Keep in Sync with Jet Replication

Click & Retrieve Source CODE!

PAUL LITWIN

Replication, once a daunting endeavor for VB and Access developers, is easy when you use VB 4.0's Jet engine.

If you wanted to keep two or more copies of a database synchronized using prior versions of Visual Basic or Access, many obstacles impeded your progress. In fact, database synchronization required so much coding that few developers attempted it. With the support for replication and synchronization built right into the Jet 3.0 database engine used in Visual Basic 4.0 and Access 95, replication is finally a reality for many developers.

Replication amounts to making special copies of a regular Access database. These copies are enabled in such a way that you can easily transfer changes made in one copy to each of the other copies. Jet replication is composed of three steps: replication, synchronization, and conflict resolution.

When you replicate a normal Access database, Jet makes many changes to the database's schema, enabling the database to exchange updates with other databases. When you convert a nonreplicated database into a replicated database, you end up with the design master of a new replica set. A replica set of one, however, is not very useful: there would be no one to exchange rules with. Normally you create a second member of the replica set by replicating the design master. You can create additional members of a replica set by replicating any of its existing replicas.

Two replicas can exchange updates only if they are descendants of the same design master—that is, members of the same replica set. The design master is a special member of a replica set. While you can make schema changes only in the design master, you can make changes to data in any replica of a replica set (unless you elect to make a replica read only).

When you make updates to a replica that's a member of a replica set, Jet tracks the updates using the extra tables and fields it added to the database when you first replicated it. Jet does not, however, send changes to the other members of the replica set without your intervention. In fact, unlike networked databases in a multiuser file server or client/server environment, replicas are not connected except when you temporarily connect them—two at a time—during a synchronization exchange.

When you synchronize two replicas, Jet sends updates from one replica to another. Jet sends only the updates, a

Topology	Star
Latency	Moderate
Load Distribution	Uneven
Network Traffic	Low
Reliability	Good as long as hub doesn't fail; bad if hub fails

Comments:
Appropriate for many situations.
All exchanges are intiated by the center (hub) node.

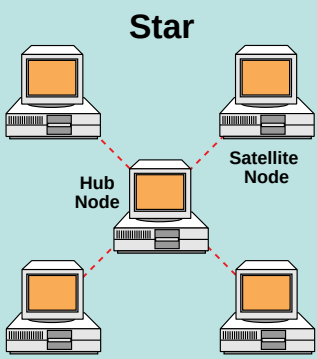
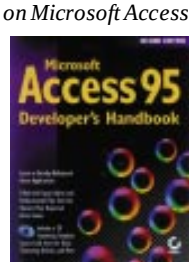


FIGURE 1 ***Pick Your Topology.*** *LANs can be implemented using a variety of topologies, but each topology presents its own set of considerations from the standpoint of synchronization. Choose a topology according to your priorities: how important is latency, network load, and synchronization reliability to you? For example, if short latency is essential but network traffic is reasonable and the size of the LAN is small, you may want to use a fully connected topology.*

Paul Litwin is the editor of Smart Access, *from Pinnacle Publishing, and an independent developer focusing on Microsoft Access and SQL Server solutions. Paul just finished work on two Access 95 books:* Microsoft Access 95 Developer's Handbook, *from Sybex, and* Microsoft Access 95 How-To, from Waite Group Press. *This article was adapted from* Microsoft Access 95 Developer's Handbook *by Paul Litwin, Ken Getz, Mike Gilbert, and Greg Reddick. Reach Paul on CompuServe at 76447,417 or on the Internet at PaulLitwin@msn.com.*



process that is much more efficient than importing and exporting records between two nonreplicated databases.

Normally, synchronization occurs in both directions, but you can elect to synchronize in only one direction if you wish. It's up to you to control when, and between which replicas, synchronization occurs. In addition, you must first establish a physical connection between the two replicas before you can make a synchronization exchange.

In between synchronization exchanges, it's possible that two users may modify the same row in different replicas. When this occurs, Jet flags a conflict the next time the replicas are synchronized. Jet uses a simple algorithm to determine which user's changes are preserved, based on which replica modified the record the greatest number of times. You can override this automated decision by creating custom conflict resolution code (unfortunately, space constraints don't permit me to discuss the creation of custom conflict resolution code in this article).

WHEN TO USE IT?

