Database Replication in Microsoft Jet 4.0

[originally from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc2k/html/dbrepjet.asp]

Debra Dove
Microsoft Corporation

January 1999

For the latest information, see www.microsoft.com/Officedev/.

Database Replication with the Microsoft Jet Database Engine: A Technical Overview

The Microsoft® Jet database engine, version 4.0, is a 32-bit engine that provides database processing and replication functionality to a variety of applications. This document is intended for experienced Microsoft Jet users who want to understand database replication as it is implemented in Microsoft Jet and use it more effectively in their applications.

This document focuses on new functionality that is introduced with Microsoft Jet 4.0 and the suggested tools specifically enhanced for this release of Microsoft Jet replication. New functionality includes the unified treatment of conflicts and errors, column-level tracking of changes, improved priority-conflict-resolution algorithm, added replica types and visibilities, and extended replication functionality exposed through the Microsoft Jet and Replication Objects (JRO) 2.1 library. Before diving into the details of these new features, the next three major sections provide an overview of replication, its uses, and the tools for performing it.

What Is Database Replication?

Database replication is a technique you can use to support multiple users, connected and/or disconnected, of an application. Replication is the process of creating multiple copies of an application and its data to be used at locations that are not always connected to each other. Collectively, the copies are called a *replica set*. One member of the replica set must be designated the *Design Master* (where changes to the database design are made)*;* any other copy is a *replica*.

A replica set can contain only one Design Master, but it can contain as many replicas as needed. The Design Master is the only copy where changes to the database design are allowed. You can designate any global replica as the Design Master, but you must be sure that only one replica is marked as the Design Master at any time. When a database is made *replicable*, properties are set that identify the database as the Design Master.

Each replica contains a common set of *replicated objects* and any single replica can also contain *local* tables and/or queries that exist only in that replica. When you are using Microsoft® Access 2000, all Microsoft Access objects (forms, reports, data access pages, macros, and modules) are either replicable or local, which must be specified prior to making the database replicable. Replicas are *synchronized* to merge design (schema) and data changes between them. Microsoft Jet replication ignores local tables and/or queries and local Microsoft Access objects during synchronization; only changes to replicable objects are synchronized.

All replicas (including the Design Master) include a number of additional fields and tables that Microsoft Jet uses to manage the replicated application. For example, one field in each record in a replicated table has a globally unique identifier (GUID) that distinguishes the record from every other record in the replica set. The s_Lineage field records the *lineage* (replica ID/version pair) of each record and if changes are tracked at the column (field) level, the additional field s_ColLineage records the *column lineage* (replica ID/version pair) of each field. When a record is updated, the version number for that replica's record in the lineage and column lineage is incremented.

Any copy—whether the Design Master or a replica—can update the data. This is called a *multimaster data update* design; it permits a fully distributed system where data updates are not centralized. A column (cell) is the most basic unit of information recognized in replication when the **ColumnLevelTracking** property is set on the table. In this case, nonconflicting fields are merged when changes are transmitted during synchronization. When the **RowLevelTracking** property is set on the table, if any field in a row (record) is modified, the whole record is marked as changed. Therefore, the whole record is updated when changes are transmitted during synchronization and individual fields are not merged. With row-level tracking, OLE Object and Memo fields are exceptions; these items—due to their potential large size—are not transmitted unless they have been changed.

Synchronization, an important part of the replication process, reconciles all data and design changes in each replica. Updating an entire replica set consists of a series of synchronizations between pairs of replicas. The *file-tracking system* in Microsoft Jet tracks and records all changes at all replicas, in preparation for updating data during synchronization. Only records marked as changed are updated when you synchronize replicas. If two replicas simultaneously update the same record at different replicas, Microsoft Jet reconciles the updates. This process might introduce a conflict, depending on whether the table involved in the synchronization is set to track changes at the column or row level. Synchronizations can be performed on a regular schedule or as often as necessary to ensure all users have current data. This means that all information will reach all replicas, but there is no guarantee changes will reach all replicas within any specified amount of time. When using Microsoft Jet, application designers must allow for this in their designs.

Microsoft Jet uses *incremental replication*. Therefore, during a single synchronization between two replicas, the only updates made are those resulting from changes made since the last synchronization. This provides significant benefits over methods of data distribution that transmit the whole database whenever new data or objects require distribution. Each record in a replicable database has a *generation counter*; Microsoft Jet uses this field to control incremental exchanges.

