

---

Subject: Re: MSSQL Exception Handling  
Posted by [Pradip](#) on Fri, 10 Apr 2020 15:08:27 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Well, I think I've managed to do it by creating a manager class, posting the code here, if it helps others. \*\*\* to be replaced by appropriate code. Also expecting comments from experienced users.

I think the key here was using `SqlSession::InstallErrorHandler`

This app has option of using either mssql or sqlite3, hence in all listing both options will be seen.

.h file:

```
enum DBTYPE {MS, LITE};

class SqlManager {
private:
    static MSSQLSession mssql;
    static Sqlite3Session sqlite;
    static DBTYPE db_type;
    static String ms_con_str, lite_file;

public:
    typedef SqlManager CLASSNAME;

    SqlManager();
    void InstallErrorHandler();
    void UninstallErrorHandler();
    void AssignSession();
    static bool Connect();
    static bool ErrorHandler(String error, String stmt, int code, const char *scode,
        Sql::ERRORCLASS cls);
};

MSSQLSession SqlManager::mssql;
Sqlite3Session SqlManager::sqlite;
DBTYPE SqlManager::db_type;
String SqlManager::ms_con_str, SqlManager::lite_file;
```

.cpp file:

```
// comment / uncomment to change database type
#define DBTYPEMS
//#define DBTYPELITE

#ifdef DBTYPEMS
#define SCHEMADIALECT <MSSQL/MSSQLSchema.h>
#endif
```

```

#ifdef DBTYPELITE
#define SCHEMADIALECT <plugin/sqlite3/Sqlite3Schema.h>
#endif

#define MODEL <***.sch>

#include <Sql/sch_schema.h>
#include <Sql/sch_header.h>
#include <Sql/sch_source.h>

SqlManager sqlm;

//=====================================================

SqlManager::SqlManager() {
#ifdef DBTYPEEMS
db_type = MS;
#endif

#ifdef DBTYPELITE
db_type = LITE;
#endif

switch(db_type) {
case MS:
ms_con_str = "Driver={SQL Server Native Client 11.0};";
ms_con_str << "Server=***,";
ms_con_str << "UID=***,";
ms_con_str << "PWD=***,";
ms_con_str << "Database=***,";
break;
case LITE:
lite_file = "***";
break;
}

InstallErrorHandler();
}

void SqlManager::InstallErrorHandler() {
switch(db_type) {
case MS:
mssql.InstallErrorHandler(&SqlManager::ErrorHandler); break;
case LITE:
sqlite.InstallErrorHandler(&SqlManager::ErrorHandler); break;
}
}

```

```

void SqlManager::UninstallErrorHandler() {
    switch(db_type) {
        case MS:
            mssql.InstallErrorHandler(NULL); break;
        case LITE:
            sqlite.InstallErrorHandler(NULL); break;
    }
}

```

```

bool SqlManager::Connect() {
    DUMP("Connect");
    switch(db_type) {
        case MS:
            while(!mssql.Connect(ms_con_str))
                if(mssql.WasError()) return false;

            break;
        case LITE:
            while(!sqlite.Open(lite_file))
                if(sqlite.WasError()) return false;

            break;
    }
    return true;
}

```

```

void SqlManager::AssignSession() {
    switch(db_type) {
        case MS:
            SQL = mssql; break;
        case LITE:
            SQL = sqlite; break;
    }
}

```

```

bool SqlManager::ErrorHandler(String error, String stmt, int code, const char *scode,
Sql::ERRORCLASS cls) {
    DUMP("ErrorHandler");
    DUMP(code);

    // code, scode or cls seem to work only for code = 53!
    // Hence for other errors using substring matching in error string
    // server timeout
    if(code == 53)
        return ErrorRetryCancel("[* Cannot connect to database server:]" + DeQtf(error));

    // unable to complete login

```

```

else if(error.Find("Unable to complete login process") >= 0)
    return ErrorRetryCancel("[* Database server login error:]\&" + DeQtflf(error));

// if error, try to reconnect; if successful, rerun query
else if(error.Find("An existing connection was forcibly closed") >= 0) {
    if(Connect()) {
        SQL.ClearError();
        SQL.Execute(stmt);
        return !SQL.WasError();
    }
    else return false;
}

// all other errors
else {
    Exclamation("[* Miscellaneous error:]\&" + DeQtflf(error));
    if(SQL.GetTransactionLevel()) {
        SQL.Rollback();
    }
    return false;
}
}
...
GUI_APP_MAIN {
    if(!sqlm.Connect()) return;
    sqlm.AssignSession();
    SQL.GetSession().ThrowOnError(false);
    ...
}

```

and for running insert or update query; select queries run without any modification; of course SQL.WasError() can be checked after select queries too and appropriate code can run if error:

```

q * Update(ACTIVITIES)(startDtId, newStartDt).Where(ACT_ID == Id);
if(SQL.WasError()) return false;

```