



TeraTech News

Tools for Programmers

We make you a computer hero every day

12221 Parklawn Dr., Ste 200
Rockville, MD 20852
<http://www.teratech.com>

November 2002
Liz Arroyave, Editor

(800) 447-9120 • (301) 881-1440
Fax (301) 881-3586
info@teratech.com

Read about:

- [What is Access?](#)
- [Views and Stored Procedures from ColdFusion](#)
- [CFMX Tip](#)
- [Prototyping](#)

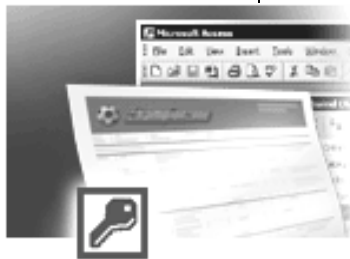
What is Access?

Access is 3 different things:

1) It is a small database designed for desktop use. That database that comes with Access is known as the "Jet Database Engine". This is the part of Access that is slow and that should not be used for large-scale deployments. However, Access is not limited to Jet. Through linked tables, Access can have direct access to other "enterprise" wide data, such as MS SQL Server or Oracle data. Access can link to data in any ODBC compliant database.

2) It is a "front-end" user interface, and rapid development environment, to manage and visualize your data. This is the part of Access that involved forms, reports and Visual Basic for Applications (VBA) code. This part of Access is very powerful and flexible, especially when the data is not stored in Jet, but in MS SQL Server or Oracle (or any other ODBC compliant database), and accessed through linked tables.

3) Access Database Project (ADP) is a feature that came out with Access2000. An Access Project is really a development environment dedicated to MS SQL Server and MSDB (Microsoft Database Engine). No other databases can be used in an Access Project. In an Access Project, you have design privileges to SQL Server objects, such as Tables, Views and Stored Procedures. Of course, you also have the forms and reports and VBA code that exists in a normal MDB file. Also, there are no linked tables, and all data access to the SQL Server is done through OLEDB, which is much faster than linked tables.



Views and Stored Procedures from ColdFusion

Views: are SQL servers name for Queries in Access. They are stored single SQL statements.

Q: Isn't it true that there are significant performance gains when using Views instead of Tables? What are some of the limitations of Views? For example, they're not updateable right....

Views enhance performance very dramatically by doing the following:

1. SQL Server manages views as if they were actual tables therefore the results of the SQL Statement defining the view are instantly available.
2. The Database connection - i.e. ODBC or preferably OLEDB does not have to translate a complex SQL statement, pass it off to the DB server and wait for the DB server to act on it. This lowers the overhead on the Web server in some cases quite significantly.
3. You can define indexes for views that enhance performance significantly.

We have seen dramatic improvements in performance by switching from ODBC to OLEDB and by incorporating views. Dramatic translates to 10 to 20-fold decrease in processing time.

Views are also an excellent way to enforce security -- you can exclude certain columns from the base table(s) when creating a view, and set permissions for retrieval of the view but not the base table(s), to ensure sensitive data isn't compromised.

Another advantage is you can create and drop views on the fly without affecting the underlying data. Another way in which a view could help is if SQL in the view is optimal. Perhaps a user writing the SQL from scratch would not write the "best" SQL. If you have your expert create the views then you know that your "non-expert" users won't be issuing bad join statements against the base tables.

[more next month]

CFMX Tip

Use Array Notation to eliminate use of Evaluate Function with CFMX and all dot variables being structures you have the added ability to use array notation, which almost completely removes the need for evaluate for dynamic variable names.

```
attributes[#somevariablename#]
```

Do this instead of
Evaluate("attributes.#somevariablename#")

Fudgability and Forgivability

Hal Helms: I went to Baltimore recently to work with John Quarto on some stuff. I was coming from Atlanta; he from New York. I found a nice hotel on Priceline and paid for both rooms.

Steve Nelson: Gee, you paid for both? Hey, why don't we get together in, say, Maui?

Hal Helms: There IS a point to this story...Anyway, John goes back home and the hotel charges him for his room.

Steve Nelson: So you didn't really pay after all, you sly dog!

Hal Helms: No, I paid. They just charged him anyway.

Steve Nelson: Ouch. Not fun.

Hal Helms: No, apparently not. John called me and I gave him the Priceline info. He then called the hotel 6 times without ever getting any help. He got locked into voicemail purgatory whence no escape is possible.

Steve Nelson: So what'd he do? Not just eat the charges?

Hal Helms: No, he called his credit card company to dispute the charges. They asked him how much he wanted to dispute and he told them "all of it". But their system wouldn't let him dispute all of it--I forget the exact reason, but he was first supposed to get something from the hotel (which he couldn't because he couldn't get to a real human being). The person John was talking to at the credit card company understood the problem, but the software wouldn't let her dispute the entire amount.

Steve Nelson: I hope this has a happy ending.

Hal Helms: Well, John, being a programmer, told her, "Try it for the full amount less one penny." Bingo. It worked.

Steve Nelson: Ha!

Hal Helms: But it made me think of an ICQ you and I had a few days earlier. You were talking about Alan Cooper's book, "The Inmates are Running the Asylum." About "fudgability."



Steve Nelson: Yeah, that's a good example of a non-fudgable system.

Hal Helms: Define fudgability.

Steve Nelson: It's obviously a made-up word, but a useful one: it describes the ability that human beings build into real-world systems to let them circumvent the rules. It provides flexibility.

Hal Helms: Give me an example in the non-computer world.

Steve Nelson: OK, let's say we have a process for taking an order from a customer. The process is this:

1. Get all customer info
2. Have customer fill out credit application
3. Get credit rating
4. Send order through for fulfillment

Hal Helms: OK

Steve Nelson: Now, one day, Bill Gates calls you.

Hal Helms: Me?

Steve Nelson: You, Hal Helms. He says, "Hal, I understand you have just written a book called 'Discovering Fusebox', is that right? Well, I'd like to order 50,000 copies. Can you do that for me?" What do you say?

Hal Helms: "Be still, my beating heart"?

Steve Nelson: No! You say, "Well, thanks for the order and all, Bill, but I need to have you fill out a bunch of forms and then I'll have to run a credit check on you."

Hal Helms: ARE YOU CRAZY? This is Bill Gates we're talking about!

Steve Nelson: Hey, a process is a process. At least, that's the attitude that most computer programs take. There's no "fudgability" built into them, no way to tell the computer, "Dude, this is the world's richest guy. He's good for the money."

Hal Helms: There's a big disconnect between the way developers WRITE software and the way users USE software. And that's one reason for using personas, as we've been talking about in the last few newsletters. Personas help maintain a connection between the software developer and that end user who the developer will never see.

[more next month]

Used by permission from HalHelms.com

Copyright TeraTech 2002

TeraTech, Inc.

12221 Parklawn Drive, Suite 200
Rockville, MD 20852

TERATECH CUSTOM PROGRAMMING

- ColdFusion
- VisualBasic
- SQL, Oracle, Access, FoxPro
- Numeric Analysis