

Re: DB design, facilitates Double entries of internal transactions

Source:

<http://www.tech-archive.net/Archive/Access/microsoft.public.access.tablesdbdesign/2004-03/0966.html>

From: Nunya (nunya_at_anon.com)

Date: 03/24/04

Date: Tue, 23 Mar 2004 22:47:05 -0600

Several comments...

I agree with Adrian that you probably shouldn't attempt to build this system from scratch. Even under ideal circumstances, I think you'd be in over your head. If you're truly needed for network/system administration & support, there's no way you'd have time for such a project -- it's daunting even full-time. Most off-the-shelf inventory software is built by teams of experienced database & application developers, with experience/specializations in SQL, programming & accounting. Most have accountants as part of the team for business logic.

The environment you're describing is also disturbing -- you're being constrained at every turn before you've even begun. Management is short-sighted and is making technical decisions without any analysis/design behind them. e.g., single server for everything, they don't want to hear your suggestions, users resistant to change). The single server is also rather risky and foolish, as they're making their whole business dependent on that one server (and more so once they move from their manual/one-off inventory solutions to a centralized system.

This sounds like a death march project. I've been on a couple. One was major customization of a commercial accounting package for a local government to replace a custom system developed and used in-house for years. The users didn't want any change. That former employer is out of the custom software business. :)

You're right to be concerned about the data-conversion/data scrubbing process. Users used to free-form entry in spreadsheets or text documents usually hate/resent/don't understand the requirements for strict validation of entries in databases (Why can't I put "Won't know till Thursday" or "TBD" in that invoice amount field?). Often large chunks of such data are virtually impossible to convert.

I was going to disagree with Adrian on the idea of double entry bookkeeping till I read his remarks more carefully. Now I'll just nitpick :)

It's true that for internal use, you don't need to duplicate all the info in a computer accounting system (though technically the pair of corresponding

transactions, even if just a pair of ID's/amounts, is still double entry. :)
However, depending on internal or external auditing requirements (or even on reporting/presentation needs and application performance), there may be requirements for more double-entry-like (and less normalized) tables.

Now some recommendations... if you must attempt this:

Access is a fantastic application, but I would tend to recommend SQL Server over Access for this project for several reasons:

- * If you have to do transactions over WAN/internet (and even more so if through dialup rather than high-speed connections). I know many of the experienced pros have built robust, reliable Access systems with remote connections, but it requires expertise.

- * Robustness/data integrity: SQL Server allows on-the-fly backups, and can recover from logs if anything goes wrong (if you leave logging on, back up logs often enough, and provide enough hard drive space for logs and data!). Access databases, especially in multiuser apps, can/do corrupt if connections are broken, and they can't be backed up reliably while users are connected. Again, pros often make fantastic Access systems supporting dozens of users, but that requires expertise.

- * Security: Access requires that users have full read/write/delete access to the directory in which the mdb is located, meaning users can copy, edit (if not secured with user-level security), or delete the file. Both SQL Server & Access provide fine-grained access control. SQL Server allows you to use Windows authentication (single login).

If you do use Access, DO use user-level security... but that is rather difficult to do well.

- * Stored Procedures and transaction control on the server.

If you use SQL Server, you ABSOLUTELY should have that on a separate server from your network file server, for both security and performance reasons.

And finally, some answers to your specific questions:

Question 1: Do I have to physically partition the database in order to meet users requirements? This seems to be very common for most company, isn't it?

No, you don't need to, and absolutely shouldn't use separate tables/databases/forms for different shops. You can use user/role-level security, stored procedures, and/or views to control what stores/individuals have access to view and change. Yes, it is a very common requirement.

Question 2: What will be the best way to modify the database when a transaction occurs?

Not exactly sure what you're asking here.

Question 3: Which one is better while handling the transactional processes, 'begin tran/commit tran' functions in SQL stored procedures or '.BeginTrans/.CommitTrans' in ADO coding?

I'm a strong believer in stored procedures. They are a natural for transaction control. They allow you to minimize network traffic by executing several operations within a single procedure rather than sending multiple commands/queries. They improve performance by allowing the server to utilize pre-compiled execution plans. And they allow you to restrict direct table access by allowing read/write/delete only through your stored

procedures.

If you use Access, you'll use ADO for transaction control.

Best of luck!

"Adrian Jansen" <qqv@noqqwhere.com> wrote in message

news:406019c2\$1@duster.adelaide.on.net...

> *Ok, if management dont want to use outside product, I just hope you have
> enough experience to keep them on the right track.*

>

> *Just to get down to details, for a change, you do realise that the very
> ideas behind things like 'double entry bookkeeping' are totally outmoded
by*

> *computer systems ? The whole point (at least as I understand it) was to
> catch errors in hand computing totals in ledgers. If computers make this
> sort of error, you have *big* problems. So 'double entry' is replaced by*

a

> *couple of ID numbers associating the transactions with debits and credits
in*

> *appropriate places. Makes the figures look right to an accountant, but by
a*

> *completely different method.*

>

> *Similarly with inventory transactions, you no longer have a 'quantity in
> stock' from which you add and subtract numbers, but a historical list of
> purchases and sales, the sum of which gives you the current stock at any
> time, after a given starting value.*

>

> *The point is that you must re–think a lot of manual processes to put them
> into a framework where data processing is easy and efficient.*

>

> *Once you have that framework designed, Access, or its bigger brother SQL
> Server can easily handle the task. But the design is the killer, and the
> fun part :-)*

>

> *BTW I dont regard myself as an expert in Access by any means, but maybe I
> have a bit of experience in data handling.*

>

> --

> *Regards,*

>

> *Adrian Jansen*

> *J & K MicroSystems*

> *Microcomputer solutions for industrial control*

>

>