When you think about it, replication is just another way to share data, and can be used in many of the same situations that classic file server and client/server systems are employed. You may benefit from Jet replication in many situations. In general, whenever you need to keep multiple copies of the same database synchronized, Jet replication is a good candidate.

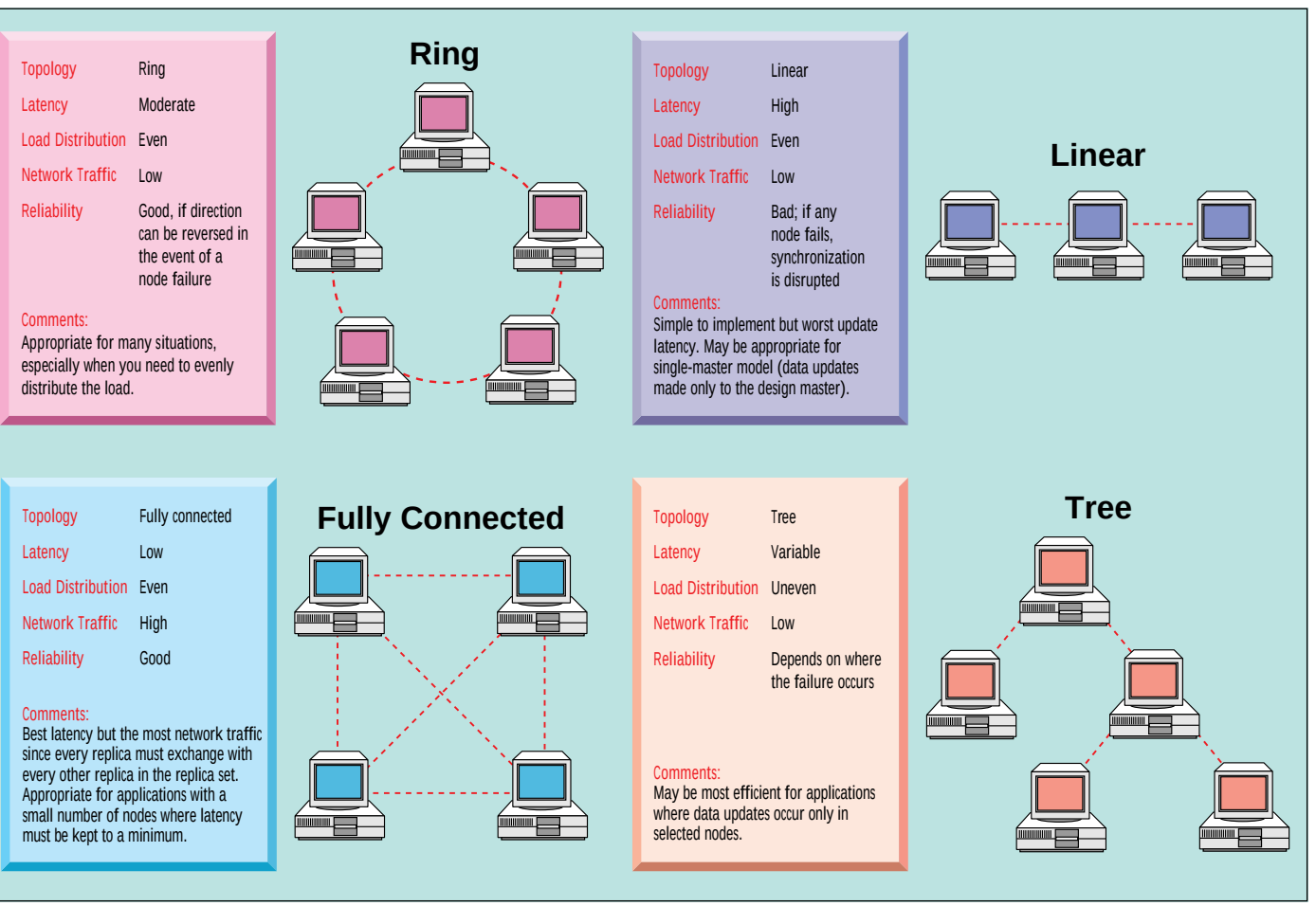
When designing a system for a local area network, you may have difficulty deciding whether to share a single copy of a nonreplicated database across a workgroup or to distribute replicated copies of the database to each user and regularly synchronize changes between the replicated copies. In most cases the more traditional file-server approach makes more sense because of the need for the immediate dissemination of updates to all users.

