「ソースファイル」のフォルダ内にはDBComparerプロジェクトに「main.cpp」がありますが、ここにはありません。

ないほうが正解（エントリポイントはここではなくC#側）かと思いますが、消したということで合ってますか？

また、DBComparerプロジェクトはビルドされず、DBComparerWrapperからコードが参照されるだけの状態になっていると思いますが、

この2つを別プロジェクトにした理由はありますか？（C++/CLIを使っているからでしょうか？すみません、C++/CLIについてはあまり詳しくないです）

特に理由がなければ、これらは1つのプロジェクトになっていたほうがわかりやすいと思いました。

あとは所感ですが、C#とC++が混在しているので、C++/CLIを使ったりなど、構造が複雑になっていると思います。

crd\_cnv, schemaのライブラリを使いたかったからでしょうか？

どの程度ライブラリの機能に依存しているかにもよりますが、混在させるよりは必要な部分をC#に換装してしまったほうがメンテはしやすくなると感じました。

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/DBComparer/DataManager.h**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103189#file103189line96)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **96** | void setAccSV(AccessSV \* AccSV); |
|  |  | **97** | /\*\* |

全体的に言えるのですが、こういうところは空行を入れると見やすくなります。

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/DBComparerWrapper/DBComparerWrapper.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103198#file103198line22)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **22** | DBComparerClass = new DBComparer(); |

C++では、newをした場合どこかでdeleteしないと、そこで確保したメモリがリークします。

ですので、特に理由がなければポインタにしないほうがよいかと思われます。

<定義>

private :

DBComparer DBComparerClass;

<実装>

DBComparerClass.getFeatureClassList(...);

DBComparerWrapperがインスタンス化されたタイミングで、メンバのDBComparerClassもインスタンス化されます。

ちなみに、「gcnew」はガベージコレクタが自動解放してくれるので、deleteは必要ないようです（ここはC#側の仕組みと同じですね）。

C++側でnewを使っているところは他にもいくつかあるようなので、確認してみてください。

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/DBComparerWrapper/DBComparerWrapper.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103198#file103198line25)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **25** | int boolValue = DBComparerClass->init(); |
|  |  | **26** | if (boolValue == 0) return true; |
|  |  | **27** | else return false; |

if (DBComparerClass->init() == 0)

return true;

else

return false;

でいいように思います。

「boolValue」は他で使っていないようなので不要ではないでしょうか。

また、「boolValue」という変数名はあまりよくないです。

「boolの値」ということしかわからないからです。

「その値はなにを示しているのか」を具体的に変数名にするほうがよいです。

たとえばここなら、

「succeedsComparing」「hasSucceeded」などでしょうか。

などです。

TDCのコーディング規約ではbool型は以下のような命名をしています。

（以下は関数ですが、変数でも同様です）

[OK]

bool isEmpty()

bool canGet()

bool hasChanged()

bool isContains(Object)

bool containsKey(Key)

[NG]

bool empty()

→動詞的な意味に取れるため良くない。

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/Interface/IOManager.cs**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103201#file103201line34)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **34** | private String empty = ""; |

""

は

string.Empty

と書いたほうがベターです。

（ここ以外も同様です）

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/Interface/IOManager.cs**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103201#file103201line51)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **51** | public String getUser1Variable() |
|  |  | **52** | { |
|  |  | **53** | return user1Variable; |
|  |  | **54** | } |

getterやsetterは「プロパティ」を使うとよいです。

private string user1Variable { get; private set; }

こう書くと、クラス外からは読み取り専用、クラス内からは読み書き可、となります。

プロパティは他にも適用できそうな箇所が見受けられるので、余裕があれば探してみてください。

| [**http://preon/svn/source/trunk/training/p\_sereeyotin/DBStatsComparer/Interface/IOManager.cs**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6708/diff/1/?file=103201#file103201line75)(Diff revision 1) | | | |
| --- | --- | --- | --- |
|  |  | **75** | /\*\* |
|  |  | **76** | \* @brief Remove table name from checklist if the name is exist in database |
|  |  | **77** | \* @param tableName [in] tablename that exists in database |
|  |  | **78** | \*/ |

C#でコメントを残す際は、doxygen形式ではなくXML形式のほうがいいです。

Visual Studioで、ファンクション等の前の行で

「///」とスラッシュを3回打つとXMLコメントのテンプレートが生成されるので便利です。

また余談ですが、TDCでは「StyleCop」というツールを導入することで、コーディングスタイルの統一化と自動チェックを行っています。

Visual Studioに組み込むことでビルド時に自動チェックしてくれるようになるので、コードが見やすくなり、オススメです。

- 中里