

C++ এ ADO ডাটাবেজ

ডাটাবেজ বিশেষ করে ADO Database-কে C++ এর মাধ্যমে ব্যবহার বা নিয়ন্ত্রণে প্রাথমিক জ্ঞান হিসেবে এই ফিচারটি খুবই উপযোগী। এতে MFC, .NET অথবা অন্যকোনো জটিল মাইক্রোসফট ফ্রেমওয়ার্ক ব্যবহার করা হয় নি। এটি করতে হলে আপনার C++ এর প্রাথমিক জ্ঞান, অবজেক্ট কনসেপ্ট জানা থাকতে হবে। আসুন তবে ডাটাবেজ প্রোগ্রামিং শুরু করি।

প্রথমে আমাদের একটি হেডার ফাইল ইনক্লুড করতে হবে। <stdafx.h> নামক। এছাড়াও msado15.dll নামক ADO লাইব্রেরির সাথেও কানেক্ট করতে হবে। এটি সবচেয়ে সহজে করা যায় #import স্টেটমেন্ট-এর সাহায্যে। যেমন—

```
#import "C:\program files\common files\system\ado\msado15.dll" rename ("EOF", "EOFFile")
এখানে rename()-টি ব্যবহারের কারণ হলো—
যাতে Existing EOF মার্কারগুলো ওভাররাইট না হয়ে যায়। এরপর একটি ভ্যারিয়েবল ব্যবহার যা ব্যবহৃত হবে OLE Library-গুলোকে Initialize/uninitialize করার জন্য। আপনি সরাসরিও ফাংশনগুলো কল করতে পারেন। তবে ভ্যারিয়েবল ব্যবহার করে সেগুলো ভালোভাবে মেইনটেইন করা যায়। নিচে এই কাজে ব্যবহৃত Structure-টি দেয়া হলো—
Struct StartOLEProcess{
StartOLEProcess(){
:: Colnitalize (NULL);
}
~StartOLEProcess(){
:: CoUninitialize();
}
}_start_startOLEProcess;
```

এখন চলুন ডাটাবেজের ওপর কাজের ফাংশনগুলো দেখি। ডাটাবেজে কানেক্ট করার জন্য বেশ কিছু ভ্যারিয়েবল লাগবে। প্রথমে রয়েছে ডাটাবেজটির সাথে কানেকশনকে স্টোর করার একটি ভ্যারিয়েবল অনেকটা FILE*-এর মতোই। একটি রেকর্ডসেট ভ্যারিয়েবল লাগবে যাতে ডাটাবেজ হতে রিট্রাইভকৃত ডাটা রাখা হবে। এখন তাহলে এ রেকর্ডসেট ভ্যারিয়েবলটি ডাটাবেজ-এর ডাটা ধারণ করছে, তবে recordset থেকে Data সরাসরি এক্সেস করা যায় না। এ জন্য প্রতিটি ফিল্ডের

জন্যও একটি করে ভ্যারিয়েবল লাগবে। ডাটাবেজ নিয়ে কাজ করার সময় ADO আমাদের তথ্য দিতে পারে কোন রেকর্ডগুলো আমাদের গ্র্যাকশনের কারণে এফেক্টেড হয়েছে। এর জন্যও একটি ভ্যারিয়েবল ব্যবহার করতে হয়। শেষে আরেকটি ভ্যারিয়েবল ব্যবহার করতে হবে OLE-কে ইনিশিয়ালাইজ করার জন্য। ভ্যারিয়েবল Declaration গুলো নিম্নরূপ—

```
/*The database connection variable*/
ADODB : :_Connection Ptr Con=NULL;
/*The recordset variable*/
ADODB : :_RecordsetPtr RecSet=NULL;
/* A single field pointer, you will probably need multiple for real database work */
ADODB : :Field Ptr Field;
/* Variable to get the affected records */
VARIANT * Records Affected=NULL;
/* The important OLE variable */
StartOLE Process OLEVar;
এই ভ্যারিয়েবলগুলোকে আরো organized ব্যবহারের জন্য Start OLE Process স্ট্রাকচারটি ভেতরেও ব্যবহার করতে পারেন। এখন আমাদের কানেকশন ভ্যারিয়েবলটিকে ইনিশিয়ালাইজ করতে হবে। এটা করলে Con-এর মেম্বার ফাংশন Create Instance()-কে কল করে। এটি সাকসেস হলে 0 রিটার্ন করে। এটাই একমাত্র ফাংশন যাকে এভাবে error চেক করা যায়। অন্যসবগুলোর ক্ষেত্রে Try/Catch মেথড ব্যবহার করে error ডিটেক্ট করা হয়ে থাকে, তবে সেসব এখানে আলোচনা করছি না।
if(Con.CreateInstance(__uidof(ADODB::C onnection), NULL)!=0{
printf("Couldn't create the connection variable");
}
```