A replicated system has a greater time lag between the dissemination of updates. This lag period—called the update latency—may vary from an hour to several hours or days, depending on the number of users, the volume of updates, the frequency of conflicts, the synchronization topology, and the synchronization schedule. On the other hand, using replication in this scenario might make sense in one of these situations: if data is updated infrequently, if updates do not usually affect other users, if the network is already overloaded, or if the network is often down.

Another alternative might be to create a hybrid system that uses replication and file-server sharing, especially when you have an overloaded network and can't afford to move to a client/server system. You would create multiple workgroups, each tied to a workgroup server machine. Users within a workgroup would share a database residing on the workgroup's file server. The database would be replicated across workgroup servers, which would be synchronized on a regular schedule. This hybrid system would distribute the load over multiple servers.Update latency would be small within a workgroup, and greater between machines in different workgroups.

The speed of the connection between two nodes (computers) on a wide-area network is slower than that of a LAN. This factor usually rules out the use of a file-server sharing model on a WAN in favor of a client/server system.

With the introduction of replication, however, you may wish to consider replication instead of a classic client/server system when using a WAN. Because synchronization exchanges transfer only the changed records, not whole tables, repli-



cation is well suited for WANs.

In addition to the reasons given for using replication on LANs, you may wish to consider using replication on a WAN if the move to a client/server system is considered too expensive.

Replication is probably not a good choice:

- If users generate many updates, and especially if multiple users often update the same records, causing a large number of conflicts.
- If there is a need for the immediate dissemination of updates to all users.
- If data consistency is critical.

In these situations you'd be better off using a classic client/server, transaction-oriented approach or the replication services of a server database. (see the sidebar, "Contrasting Jet Replication with Microsoft SQL Server 6.0 Replication").

LOOSELY CONNECTED NETWORKS

Mobile laptop users, connecting either infrequently or over slow modem lines to LANs or WANs, do not fit well into either the file-server or client/server system, especially when two-way transfer of updates is needed. This type of system is often ideally suited for Jet replication because users can take a replica with them when they go on the road, plugging back into the corporate network every so often to exchange updates with the "main" database.

Two potential stumbling blocks, however, affect the use of Jet replication on laptops. First, many existing laptops will be underpowered for replication—laptops running VB must have at least 8 MB, and laptops running Access 95 must have 12 MB of RAM. Because there's no support for the replication of partial tables, there's no easy way to have each salesperson's laptop contain only the subset of data pertaining to his or her region.

Fortunately, several possible workarounds address this shortcoming. One possibility is to have laptop users synchronize their data only to the main database. Another potential workaround would be to have separate "territory" tables for each user that could be combined for analyses using a Union query or a routine that created a temporary combined table.

Even if you decide not to use replication to share data in a file-server environment, you may wish to consider using it for maintaining "warm" backups. By replicating a database and regularly synchronizing it with another replica (every 15 minutes or hourly, for example), you'll be ready in the event of a disaster that corrupts or destroys the main database.

If you're developing applications using Access, where the application is stored in the database, you may also wish to consider using replication to distribute application updates, significantly reducing the maintenance burden associated with updating an application distributed to tens or hundreds of workstations.

When replicating a database, you must decide on a synchronization topology and schedule for the replica set. The topology defines the replicas that exchange updates with other replicas, and the direction of such exchanges. The schedule defines when the exchanges occur and who initiates the exchanges. The topology and schedule you choose for your replica set affect update latency.

Various topologies might be used on a LAN (see Figure 1). The synchronization topology you choose depends on the importance of latency, network load, and synchronization reliability. If a very short latency is of utmost importance, network traffic is not a concern, and you don't have many nodes, then you may want to use the fully connected topology. Otherwise, the star usually works best, with the ring topology another common choice.

All the topologies shown in Figure 1 are possible on a WAN or a loosely connected network. You might also use a topology that interconnects several stars or rings.

WHAT CHANGES IN REPLICATION?

When you replicate a database, Jet makes

Contrasting Jet Replication with Microsoft SQL Server 6.0 Replication

Microsoft has added replication capabilities to Microsoft SQL Server 6.0, as well as to Jet 3.0. However, the two products handle replication in substantially different ways.

On the most fundamental level, Jet replication treats the data equally in every replica. You can make a change to data anywhere, and it will eventually propagate around the entire replica set. In contrast, SQL Server replication employs a "publish and subscribe" metaphor. One copy of the database publishes a table, and many others may subscribe to it. SQL Server replication is a one-way-only process (multiple SQL Server databases may publish parts of the same table using horizontal or vertical partitioning, but these multiple publications may not overlap).