Not all applications that support Microsoft Jet use its features in the same way. For example, although Microsoft Excel cannot replicate a database, it can update a database replicated by another product. Microsoft Jet monitors the changes made by Microsoft Excel, Microsoft Visual Basic®, or Microsoft Access to a replica and updates these changes when you synchronize the replicas. These products support Microsoft Jet either directly or through Jet and Replication Objects (JRO), which is a component of ActiveX® Data Objects (ADO), or Data Access Objects (DAO). Replication is a feature of the Jet database engine, not of the specific applications that include the Jet database engine.

Microsoft Windows® 95 or 98, Microsoft Windows NT® Server, Windows 95 peer-to-peer, and Novell NetWare networks are platforms that support Microsoft Jet. Banyan VINES and LANTastic do not support Microsoft Jet replication. Microsoft Jet replication presumes that files are named in accordance with Microsoft and Novell network file-naming conventions, which are different from the file-naming conventions used by Banyan and LANTastic.

When Should I Use Replication?

Replication is well-suited to distributed systems that focus primarily on adding new records rather than on updating existing records. Sales representatives who visit customer offices, parcel delivery drivers, and inspectors who visit a variety of construction sites are all examples of users who might benefit from replication. There are many tools and techniques for implementing replication. Some factors to consider when choosing a replication technique are:

- How quickly you need data synchronized across all sites.
- Budget for hardware, software, and communication services.
- Overall system-reliability requirements.

The best candidates for replication are applications that can tolerate some latency in data updates in exchange for a robust configuration that can allow updates from any replica and that supports users who are only occasionally connected. This flexibility means the system can work more effectively, potentially improving business performance. Using flexible, low-cost, off-peak asynchronous communication links and asynchronous data duplication provides "real-time-enough" updates without the expense and vulnerability of full-time connections between all nodes. When the application's users are connected, it might be through a direct connection on a local area network (LAN) or wide area network (WAN), or through the Internet or an intranet. Data can be exchanged on a LAN, a WAN, or the Internet.

Microsoft Jet replication is a good solution if you want to:

- Share data among users at multiple remote locations.
- Automate the distribution of new features and updates to multiple users.
- Use different machines for system queries and transaction processing (this can improve transaction-processing performance).
- Automatically back up data without disabling the system (each replica serves as a backup, so a separate backup procedure is not needed).

If your multiuser application requires very frequent data updates, if it will update a large number of records at one or more sites, or if it is critical for data changes to be very quickly obvious to other users, Microsoft Jet replication may not be the best solution for you to use. Applications in these categories are better served by two-phase commit solutions. In a two-phase commit, replicas are connected all the time and an update at any one site will be accepted only if agreement is immediately given from all other sites. It's called a two-phase commit because the initial phase is notification of a proposed update sent to all replicas, and the second phase is the actual update only when all sites have agreed (that is, committed) to the update.

Tools That Implement Replication

You can use several tools to implement Microsoft Jet replication. These tools allow you to convert a database to replicable format, identify a replicable database as the Design Master or a replica, initiate synchronization of the replica set, and a variety of other management tasks. You can use the following tools to implement Microsoft Jet replication:

- Microsoft Access 2000 running under Windows 95 or 98, Windows NT Server, or Windows NT Workstation.
- Microsoft Replication Manager, available with Microsoft Office Developer 2000.
- Briefcase replication.

- JRO programming, which is available on Windows 95 or 98 or Windows NT Server or Workstation version 4.0 or later.
- DAO programming, available on Windows 95 or 98 or Windows NT Server or Workstation version 3.51 or later.

The first three of these tools provide an easy-to-use visual interface, while the last two enable programmers to build replication directly into their applications.

Microsoft Access Replication Commands

The **Replication** submenu on the **Tools** menu in Microsoft Access provides several commands to help you create a replica, synchronize a replica with another member of the replica set, resolve synchronization conflicts, and recover a replica set's Design Master. For more information about these commands, refer to the Help system provided with Microsoft Access.

> **Note** Replication is not installed by default when Microsoft Access 2000 is installed by using the **Typical** option. However, the replication components will be installed upon first use from the Access user interface. To install the replication components when you are installing Access, choose the **Customize** option and select the Database Replication feature.