এখন আমরা দেখব কীভাবে ডাটাবেজে কানেক্ট করব। এর জন্য আমাদেরকে একটি কানেকশন স্ট্রিং তৈরি করতে হবে আর কানেকশন পয়েন্টারের সদস্য ফাংশন Open()-কে কল করতে হবে। এই কানেকশন স্ট্রিং আর কিছুই নয়, এটি হচ্ছে ডাটাবেজটি কোথায় আছে, কী নাম, কোন ধরনের ডাটাবেজ ইত্যাদি তথ্যসমৃদ্ধ একটি Formatted String. ADO দিয়ে অনেক অনেক রকমের

ডাটাবেজে কানেক্ট করা যায় আর প্রত্যেক ধরনের জন্য এই কানেকশন স্ট্রিংও বিভিন্ন ধরনের হয়ে থাকে। এখানে উদাহরণের সুবিধার্থে আমরা Access 2000/XP ডাটাবেজ ব্যবহার করব। কানেকশন স্ট্রিংটি এরকম হবে—

```
"Provider=Microsoft. Jet.OLEDB.40; Data Source= C:\Database.mdb",
এখানে ধরে নিয়েছি ডাটাবেজটির নাম Database.mdb আর এটি C:drive-এর root-এ আছে।
```

Open() ফাংশনটি 4-টি প্যারামিটার সাপোর্ট করে— প্রথমটি কানেকশন স্ট্রিং, দ্বিতীয়টি ইউজার নেম-এর পর পাসওয়ার্ড আর শেষেরটি হলো এক্সেস টাইপ রিড/রাইট। উভয় এক্সেস পাওয়ার জন্য ডিফল্ট মান হচ্ছে 0। username আর Password তখনই দেয়া হয় যদি ডাটাবেজটি এনক্রিপটেড হয় বা এতে এক্সেসের জন্য অনুমতির দরকার হয়। এখানে খেয়াল করুন Open() ফাংশনটি কানেকশন ভ্যারিয়েবলটিকে এমনভাবে এক্সেস করে যেন সেটি একটি পয়েন্টার। যা CreateInstance() ফাংশনের বেলায় করা হয় নি। এর কারণ হলো, আমাদের তৈরি কানেকশন ভ্যারিয়েবলটি একটি পয়েন্টার আর CreateInstance() ফাংশন একটি কানেকশন ভ্যারিয়েবল তৈরি করে তাতে পয়েন্ট করে।

```
try{con->Open
("Provides=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\Database. mdb ","", "",O);
}
Catch (int Exception)
{ /* A try/catch example */
printf("Unable to connect");
}
```

এতক্ষণে আমরা ডাটাবেজটিকে ওপেন করতে পেরেছি। এখন ডাটাবেজের সাথে আমাদের ইন্টার্যাক্ট করতে হবে। ইন্টার্যাকশনের সবচেয়ে ভালো উপায় হচ্ছে SQL স্টেটমেন্ট।

