

Re: Creating tables

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2004-10/0500.html>

From: Peter E.C Dashwood (*dashwood_at_enternet.co.nz*)

Date: 10/16/04

Date: Sat, 16 Oct 2004 15:07:23 +1300

"Dionisis Vrionis" <diov@dksoft.gr> wrote in message
news:ckotcq\$130t\$1@ulysses.noc.ntua.gr...
> *Your sample it works fine.*
> *Thanks all (specially you Peter) for help.*
>

Excellent! Glad it was of use.

Frederico has recommended you to replace the solution with an ADO one.

I neither agree nor disagree with this.

Here's some quick thoughts on the matter...

1. ADO is ActiveX Data Objects and is MicroSoft's stated "direction" for database access. It is component based and is more flexible and intended to handle more data sources than even ODBC. Three years ago they were telling everybody to go this way and they still are.
2. DAO is "Data Access Objects". It utilises a model which is based on direct interface to the database engine and it is being implemented in open software, outside MicroSoft. (of course, MS have their own DAO also). The ADO model is simpler and less hierachic because it is component based, but I have found the DAO model to be excellent when using ACCESS.
3. A quick web search reveals current posts on DAO so, despite recommendations from MS, it is still being used. You should do a few searches of your own on these two technologies and make your own mind up. I think they both have merit and I use them both.

Fujitsu PowerCOBOL provides an ADO based control for connection to databases and it works pretty well. (I don't use it, although I tried it once. I have a slightly different philosophy about databases and believe they should be used solely as repositories (no database dependent SQL and I'm still making my mind up about stored procedures), I also believe in separation layers (multi-tier model), so I like ODBC and COBOL or Java.

comp.lang.cobol: Re: Creating tables

Around two years ago, I was managing the Legacy systems for a major UK utility and most of my team were ex-COMPAQ guys who were born and bred into Client/Server. (All of the Legacy was on Client/Server, so it is not only mainframes that are being replaced by "new technology"; in this case the Siebel Solution (£30,000,000 over three years...)). Many of our team used SQL Server with stored procedures and triggers, and they explained to me their reasons for doing so. I explained why I don't, and we had some really interesting discussions. On balance, I think they were right and I probably would use stored procedures now, but ONLY if I was totally persuaded there was not going to be a change of DB technology or vendor any time soon...

4. It would be a fairly simple change to "convert" the DAO solution to ADO (I'm surprised Frederico didn't provide some code, rather than just a personal opinion); You can use ADO at different levels of the model, directly.

5. One always needs to think carefully before replacing a working solution with one that is based on intangibles ("It's more efficient" (How much and does it matter?), "we're phasing out the old way" (When?), "Everybody's doing it this way..." (Sure, Doc's favourite: "Brooklyn Bridge"). There may be benefits with an ADO approach but how quantifiable are they? This is something you need to decide for yourself.

Regards,

Pete.

(Top post, nothing more below.)

> *"Peter E.C Dashwood" <dashwood@enternet.co.nz> wrote in message
> news:1097670067.BM6bnx51U2GOAH7FlW/9Hg@teranews...
>>
>> "Dionisis Vrionis" <diovrr@dksoft.gr> wrote in message
>> news:ckg68k\$1km9\$1@ulysses.noc.ntua.gr...
>>> I Have a database named test.mdb without tables.
>>> How can i create a table in a database from netcobol.
>>>
>> What Richard wrote is true, however, there is more than one way to skin
a
>> cat, and tables can be created by more than just embedded SQL.
>>
>> With a little imagination, some knowledge of the DB object model, and an
> OO
>> COBOL compiler (like NetCOBOL), you can certainly create tables "on the
>> fly".
>>
>> Here's how....
>>
>> ENVIRONMENT DIVISION.
>> REPOSITORY.
>> CLASS COM-EXCEPTION AS *"*COM-EXCEPTION"**

