 호출되고 해당 입출력 작업이 완료가 될 때 함수가 반환이 되는 형태이다. 이로 인해 동기 입출력은 입출력이 완료가 되기 전까지 쓰레드는 다른 작업을 할 수가 없다.

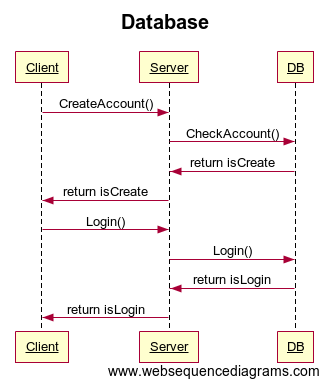
반면에 비동기 입출력은 입출력 함수를 호출하는 시점에 바로 반환을 해서, 입출력 작업이 진행되는 중에도 쓰레드는 다른 작업을 진행할 수가 있다. 때문에 비동기 입출력에서 쓰레드는 다른 작업을 진행한 후에 입출력 작업이 완료되었는지 확인을 해야 할 필요가 있다.

# DB 프로그램 분석

DB프로그램은 Socket프로그래밍에서 했던 채팅서버에 DB를 붙이는 방식으로 구현되었다.

클라이언트는 서버에 접속을 하게 되면 로그인 화면이 나오게 되는데 여기서 ID를 생성하고 생성한 ID로 서버에 로그인을 하게 되면 이전과 똑같이 채팅을 할 수 있는 방식이다.

클라이언트에서 ID를 생성하면 서버로 그 내용을 전송하고, 서버는 그 내용 그대로 다시 DB에 보내게 된다. DB에서는 전송된 정보를 확인해서 ID가 DB상에 존재하는지를 확인하게 된다. 만약 존재한다면 서버로 생성을 못했다는 정보를 반환하고, 서버는 다시 클라이언트에게 그 내용을 전송하게 된다. ID가 존재하지 않는다면 DB는 해당 ID와 Password를 등록하고 서버에 등록을 했다고 반환을 해주면, 마찬가지로 서버는 다시 클라이언트에게 ID가 생성되었음을 알려준다. 이렇게 해서 ID가 생성됐으면 클라이언트는 그 ID로 서버에 접속을 하게 되는 방식이다. 로그인을 하는 과정도 이와 비슷하게 진행이 된다. 클라이언트에서는 ID와 Password를 입력해서 서버에 전달하면 서버는 똑같이 DB에 해당 내용을 전송하고, DB에서는 ID를 조회하고 Password가 일치하는지를 확인한 뒤 서버에 다시 재전송해준다. ID와 Password가 일치하면 서버로 접속이 되고, 아니면 할 수 없게 되어있다.



[그림 3-1] 클라이언트와 서버 DB의 시퀀스 다이어그램

서버는 클라이언트가 ID를 생성하거나 로그인 할 때 외에도 DB에 접속해서 데이터를 확인할 수 있는데, 데이터 조회, 추가, 편집, 삭제가 가능하다. 데이터 조회는 ID와 Password외에 DB에 기록되는 계정 생성 시간, 최근 로그인 시간, 로그아웃 시간을 확인할 수 있고, 추가를 하게 되면 클라이언트에서 ID를 만들었을 때와 같은 방식으로 ID를 생성하게 된다. 데이터의 편집과 삭제는 조금 주의가 필요한데, 현재 서버에 접속중인 유저의 데이터를 편집, 삭제하게 될 시 변경된 내용 또한 서버에 있는 유저리스트에 적용이 될 필요가 있다. 따라서 현재 프로그램은 서버에서 DB의 내용을 편집할 때 그 유저가 서버에 접속 중이라면 서버에 유저리스트도 적용이 되게 구현이 되어있다.