Because SQL Server replication operates on a table rather than a database level, it's easy to replicate tables from one database into a second, otherwise dissimilar database.

SQL Server offers rich scheduling options for maintaining synchronization between replicas. Because the server keeps a transaction log, it has the ability to synchronize every time a specified number of transactions has taken place in a table. Thus, you can ensure that SQL Server replicas differ by no more than a maximum number of changed records from the publishing database without resorting to frequent, possibly unneeded synchronizations. Text and image columns—the SQL Server rough equivalents of memo and OLE fields—cannot participate in transaction-based replication. They must be replicated using schedule-based replication instead.

SQL Server does not replicate schema changes to a published table. If you need to change the design of a table in a replicated SQL Server database, you must delete the old publication, create a new one, and manually synchronize the new table into every replica.

SQL Server replication offers an additional level of flexibility through the use of stored procedures to customize replication. A subscribing database can call a stored procedure on any replicated insert, delete, or update. This capability allows you to design replication schemes in which the data on the subscriber is stored differently than it is on the publisher.

Although SQL Server uses ODBC to move data into subscribing databases, neither Jet 3.0 nor SQL Server 6.0 is yet capable of heterogeneous replication. Microsoft has announced that SQL Server 6.5 will support replication to Jet and other ODBC data sources. Meantime, if you have a client/server database using VB or Access for the front end and SQL Server for the back end, you'll need to use the replication capabilities of both products to maintain multiple copies in synchronization. For example, you might use Jet replication to ensure that geographically diverse users have the same forms and reports while using SQL Server replication to publish the data to servers to which those same users connect. —Mike Gunderloy

a number of changes to the database, such as adding fields to each replicated table in the database, changing sequential AutoNumber fields to random AutoNumber fields, adding several system tables to the database, and adding properties to database document objects. These changes can significantly increase the size of your database.

The tables that replication adds to the database are used to track the names and locations of replicas and replicated tables in the replica set, log synchronization exchanges, track deleted records, and log synchronization conflicts and errors. All replication system tables are read only. Most of the fields in these tables are readable, except for some fields where the data is stored in binary form as OLE objects.

In addition to the hidden system tables, Jet creates local (non-replicated) conflict tables whenever outstanding conflicts occur in a database table as a result of a synchronization exchange. Jet constructs the name of the conflict table by appending "_Conflict" to the end of the table that contains conflicts. For example, the conflict table for tblCustomer would be tblCustomer_Conflict.

When a conflict occurs during synchronization because a row has been updated in both replicas, Jet creates a row in the conflict table of the losing replica (if this is the first conflict for a table, Jet first creates the conflict table) and stores in it the losing row. Conflict rows appear only in the losing replica. The replica with the winning update is not notified in any way of the conflict. Jet also adds several fields to each replicated table (see Table 1). All the replication fields are read only.

Jet also alters the behavior of existing AutoNumber fields. If a table contains a long-integer AutoNumber field with a NewValues property setting of Increment, Jet changes the property to Random. This significantly reduces the chance that two replicas will assign the same AutoNumber value, because each AutoNumber field will be based on a randomly selected number between -2 billion and +2 billion. If this still produces too many duplicate values across a replica set, you may wish to use an AutoNumber field of type Replication ID instead. When you use an AutoNumber field with a Replication ID field size, Jet assigns numbers using a globally unique identifier.

While no system can ever guarantee that a number will always be unique, globally unique identifier (GUID) numbers have been designed with global uniqueness in mind. These numbers are also sometimes referred to as universally unique identifiers, or UUIDs.

A Jet-generated GUID is a 16-byte string made of several parts that, when concatenated, have an infinitesimal chance of ever generating duplicate values. And the "global" in GUID means that each GUID will be unique throughout the world, regardless of where or when it was generated (see Figure 2.)

GUIDs are used in several places in a replicated database to uniquely identify many parts of a replicated system, including rows in a replicated table, each table in a replicated database, each replica in a replica set, each synchronization exchange, each database generation, and each schema change.

Replicating a database adds properties to database objects (see Table 2). In addition to the properties in Table 2, Jet adds several other properties to the MSysDb and UserDefined documents of the Databases container that only Access uses to manage replication.