Microsoft Replication Manager

Microsoft Replication Manager is included with Microsoft Office Developer 2000. It is another tool that lets you use replication to administer a distributed application. It offers more functionality and more features than the Microsoft Briefcase. Key features include:

- Graphical user interface and tools for system development and maintenance.
- Visual representation of replica set topologies, which greatly assist system management.
- Activity reports to assist with troubleshooting and synchronization reports to help monitor the activity among replicas.
- Property dialog boxes that provide valuable information about components.
- Administration commands controlling the conversion, creation, location, and management of replicas.
- Scheduled exchanges between replicas administered through a graphical user interface, plus immediate synchronization with remote replicas through point-and-click commands.
- Direct and indirect exchanges provide additional support for "rarely connected" users. Laptop users may specify a networked file location where exchange information may be deposited for later processing. Synchronization is optimized over a LAN or WAN for indirect exchanges, and optimized for direct exchanges when a direct connection can be established between local replicas.
- Synchronization over the Internet.
- Exchanges can be configured to only send data, only receive data, or send and receive.

Synchronizer

The Synchronizer is an agent you can use with the Replication Manager to provide scheduled background exchanges between replicas. These exchanges can be made while two replicas are directly connected, or through a file-system transport (indirect and Internet) that does not require a direct connection for the exchange. In either type of transfer, the Synchronizer collects the changes at one replica and transmits them to other replicas. If the file-transfer system is used, one replica must deposit changes in a temporary file. The Synchronizer, initiated from the target replica, collects the updates at a later time and applies them to the target replica. This is a great benefit for rarely connected users; they can post changes whenever convenient rather than depending on an available connection. The Synchronizer is required for both indirect and Internet synchronizations.

> **Note** When running the Synchronizer to schedule exchanges between replicas, you may find it advantageous to disable the Windows 95 or 98 System Agents. Disk compression and defragmenting can make heavy demands upon your PC that prevent scheduled replication exchanges from completing in a timely fashion.

Indirect synchronization

To use indirect synchronization, you must install and configure the Replication Manager on both computers participating in the exchange. In contrast to direct synchronization, which opens both members of the replica set involved in the exchange, indirect synchronization relies on a series of message exchanges between replicas. The Synchronizer managing each replica collects changes into one or many "message files" (*.msg), which are then sent to a shared folder, called a dropbox, which is being used by the partner Synchronizer. The partner Synchronizer then processes these message files. Additional message files continue to be sent back and forth between Synchronizers until the synchronization is completed.

Internet synchronization

Internet synchronization is a way to exchange the data in a replicated database over an Internet or intranet connection, requiring only the replica on the Internet server to have Replication Manager installed and configured. Internet synchronizations also use a dropbox, as in indirect synchronization, but the dropbox on the Internet server is used for all replicas in the replica set. Microsoft Jet 4.0 introduces several new features to Internet synchronization, including:

- Support of the HTTP 1.1 protocol:

- Eliminates the reliance on FTP for synchronization.
- Enables support through a proxy server.
- Performance enhancements to reduce transfer times.
- The addition of several registry keys to control Synchronizer timeouts.

Briefcase Replication

The Briefcase is an accessory available in Windows 95, Windows 98 (the feature is not installed by default on a new installation), and Windows NT Workstation or Server version 4.0. When Microsoft Access is installed on your computer, you can use the Briefcase as a replication tool by simply dragging an .mdb file from the Windows Explorer onto the Briefcase icon on the Windows desktop. Your database is converted into replicable format and becomes a member of your replica set. The Briefcase menus include commands to synchronize the replicas.

When you install Microsoft Access, the Setup program adds class ID (CLSID) entries for .mdb files and for the Briefcase reconciler to the Windows registry. (Only Microsoft Access installs the Briefcase reconciler.) The reconciler includes the code required to support replication and synchronization. When you drag an .mdb file onto the My Briefcase icon, Windows recognizes the class ID and responds by calling the reconciler. The reconciler converts the database into a replicable form, then gives you the option of specifying the location of the Design Master; you can use this feature to designate either the replica in your Briefcase or the replica in the original location the Design Master. When you synchronize the replicas, the Briefcase calls the reconciler to merge the replicas. With Briefcase replication, synchronization cannot be scheduled; it occurs only when the Update command is clicked and only between the current member and the specified member.