আপনারা কি SQL জানা আছে? ভয় পাবেন না, এতক্ষণ যা যা শিখেছেন তার কাছে SQL-কে নসি মনে হবে। এখানে আপনাদেরকে SQL শেখাব না, তবে SQL-এর বেশিরভাগ স্টেটমেন্ট পড়েই বোঝা যায় (যদি অনেক বড় এবং Nested না হয়)। SQL স্টেটমেন্ট ব্যবহারের জন্য আমরা কানেকশন ভ্যারিয়েবলের মেম্বার ফাংশন Execute()-কে ব্যবহার করব। আচ্ছা, বলুন তো আপনি VB-তে Database প্রোগ্রাম করেছেন? এতক্ষণ করা বিষয়গুলো চেনাচেনা মনে হচ্ছে না? তাহলে আর ভয় কী? VB-র মতো এবার থেকে রাজকীয় ভাষা

C/C++ এ এখন থেকে ডাটাবেজ প্রোগ্রামিং-এ হাত পাকানোর সংকল্প নিন।

এই Execute() ফাংশনটি তিনটি প্যারামিটার গ্রহণ করে— প্রথমে SQL স্টেটমেন্টটি একটি Null টার্মিনেটেড String হিসেবে, দ্বিতীয়টি Records Affected ব্যারিয়েবল আর শেষেরটি অপশন। যদি SQL Statement-টি রান করতে ভুল হয় বা সিনট্যাক্স ঠিক না থাকে তাহলে কিছু Error হবে Execute()-এর থেকে। সুতরাং এটির থেকে Exception Catch() করতে ভুলবেন না। Execute()-টি ঠিকঠাক এক্সিকিউট হলে এটি একটি recordset রিটার্ন করবে যা SQL স্টেটমেন্টটি চালানোর ফল। তাহলে এর রিটার্নটি আমরা সেই রেকর্ডসেট ব্যারিয়েবলে এসাইন করে দেব। দেখুন—

```
RecSet=Con->Execute ("Select *From
Table1", Records Affected,1);
```

আগেই বলেছি রেকর্ডসেট হতে Data সরাসরি পাওয়া যাবে না। প্রতিটি ফিল্ডের জন্য অতিরিক্ত আরো ফিল্ড ব্যারিয়েবল ব্যবহারের প্রয়োজন হবে। এই ফিল্ড ব্যারিয়েবলগুলোই ঐ SQL স্টেটমেন্টের ফলাফল ধারণ করবে যা আমরা ব্যবহার করতে পারব। আর এই ফিল্ড ব্যারিয়েবলগুলি এই মুহূর্তে যে রেকর্ডে অবস্থান করছে সেই রেকর্ডের ডাটাই দেখাবে। তাই রেকর্ডগুলোতে নেভিগেট করতে আমরা recordset ব্যারিয়েবলের MoveNext() আর MovePrevious() মেম্বার ফাংশনদ্বয়কে ব্যবহার করতে পারি। VB প্রোগ্রামারটা এটি জানেন যে, রেকর্ডে নেভিগেশন করার সময় ফিল্ডের ভ্যালুও পরিবর্তন হবে অর্থাৎ কারেন্ট রেকর্ডের Data দেখাবে। রেকর্ডসেট ব্যারিয়েবলের একটি EOFFile সদস্য আছে (সাধারণত EOF, কিন্তু আমরা #import-এর সময়ে একে রিনেম করেছি), যদি এর মান 0 না হয় তাহলে রেকর্ডের শেষ প্রান্তে পৌঁছেছে বোঝা যাবে। একটি While লুপ দিয়ে ডাটাবেজটির (তথা রেকর্ডসেটটির) সকল রেকর্ডে চক্কর মেরে আসা যাবে। আসুন দেখি—

```
/*Assigned the field variable to the field
"Field1"*/
Field = RecSet-> Fields->
getItem("Field1");
While (!RecSet-> EOFFile)
{ /*Access the data from the field variable
*/
```

```
RecSet-> MoveNext();
}
```

যেহেতু ডাটাবেজের ফিল্ডগুলো বিভিন্ন ধরনের ডাটা টাইপের হতে পারে, সুতরাং প্রত্যেক প্রকারের ডাটা টাইপের জন্য সেসব ফিল্ডকে বিশেষ বিশেষ উপায়ে হ্যান্ডল করতে হবে। সাধারণভাবে বলা যায়, ডাটাবেজে তিন প্রকারের ডাটা থাকে— String, numbers এবং date। এগুলোর আবার Subtype আছে, কিন্তু প্রোগ্রামে সেগুলোকে একটি সাধারণ উপায়ে হ্যান্ডল করা হয়। আমরা ফিল্ড ব্যারিয়েবল এবং Type চেক করে ঐ ফিল্ডের Data Type জানতে পারব। এর ফলে আমরা কিছু ইন্ডেক্সার মান পাব। এখানে সবগুলো দেয়া সম্ভব নয়, তাই কিছু কিছু দিচ্ছি।