VB3

User Tip

OVERCOME ODBC ERROR

While using MS SQL Server by way of ODBC, Visual Basic 3.0 and Access 2.0 Compatibility layer, the ODBC error message 3146 may occur:

```
ODBC-call failed.  [Microsoft][ODBC SQL Server
Driver]Connection is
busy with results for another hstmt [#0]
```

This occurs when you use DB_SQLPASSTHROUGH on either CreateSnapshot or CreateDynaset methods and another snapshot or dynaset is created.

This behavior did not occur before the compatibility layer was installed. The solution for the changed behavior of the Microsoft Access 2.0 engine is to add an explicit <your data access object>.MoveLast after each query executed using DB_SQLPASSTHROUGH, thereby forcing full population of the result set. This is no slower than the original Visual Basic version 3.0 performance (and possibly faster), and it avoids the error. Note that this problem is fixed in Visual Basic 4.0 for Windows.

—Douglas Haynes

SEND YOUR TIP

If it's cool and we publish it, we'll pay you $25. If it includes code, limit code length to 10 lines if possible. Be sure to include a clear explanation of what it does and why it is useful. Send to 74774.305@compuserve.com or Fawcette Technical Publications, 209 Hamilton Ave., Palo Alto, CA, USA, 94301-2500.

Jet also adds properties to some TableDef fields. The ColGeneration property is added to Memo and OLE Object fields. ColGeneration identifies the name of the field used to track generations for these large object fields.

GET REPLICATED

The first step in employing Jet's replication services is to convert an existing nonreplicated database into a replicated design master.

Using the Jet Data Access Object (DAO), you convert a nonreplicated database into a replicated design master by setting the Replicable property of the database to "T." In an odd stroke of inconsistency with the rest of VB, Access, and Jet, Microsoft has given this and other true/false replication properties a string datatype that must be set to the literal String value of "T" or "F."

Since the Replicable property of the database will not exist until you set it for the first time, you'll need to add the property to the database's properties collection in order to set it. For example, you could use this code to make a new design master from a nonreplicated database:

```
Sub CreateNewReplicaSet(ByVal strDb As String)
   Dim db As Database
   Dim prp As Property
   Set db = DBEngine.Workspaces(0). OpenDatabase(strDb, _
      Exclusive:=True)
   Set prp = db.CreateProperty("Replicable", dbText, "T")
   db.Properties.Append prp
   MsgBox txtDb & " has been replicated.", _
      vbOKOnly + vbInformation, "Create Replica Set"
End Sub
```

Note that I have set the Exclusive parameter of the OpenDatabase method to True. To convert a nonreplicated database to a replicated design master, you must have the database open exclusively. When using Access, you can't convert the currently open database.

CREATING ADDITIONAL REPLICAS

Use the CreateReplica method of the database object to create additional replicas. Here's the syntax:

```
database.MakeReplica replica_name, _
```

Field	DataType	Purpose	Comments
s_Generation	Long Integer	Tracks changes (generations) to a row.	If value is 0, this represents an added row or a changed row that needs to be sent to other replicas during the next synchronization exchange. If it is a nonzero value, it represents the generation of the replica during which this change was made.
s_GUID	AutoNumber, Replication ID	Uniquely identifies the row across replicas, even if the primary key values change.	This field will not be added to the table if the table contains an existing AutoNumber field with a field size of Replication ID.
s_Lineage	OLE Object	This field tracks the history of changes to the record.	
Gen_*Field*	Long Integer	One Gen field is added for each large object (memo or OLE object) user field in the table.	This field tracks changes to the large object It name takes the format Gen_*Field* where *Field* is the name of the large object field.

TABLE 1 ***Replication Fields Added to Each Table.*** *These fields will appear in the Access UI only when you have elected to show system objects. All added replication fields are read only.*

```
   description [, dbRepMakeReadOnly]
```

For example, you could use this subroutine to create a new read/write replica:

```
Sub CreateReplica( _
   ByVal strReplica1 As String, _
   ByVal strReplica2 As String, _
   Optional ByVal varDesc As Variant)
   Dim db As Database
   Set db = DBEngine.Workspaces(0). _
   OpenDatabase(strReplica1)
   If IsMissing(varDesc) _
      Then varDesc = strReplica2
   db.MakeReplica strReplica2, _
      varDesc
   MsgBox strReplica2 _
      & " has been created.", _
      vbOKOnly + vbInformation, _
      "Make Replica"
End Sub
```

Jet stores the value of the description parameter in the Description field of MSysReplicas. You can view and change the value using Replication Manager. In fact, when you use Replication Manager, the description—not the replica name—appears in the Replicas drop-down box, so you may want to create succinct but informative descriptions.