> **Note** Before converting the database, Microsoft Jet asks if you want to make a backup. If you anticipate that any users will need to use a nonreplicable version of the database, it's a good idea to make this backup. Also, if you attempt to convert an .mdb from a previous version, you will be asked to first convert the database to Microsoft Access 2000.

You can use the Briefcase with files other than .mdb files, and with applications other than Microsoft Access. However, doing so will not call the Microsoft Jet replication code; it will call the default Briefcase code instead. If you use the Briefcase when Microsoft Access is not installed or with a non-Microsoft Jet database, dragging a file into the Briefcase is equivalent to simply copying the file into the Briefcase—there is no conversion to replicable format. Therefore, when you update files on your main computer with files from the Briefcase, the Briefcase simply copies over the original file—changes to data and objects are not merged, they are overwritten.

Jet and Replication Objects (JRO)

In any application that supports ADO, JRO provides a programmatic interface to replication functionality in Microsoft Jet databases. Methods and properties within JRO can be used to make a database replicable, change the replicability of objects within the database, create replicas, synchronize replicas, and manage certain properties within a replicated database. The new features in Microsoft Jet replication 4.0 are exposed only in JRO. They include setting a replica's priority, the ability to execute an indirect synchronization in code, setting a replica's visibility, and much more.

Data Access Objects (DAO)

In applications that support Microsoft Visual Basic for Applications (VBA), DAO provides a programmatic interface to replication functions. Microsoft Jet includes a variety of extensions to the DAO programming interface. These extensions allow developers to convert a database to replicable format, make additional replicas, synchronize replicas, and manage certain properties of a replicated database. These properties include the description of a single replica or of a replica set, the ID of a particular replica or of the replica set's Design Master, the default replica to be used in an exchange, and the local/global property of each object in the database.

> **Note** DAO does not support any new features that are introduced in Microsoft Jet replication 4.0. In all new feature cases, default values are used. For example, when you are creating a replica, a replica with global visibility is always created.

Conflict Resolution

Microsoft Jet uses a merge reconciler that merges individual changes from two replicas involved in a synchronization, so that after the synchronization is complete, the resulting data set is a combination of the data in both replicas. This merge process may cause conflicts, which need to be resolved. Note that in Microsoft Jet 4.0 replication, there is no longer a distinction between a conflict and an error; both are just conflicts. The possibility of a conflict might depend upon whether the table involved in the synchronization is set to track changes at the column or row level. Once it is determined that a conflict exists, an algorithm based on the priority assigned to each of the replicas is used to select the winner and the loser. Each of these steps is described in the following section.

Unified Treatment of Conflicts and Errors

In previous versions, Microsoft Jet replication differentiates between synchronization conflicts and synchronization errors.

Synchronization conflicts occur when two users update the same record in two different databases in a replica set. Synchronizing the two databases succeeds, but only one of the two sets of changes are applied to both databases. Thus, one replica "loses." Synchronization errors occur when a change to data in one database in a replica set cannot be applied to another database in the replica set because the change would violate a constraint, such as referential integrity or uniqueness. Synchronizing the two databases succeeds, but the records that caused the error are not exchanged. Therefore, each replica retains the original data for that record.

With Microsoft Jet 4.0 replication, the events that cause synchronization conflicts and synchronization errors are both viewed simply as synchronization conflicts, and a single mechanism is used to record and resolve them, making resolution of such problems easier. When a conflict occurs, a resolution algorithm is used to determine a winner and a loser. The winning record is placed in the table in both replicas. The losing record is placed in a "conflict table" and replicated to both replicas. The conflict table is a replicated table named *TableName*_Conflict (where *TableName* is the name of the table where the conflict occurred). Microsoft Access automatically invokes an application to assist the user in resolving entries in conflict tables. The new Microsoft Replication Conflict Viewer can then be used to reconcile and resolve synchronization conflicts.

> **Note** The same Microsoft Replication Conflict Viewer can be used with either SQL ServerÂ™ 7.0 or Microsoft Jet 4.0 replicable databases.

Column-Level Tracking vs. Row-Level Tracking

In previous versions, conflicts are determined at the row (record) level. In other words, if two users in two different replicas changed the same record for customer, but each changed a different column in the record, the two records would conflict when the replicas were synchronized. Suppose that one user changed the zip code and the other changed the phone number. Note that although the changes themselves do not conflict because they were made in two separate fields, a synchronization conflict would still occur as the changes are being tracked on a record-level basis.