সকল ধরনের ফিল্ডের জন্য Type নাম্বার পাবার নিমিত্তে আমি একটি ডাটাবেজ তৈরি করে নিয়েছি যাতে সকল রকমের ফিল্ড ছিল, এরপর Recordset-এ ঐ সকল ফিল্ডকে রেখে প্রোগ্রাম থেকে প্রতি ফিল্ডের সাথে Type number মিলিয়ে দেখার ব্যবস্থা করেছিলাম। এভাবে আপনিও করতে পারেন। এটি এক্ষেত্রে এবং সময় সাপেক্ষ একটি কাজ কিন্তু তারপরও ধৈর্য ধরে করে দেখুন।

প্রতিটি ফিল্ড ব্যারিয়েবলের অভ্যন্তরে Value নামে _variant_t টাইপের একটি সদস্য আছে যা দিয়ে



প্রকৃত ফিল্ড ডাটা পাওয়া যেতে পারে। এই ভ্যালুতে সঠিক ভ্যালুটিই থাকে। যেমন— কোনো Database ফিল্ড যদি ইন্ডেক্সার হয়, তবে Value ও ইন্ডেক্সারই হবে। ইন্ডেক্সার ভ্যালুকে Value মেম্বার হতে সরাসরি এক্সেস করা যায়। তবে স্ট্রিংকে স্ট্যান্ডার্ড Null Terminated এ্যারেতে

রূপান্তর করা লাগে।

এ জন্য আমাদের WideCharToMultibyte()-এর ব্যবহার করতে হবে, এটি আটটি প্যারামিটার গ্রহণ করে। তবে ভয় পাওয়ার কিছু নেই তৃতীয়, পঞ্চম আর ষষ্ঠ প্যারামিটার বাদে অন্যগুলো, প্রায় সর্বদা একই হয়ে থাকে। তৃতীয় প্যারামিটার হচ্ছে— যে স্ট্রিংকে আপনি কনভার্ট করবেন, পঞ্চমটি হচ্ছে— যে বাফারের কনভার্টকৃত স্ট্রিংটি স্টোর করা হবে তার পয়েন্টার, ষষ্ঠটি হচ্ছে বাফারটির সাইজ। এখানে কতগুলো কন্ডিশনাল স্টেটমেন্ট দেখানো হলো যার মাধ্যমে ফিল্ড ব্যারিয়েবল হতে ডাটা হ্যান্ডল করা যাবে।

```
if((Field-> Type==202) || (Field->
Type==203))
{ /* Handling a string Field type of
database /
char String [100];
WideCharToMultibyte(CP_ACP,0,
Field.bstrVal,-1, String, 100, NULL,
NULL);
printf ("Data : %s\n",String);
}
if((Field->Type==7) || (Field-> Type ==
11))
{ printf("Data;%d\n",Field-> Value,
boolVal);
}
if(Field-> Type==2){
printf("Data : %d\n", Field-> Value.IntVal);
}
Database-এ সকল কাজ শেষ হয়ে গেলে তাকে
এবং তার recordset-কে আবার বন্ধ করা লাগে।
এর জন্য এই দু'য়েরই Close() নামক একটি
ফাংশন রয়েছে।
RecSet-> Close();
Con-> Close();
এখানে কিছু ADO-এর সকল Aspect আলোচনা
করা হয় নি অথবা ব্যবহৃত সকল ফাংশনেরও
বিশেষ কোনো আলোচনা করি নি। এমনকি
কানেকশন স্ট্রিংটিও অনেক অপশনের সমন্বয়ে
গঠিত হয়ে থাকে।
```

■ গাজী লেনিন

BBIT