If you include the optional dbRe MakeReadOnly property, the created replica will be read only.

Before you convert a database into a replicated design master, prevent document objects from being replicated by setting their KeepLocal property to "T." In Access, you can manipulate KeepLocal for any document object, including tables, queries, forms, reports, macros, and code. In VB applications, you need to concern yourself only with tables and queries, both of which are members of the Tables container. By default, the KeepLocal property doesn't exist until you create it.

For example, to prevent Jet from replicating the tblLocal table in the ToBeReplicated.mdb database, use:

```
On Error Resume Next

Dim db As Database
Dim doc As Document
```

tblCustomer : Table (Replicated)

Gen_Notes	s_Generation	s_GUID	s_Lineage	LastName	FirstName
1	1	{F4C4088D-44FB-11CF-83F2-444553540000}	Long binary data	Johnson	Alicia Anne
1	1	{F4C4088E-44FB-11CF-83F2-444553540000}	Long binary data	Reddick	Gary
1	1	{F4C4088F-44FB-11CF-83F2-444553540000}	Long binary data	Stevens	Kenneth
1	1	{F4C40890-44FB-11CF-83F2-444553540000}	Long binary data	Jones	Jerry
1	1	{F4C40891-44FB-11CF-83F2-444553540000}	Long binary data	Smithe	Myrna
1	1	{F4C40892-44FB-11CF-83F2-444553540000}	Long binary data	Jenning	Roger
1	1	{F4C40893-44FB-11CF-83F2-444553540000}	Long binary data	Fallon	Jane
1	1	{F4C40894-44FB-11CF-83F2-444553540000}	Long binary data	Phoner	Phil
1	1	{F4C40895-44FB-11CF-83F2-444553540000}	Long binary data	Jones	Bert
1	1	{F4C40896-44FB-11CF-83F2-444553540000}	Long binary data	Babitt	Lucy
1	1	{F4C40897-44FB-11CF-83F2-444553540000}	Long binary data	Litwin	Paul
1	1	{F4C40898-44FB-11CF-83F2-444553540000}	Long binary data	Ayala	Steve
(AutoNumber)	(AutoNumber)	(AutoNumber)			

Record: 1 of 12

FIGURE 2 ***Evidence of Replication.*** *The tblCustomer table shows the replication fields. The s_GUID field is a 16-byte string that uniquely identifies each record across all replicas using a sophisticated algorithm that incorporates the system clock, the network ID, and several other numbers.*

```
Dim prp As Property
Const conErrPrpNotFound = 3270

Set db = DBEngine.Workspaces(0). _
   OpenDatabase("ToBeReplicated.mdb")
Set doc = _
   db.Containers!Tables. _
   Documents!tblLocal
doc.Properties!KeepLocal = "T"
If Err = conErrPrpNotFound Then
   Set prp = _
      doc.CreateProperty("KeepLocal", _
      dbText, "T")
   doc.Properties.Append prp
End If
```

Because you cannot be sure if the KeepLocal property has already been created, you need to set it in two steps. This routine first attempts to set the value of the property. If the property doesn't exist, error 3270 will occur. If this error occurs, the code creates the property and appends it to the document's Properties collection.

After you have replicated a database, the KeepLocal property becomes read only. However, you can use the Replicable property of objects to make an object local or replicated. You can set this property to "T" or "F" (for true or false) in the design master, but you can't set this property to "T" for local objects in nondesign master replicas.

When you change the Replicated property of an object from "T" to "F," Jet makes the object local to the design master and deletes it from other replicas during the next synchronization.