In Microsoft Jet 4.0, replication can now track data updates at the column (field) level. Column-level tracking lets you merge the same two records and only reports a conflict if simultaneous changes have been made to the same field. Thus, in the above scenario, there would no longer be a synchronization conflict since the two users changed the values of different fields. A table that has its **ColumnLevelTracking** property set to **True** will significantly reduce the potential for conflicts and simplify the maintenance of replicated databases, if different users frequently edit the same data.

> **Note** Microsoft Jet column-level tracking will work in conjunction with the corresponding Microsoft SQL Server 7.0 capability when Microsoft Jet and Microsoft SQL Server replication are used together.

Column-level tracking is the default behavior for all tables created in Microsoft Jet 4.0. Column-level tracking does create a small performance hit, due to the added system columns and compare logic, and therefore should not be used in cases where users are doing large updates to isolated tables, that is, tables other users would not be updating. Selecting the **Row Level Tracking** check box (in the Database window, right-click the table to open the ***TableName*** **Properties** dialog box) prior to making the table replicable will set a table to use row-level tracking. A table's tracking behavior cannot be changed once the table has been made replicable. If you would like to change the tracking of a replicable table, you will need to make the table local (in the Database window, right-click the table and clear the **Replicable** check box and click the **Apply** button), select or clear the **Row Level Tracking** check box, then make the table replicable again.

> **Note** For replicas converted from an earlier version of Microsoft Jet, the default behavior is to retain the previous behavior, that is, row-level conflict resolution.

Priority-Based Conflict Resolution

In previous versions, synchronization conflicts were resolved based upon an algorithm whereby the most often changed copy of a record won. This algorithm worked, but was unsophisticated and confusing.

Microsoft Jet 4.0 introduces an algorithm whereby replicas in a replica set are assigned priorities and the highest priority replica wins in the case of a synchronization conflict. Where priorities are equal, the replica with the lowest replica ID wins.

> **Note** The priority-based conflict-resolution algorithm will work in conjunction with the corresponding Microsoft SQL Server 7.0 capability when Microsoft Jet and Microsoft SQL Server replication are used together.

Replicas are assigned a priority, a real number between 0 and 100, inclusive, when the replica is created. At the time a database is made replicable, the Design Master's priority is set to 90. The default priority of additional replicas is 90 percent of the parent replica's priority. You can easily specify another priority value either by using the **Create Replica** dialog box in Access or by using the **CreateReplica** method in JRO. Users must have dbAdminister privileges in order to specify that the priority of a replica be greater than the priority of the replica from which it is being created. Using the file system to make a copy of a replica will create a replica with the same properties and priority as the source replica.

> **Note** All replicas upgraded from an earlier version of Microsoft Jet are given an identical priority value of 90. Replicas created through the **MakeReplica** method in DAO have a default priority equals to 90 percent of the parent replica's priority.

Historical priority

One complicated concept pertaining to priority is that of historical priority. The easiest way to describe historical priority is through an example. Suppose you have three replicas: ReplicaA with a priority of 100, ReplicaB with a priority of 95, and ReplicaC with a priority of 90. The users of all three replicas make a conflicting update to the same record. When ReplicaA synchronizes with ReplicaC, ReplicaA will win the conflict because it has a higher priority. When ReplicaC synchronizes with ReplicaB, ReplicaC will win the synchronization conflict because the record involved actually came from ReplicaA, which has a higher priority than ReplicaB. The benefit to using the historical priority is that changes made at the highest priority replica never lose during synchronization if there is a conflict.

Conflict Types

The following summarizes the types of synchronization conflicts that can be encountered:

> **Note** Design changes are always processed before data changes. Also, all conflicts, except update-delete and locking, are resolved by using the priority of the replicas involved.