comp.lang.cobol: Re: Creating tables

```
> > CLASS COM AS "*COM".
> > DATA DIVISION.
> > WORKING-STORAGE SECTION.
> > 01 DAO-connection pic x(8192) value "DAO.DBEngine.36". *> Make sure
> this
> > is the COM ProgID of the
> >
> > *> the version of ACCESS or DAO you have on
> >
> > *> your system. Search the registry for DAO...
> >
> > *> (Data Access Objects)
> > 01 OBJ-DBEngine OBJECT REFERENCE COM.
> > 01 OBJ-DAOCONNECTION OBJECT REFERENCE COM.
> > 01 OBJ-Database OBJECT REFERENCE COM.
> > 01 NAME-DB PIC X(whatever) value "Name of the DB you want to access
(can
> > be ODBC DSN)".
> > * [You might want to pass this through Linkage...]
> >
> > 01 create-stuff pic x( whatever) value "CREATE TABLE tablename (field,
> > field, ...) CONSTRAINTS etc".
> >
> > ===== Examples of valid CREATE entry (valid for SQL
> > Server and ACCESS 2003)=====
> > You could set up "create-stuff" to contain the following...
> >
> > CREATE TABLE jobs
> > (
> > job_id smallint
> > IDENTITY(1,1)
> > PRIMARY KEY CLUSTERED,
> > job_desc varchar(50) NOT NULL
> > DEFAULT 'New Position - title not formalized yet',
> > min_lvl tinyint NOT NULL
> > CHECK (min_lvl >= 10),
> > max_lvl tinyint NOT NULL
> > CHECK (max_lvl <= 250)
> > )
> >
> > It will create the table as described.
> >
> > If you want a temporary table try...
> >
> > CREATE TABLE #tempTable (tempField int... etc)
> >
> > Here's the offical story on temporary tables...
> >
> > Temporary Tables
> > You can create local and global temporary tables. Local temporary tables
> are
```

comp.lang.cobol: Re: Creating tables

> > *visible only in the current session; global temporary tables are visible*
> > *to*
> > *all sessions.*
> >
> > *Prefix local temporary table names with single number sign*
> > *(#table_name),*
> > *and prefix global temporary table names with a double number sign*
> > *(##table_name).*
> >
> > *SQL statements reference the temporary table using the value specified*
for
> > *table_name in the CREATE TABLE statement:*
> >
> > *CREATE TABLE #MyTempTable (cola INT PRIMARY KEY) (this would be one*
> > *EXECUTE*
> > *statement)*
> > *INSERT INTO #MyTempTable VALUES (1) (This would be another EXECUTE*
> > *statement)*
> >
> >
>

=====
> > =====
> >

> > *01 Temp-text pic x(50).*
> > *01 connection-flag pic x value '1'.*
> > *88 no-connection value zero.*
> > *88 connected value '1'.*
> > *PROCEDURE DIVISION. [using databaseName, TableName perhaps...]*
> > *DECLARATIVES.*
> > *ERR SECTION.*
> > *USE AFTER EXCEPTION OLE-EXCEPTION*
> > *move spaces to DB-path*
> > *set no-connection to TRUE*
> > *.*
> > *END DECLARATIVES.*
> > *main section.*
> > *set connected to TRUE*
> > *perform a-section*
> > *.*
> > *return-to-caller.*
> > *exit program.*
> > **=====*
> > *a-section section.*
> > *a000.*
> > ***
> > ** Establish COM Object references to the specified*
> > ** ACCESS Database using DAO.*
> > ***
> > *invoke COM "CREATE-OBJECT"*
> > *using DAO-connection*
> > *returning OBJ-DBEngine*

```
> > end-invoke
> >
> > invoke OBJ-DBEngine "OpenDatabase"
> > using NAME-DB
> > returning OBJ-Database
> > end-invoke
> > *
> > * Check we got to the DB OK...
> > if no-connection
> > yada-yada-yada...
> >
> > * Now you can use the EXECUTE method of the Database to create your new
> > table...
> >
> > invoke OBJ-Database "Execute"
> > using create-stuff
> > end-invoke
> > ....
> > a999.
> > exit.
> >
> > The above is not definitive, but it has been put together from currently
> > working code. Hopefully there's enough here to get you going.
> >
> > Pete.
> >
> >
> >
>
>
>
```