VB4

```
Function glrDbSync(ByVal
strFromDb As String, _
  ByVal strToDb As String, _
  Optional ByVal varExchType As
Variant) As Boolean

   ' Synchronizes two databases
   ' varExchType must be one of
   ' following constants:
   ' dbRepImpExpChanges (the
   ' default),
   ' dbRepExportChanges, or
   ' dbRepImportChanges
   ' Returns True if successful;
   ' otherwise returns False

   On Error GoTo glrDbSync_Err

   Dim dbFrom As Database
   glrDbSync = False

   Set dbFrom = _

DBEngine.Workspaces(0)_
   .OpenDatabase(strFromDb)

   If IsMissing(varExchType) Then _
     varExchType = _
     dbRepImpExpChanges

   dbFrom.Synchronize strToDb, _
     varExchType

   glrDbSync = True

glrDbSync_Exit:
  On Error GoTo 0
  Exit Function

glrDbSync_Err:
  Select Case Err
  Case Else
    MsgBox "Error" & Err.Number _
      &  ": " _
      & Err.Description, _
      vbOKOnly + vbCritical, _
      "Synchronize Replicas"
End Select
  Resume glrDbSync_Exit
End Function
```

LISTING 1 ***A Synch Function.*** *glrDbSync() synchronizes changes between two members of a replica set. While Jet will synchronize two databases for you, you must decide which databases to synchronize, and when. You're also responsible for making sure the two replicas are connected when you want to synchronize them.*

SYNCHRONIZING REPLICAS

When you make a synchronization exchange between two replicas, Jet copies schema changes and data updates between the two replicas. The default exchange method is two way, which means that data updates move in both directions. You can also opt for a one-way data exchange between two replicas. Regardless of whether you choose two-way or one-way exchanges, Jet always propagates schema changes prior to data synchronization.

Synchronizing two databases is simple: Jet does all the work. It's up to you, however, to decide when to synchronize and with whom. You're also responsible for making sure the two replicas are connected when you want to synchronize them.

Initiate synchronization exchanges using the Synchronize method of the database object. The Synchronize method uses this syntax:

```
database.Synchronize DbPathName _
   [, ExchangeType]
```

where DbPathName is the path and name of the replica you wish to exchange with, and ExchangeType is one of these constants: dbRepExportChangesSend, dbRepImportChangesReceive, and dbRepImpExpChanges.

The dbRepImpExpChanges constant, which tells Jet to send and receive data changes between both replicas, is the default. A function called glrDbSync() synchronizes changes between two members of a replica set (see Listing 1).

If you need to synchronize replicas on a regular basis and you've purchased the Access 95 Developer's Toolkit, you may want to use it instead of DAO to schedule synchronizations. It's easier to use, requires no programming, and maintains an excellent history of the exchanges.

However, creating your own synchro-

Object	Property	Description	Read/Write Status
Database	DesignMasterID	Unique identifier assigned to the design master of a replica set.	Read/write
	ReplicaID	Unique identifier for replica.	Read-only
All document objects	KeepLocal	Set to "T" *before* the database is first replicated to make an object nonreplicated (local).	Read/write before the database has been replicated.
	Replicable	Set to "T" *after* the database is replicated to start replicating an object that was local or to "F" to stop replicating an object.	Read/write
Table document objects only	ColIsGuid	The column in the table that serves as the globally unique identifier (GUID); usually s_GUID, but Jet will use a user-created field of type GUID if the table already contains one.	Read-only

TABLE 2 ***Properties Added to a Database.*** *In addition to these properties, the database object has two replication methods: MakeReplica and Synchronize.*

nization schedule using DAO has some advantages. For example:

- It allows you to deliver a VB-only solution. You don't have to install and use Replication Manager (see the accompanying sidebar, "Tools for Managing Jet Replication" for more information on managing replicated databases).
- You have more flexibility in scheduling. Replication Manager uses a day-of-week schedule and allows synchronization only every 15 minutes.
- You can create a hybrid synchronization system based on both a regular timed schedule and update volume.

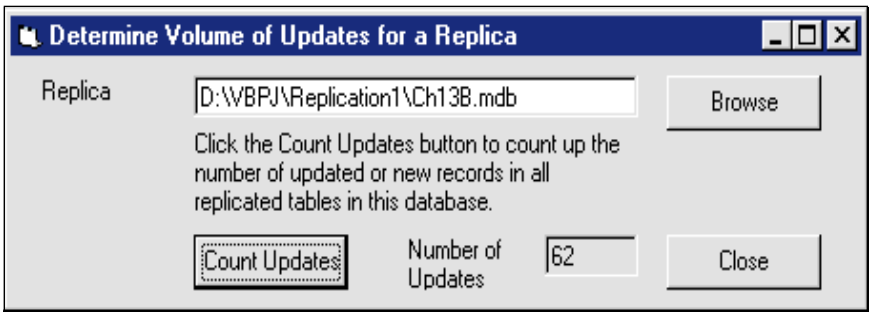


FIGURE 3 ***Count the Updates.*** *The UpdateVolume form counts the number of updated records since the last synchronization exchange. This number could be used to initiate a synchronization exchange only after so many updates were made to a replica.*

Tools for Managing Jet Replication

VB is great for creating applications that manipulate replicated databases, but you may find it easier to use Access 95 or Replication Manager to set up and manage replicas. You can use the Access 95 menus to easily replicate and synchronize databases. You can also use Access to resolve synchronization conflicts and transfer the design master status from one replica to another. The replication-related commands are located under Tools selection of the Replication submenu.