- **Simultaneous Update**—The most frequent type of conflict is when two users simultaneously update data in the same record or field, depending upon the tracking level that has been set for the table.
- **Update-Delete**—Microsoft Jet 4.0 replication has retained behavior from Microsoft Jet 3.*x*, in that delete actions are always processed. This means that, no matter what the replica's priority is, the deleted record will always win in the case of a conflict and be deleted. However, in Microsoft Jet 4.0, the record that is about to be deleted is checked to see if there are updates that were unknown prior to the delete occurring. If this is the case, the updated record is logged to its appropriate conflict table.
- **Unique Key**—Two records contain the same key value, even though only unique values are permitted.
- **Table-Level Validation**—A record contains a field value that does not meet a table-level validation rule.
- **Referential Integrity**—There are three kinds of referential-integrity conflicts:
- **On Delete**—The primary key record has been deleted in another replica, and therefore the foreign record has been rejected.
- **On Update**—The primary key record has been updated in another replica, and therefore the foreign record has been rejected.
- **Foreign Key**—A foreign-key violation resulted from an invalid primary key record that was involved in another replication-conflict type.
- **Locking**—The record change could not be applied during synchronization, because another user locked the table involved. The synchronization fails, but no conflict is logged. The solution is to try the synchronization again when the table is not locked.

> **Note** Partial replicas receive conflicts associated with all the records in their filters, including newly added records that are added to a partial replica during synchronization and might have conflicts associated with them.

Creating a Custom Conflict-Resolution Function Using JRO

In Microsoft Jet 4.0, all conflicts and errors are logged as conflicts in conflict tables, which are replicated throughout the entire replica set. This is a decentralized conflict model where any user can view a conflict and change the outcome of the conflict by using the Microsoft Replication Conflict Viewer, automatically invoked when a conflict is encountered. If you prefer to automate conflict resolution for your application, you can write a custom function to resolve conflicts, then override the built-in application by using JRO to set the **ConflictFunction** property to the name of your custom function.

When you enhance the Microsoft Jet algorithm with your own VBA function, Microsoft Jet will still initially resolve conflicts by using its own algorithm, but you can use your code to manipulate the results. The **ConflictFunction** property can be set only in the Design Master, since it is considered a design change to the database.

Here is a simple function that uses the **ConflictTables** property in JRO to display the name of any table with a conflict in the Debug window:

```
Public Function Resolve()     'Find tables with conflicts
    Dim conn As New ADODB.Connection
    Dim rs As New ADODB.Recordset
    Dim repRW As New JRO.Replica

    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data source=C:\demo\NWindRW.mdb;"
    repRW.ActiveConnection = conn
    Set rs = repRW.ConflictTables
    rs.MoveFirst
    While NOT rs.EOF
        Debug.Print rs(0) & " has a conflict and"
        Debug.Print rs(1) & " is the name of the a conflict table."
        rs.MoveNext
    Wend
End Function
```

You can customize this function to resolve conflicts according to your own business rules. For example, if your application includes a special Date/Time field that is always updated when a record is inserted or edited, your VBA function could use this field to select the record with the most recent update as the winner. However, because your code will replace the Microsoft Replication Conflict Viewer, your function must resolve all conflict situations.

> **Note** For more information about replication methods and properties exposed through JRO, see the Jet Replication Objects Help file (Msjro.chm).

Preventing Record Deletion in Replicas

Many replication users requested that a special type of replica be provided that would permit a replica to be defined as one where a user could not delete records. This attribute is supported in Microsoft Jet 4.0 replication.

An example of when this attribute would be useful is when a full replica is given to a salesperson. The salesperson might be tempted to delete all customer records that were not of interest. However, it would be unfortunate, to say the least, if these deletes were then propagated to all replicas in the replica set. While you can prevent this from happening by setting the appropriate security permissions, that solution requires you to understand Microsoft Jet security and to set the appropriate permissions on every table. Using the prevent deletes replica type is a simpler solution.

> **Note** A prevent deletes replica can be created only through the Access user interface, by selecting the **Prevent deletes** check box in the **Create Replica** dialog box.

Replica Visibility: Global, Local, and Anonymous

Microsoft Jet 4.0 replication defines three degrees of visibility for replicas. A replica's visibility can be defined as global, local, or anonymous.

Global replicas have the same visibility as replicas that are created in Microsoft Jet 3.*x*. A global replica is a replica that can synchronize with all other global replicas in the replica set. A global replica can also synchronize with any replica created from it, with some exceptions. (For details, see the following paragraphs about local and anonymous replicas.) The ability of a global replica to synchronize with any other global replica in the replica set is possible because the information for the replicas is stored in the system table MSysReplicas. When a Jet database is initially converted to a replica and becomes the Design Master of the replica set, its default visibility is global. By default, any new replica created from a global replica is also a global replica. From a global replica, you can create a global, local, or anonymous replica.

Local replicas can synchronize only with their parent, a global replica, and are not permitted to synchronize with other replicas in the replica set. This way, you can easily control the topology in a replica set. For example, local replicas can be used to enforce a star topology at individual sites where you want to ensure that synchronization between the sites goes through a global hub at each site. Other replicas in the replica set will not be aware of the local replica. The parent replica can schedule synchronizations with a local replica by using Replication Manager. The parent replica proxies any replication updates and conflicts for the local replica. Local replicas always have a priority of 0. This value cannot be changed.

Anonymous replicas are important for the Internet scenario where you do not want to keep track of every download of the database. An anonymous replica can synchronize only with its parent, a global replica. Anonymous replicas are very similar to local replicas, with one big exception. The anonymous replica information is not permanently stored in the system table MSysReplicas. These are replicas that, for example, subscribe through the Internet and do not have any particular identity in the parent replica, but instead proxy their identity for updates through the parent replica. By removing the information about the anonymous replica after a period of time of inactivity, the overall size of replicas in the replica set is reduced if many replicas in your set are anonymous replicas. In addition, removing information about anonymous replicas helps to keep out unnecessary topology information about replicas that participate only occasionally. The parent replica, always a global replica, cannot schedule synchronizations with an anonymous replica. Anonymous replicas always have a priority of 0. This value cannot be changed.

> **Note** It is recommended that anonymous replicas not be managed by using Replication Manager. If a replica needs to be managed for scheduled synchronizations, a global or local replica should be used.

Other limitations:

- If the parent replica is moved through any method other than the **Move Replica** command in Replication Manager, it will receive a new replica ID and will no longer be visible to its local or anonymous replicas. Thus the replicas will no longer be able to synchronize.
- Local and anonymous replicas are not supported for Briefcase replication.
- Local and anonymous replicas cannot be converted into a Design Master.
- You can create replicas from a local or an anonymous replica. The new replica will inherit the same properties as the original replica, except for the replica ID. Therefore, from a local replica, you can create only a local replica; likewise, you can create only an anonymous replica from an anonymous replica.
- A Microsoft SQL Server 7.0 global (publishing) replica will be able to create Microsoft Jet replicas with any of the three degrees of visibility. However, a Microsoft SQL Server 7.0 local or anonymous replica cannot create Microsoft Jet replicas.

New Microsoft Access Project-Storage Format

In previous versions, individual Microsoft Access objects (for example, forms, reports, macros, and modules) are identified and tracked, allowing changes to individual objects to be synchronized. In other words, if a Microsoft Access form is changed in the Design Master replica and no other objects are changed, only the changes to the form are replicated when the replica set is synchronized.

However, in Microsoft Access 2000, all Microsoft Access objects (for example, forms, reports, data access page links, macros, and modules) are stored in a single binary large object (BLOB) within the database file. (Note that if you make a data access page link replicable, the data access page .htm file it points to must be stored on a network location so that the page can be accessed from the Design Master and all replicas in the replica set.) In this format, the individual objects cannot be identified or tracked by Microsoft Jet replication. Therefore, if the Microsoft Access project in the Design Master is made replicable, all Microsoft Access objects are made replicable and synchronized if any single object is modified. However, you can choose to not make the Microsoft Access project replicable prior to making the database replicable by setting the **ReplicateProject** property in the custom database properties to **False**. In this case, the Microsoft Access project in each of the replicas is not replicable, and all Microsoft Access objects created in a replica are local.

> **Note** Each replica in a replica set must be individually upgraded. When upgrading, all Microsoft Access objects (forms, reports, macros, and modules) stored in the Design Master are made replicable. Local Microsoft Access objects stored in each replica, except those in the Design Master, will be lost unless they are first imported into the Design Master prior to upgrading. Local tables and queries remain local in all replicas of the replica set.