The Access 95 Developer's Toolkit includes a utility called Replication Manager (see Figure A). Like Access, you can use Replication Manager to replicate and synchronize replicas and transfer the design master status from one replica to another. In addition, Replication Manager includes a facility for creating a regular synchronization schedule that a companion program called the Transporter uses to perform unattended synchronization exchanges. Replication Manager also includes tools for reviewing synchronization logs and for easing the management of remote replicas. —P. L.

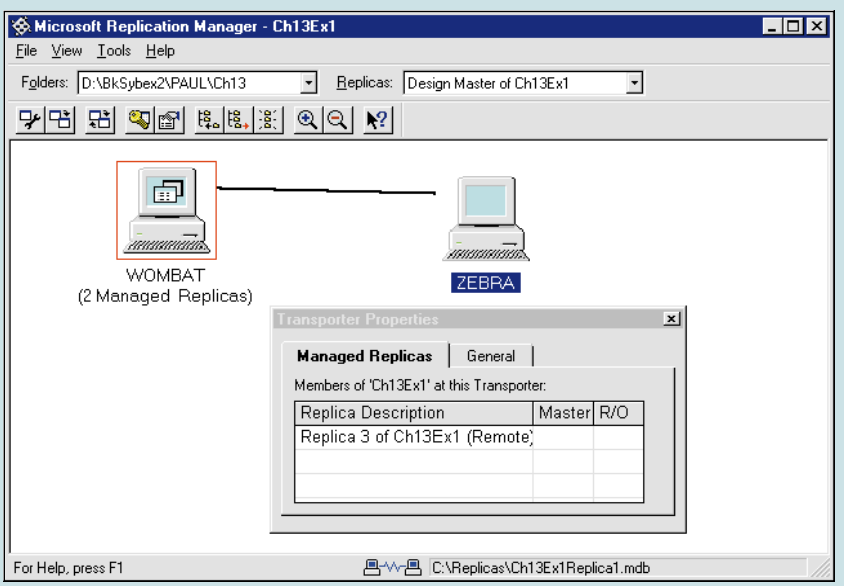


FIGURE A ***Remote Management.*** *Replication Manager is used to manage a replica set across a WAN. The local machine, Wombat is exchanging updates with a replica on the remote machine, Zebra.*

If you decide to implement a synchronization system using DAO, you'll need to decide how the process will be driven. Most likely you'll employ a hidden form that's automatically loaded when the application is started with code attached to the form's Timer event.

This form would likely follow a schedule that was stored in a table in the database. But where will this hidden form and table reside? Should it be part of the normal application that runs on each desktop, or should it perhaps run only on selected desktops?

The answer depends on your chosen synchronization topology. In any case, you don't want all replicas attempting to synchronize with each other at the same time.

One solution might be to keep this part of the application separate from the rest of your application. This application could run off the file server or the database administrator's desktop.

You may wish to implement a synchronization system based on update volume rather than (or in addition to) a regular schedule. You can determine the update volume by counting the number of records in each replicated table in the database where the s_Generation field equals 0. This number represents the number of records that have been updated or added since the last synchronization exchange.

I've created a sample form, UpdateVolume, that demonstrates this technique (see Figure 3). Open this form, select a replica, and click the Count Updates button. After a brief delay, the number of updated records in the replicated tables in the database is displayed in a text box on the form.

This example doesn't do anything with the value, but once you've determined the number of updated records, you can easily decide whether it's time to synchronize.

Replication is an exciting new addition to the Jet engine. Jet 3.0's support for replication increases your available options for sharing data. Now it's your turn to dig in and put Jet replication to work.

The code in this article is included in a sample VB project called REP1.VBP that you can download from the *VBPJ* Developer's Exchange on CompuServe (GO WINDX), The Microsoft Network (GO WINDX) and the World Wide Web (http://www.windx.com).