Microsoft Jet/SQL Server Bidirectional Replication

In previous versions of Microsoft SQL Server and Microsoft Jet, data could be replicated to a Microsoft Jet database, but changes made in the Microsoft Jet database could not be used to update the Microsoft SQL Server database. With Microsoft Jet replication 4.0 in combination with Microsoft SQL Server 7.0, support for bidirectional replication between Microsoft Jet and SQL Server has been implemented. Not only can changes made to data in a Microsoft SQL Server database be replicated to a Microsoft Jet database, but changes to the data in a Microsoft Jet database can be synchronized to and reconciled with a SQL Server database.

A mixed replica set containing both SQL Server replicas and Microsoft Jet replicas is required for this feature to work. To begin creating this replica set, you need to first create a SQL Server database. If you are starting with a Microsoft Jet database, you can use the Upsizing Wizard in Microsoft Access 2000 to create a SQL Server version of your database. Once your SQL Server database is made replicable, you can start creating SQL Server or Microsoft Jet replicas.

There are some limitations:

- Only data can be replicated between Microsoft Jet and Microsoft SQL Server. Microsoft Access objects (for example, forms, reports, data access pages, macros, and modules) cannot be replicated to Microsoft SQL Server and will continue to reside only in the Microsoft Jet database.
- The only topology supported in Microsoft Jet/Microsoft SQL Server replication is "hub and spoke." The Microsoft SQL Server database is always the hub. The Microsoft Jet replicas at the spokes cannot synchronize with other Microsoft Jet replicas; they can synchronize only with their Microsoft SQL Server hub database.

For more information about replicating data between Microsoft SQL Server and Microsoft Jet databases, see "Implementing Merge Replication to Access Subscribers" in the documentation provided with SQL Server.

Partial Replicas

A partial replica is one that contains a subset of data. You can use either the Partial Replica Wizard or VBA code to create a partial replica. To run the Partial Replica Wizard in Access, point to **Replication** on the **Tools** menu, then click **Partial Replica Wizard**.

A partial replica is defined by a filtering expression on a table, similar to a WHERE clause and relationship filters. For example, to create a partial replica with customers from California, you would specify `"Region = 'CA'"`. If you wanted to see all the orders for customers in California, you would also set the relationship filter (either by using the Partial Replica Wizard or by using JRO, as shown in the PartialRep code sample below) on the CustomersOrders relationship. You cannot use user-defined or aggregate functions, nor can you prompt the user at run time for parameter values. By using VBA, you can apply restrictions to any number of tables. The Partial Replica Wizard limits you to placing restrictions on a single table.

You must enclose date variables with the number sign (#). For example, to select orders placed after March 31, 1997, and before December 31, 1998, enter the following:

```
[Order Date] > #3/13/95# AND [Order Date] < #12/31/96#
```

You must surround the contents of Text and Memo fields with quotation marks. For example, to specify Jane Doe's name, enter the following:

```
[FirstName] = "Jane" AND [LastName] = "Doe"
```

To enter numeric values, use the field's name and the value. For example, to enter 1 as the category ID, enter the following:

```
[CategoryID] = 1
```

> **Note** Multiple filters are OR'ed together. For example, if you set a filter of `"Region = 'CA'"` for the Customers table and a filter of `"Value > $1000"` for the Orders table, the result will include all records for customers from California *plus* all orders with a value greater than $1,000. You would *not* get orders of over $1,000 only for customers from California.

JRO provides a simpler way than DAO to create more sophisticated partial replicas by applying restrictions to any number of tables, through filter and criteria properties. The following example shows how you can create a partial replica, set a filter on a table and a filter on a relationship, and populate the partial replica with data:

```
Public Sub PartialRep()

' This code demonstrates how to create a partial replica with a
' relationship filter and a table filter.
' NOTE: PopulatePartial requires an exclusive connection.

    Dim repMaster As New JRO.Replica
    Dim repPartial As New JRO.Replica

    repMaster.ActiveConnection = _
        "C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb"

    ' Northwind.mdb is already replicable.
    repMaster.CreateReplica "C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Partial of Northwind.mdb", _
        "Partial Replica of Northwind", jrRepTypePartial

    Set repMaster = Nothing

    ' PopulatePartial requires an exclusive connection to the database.
    repPartial.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=C:\Program Files\Microsoft Office\" & _
        "Office\Samples\Partial of Northwind.mdb;" & _
        "Mode=Share Exclusive"

    repPartial.Filters.Append "Orders", jrFilterTypeRelationship, _
        "CustomersOrders"

    repPartial.Filters.Append "Customers", jrFilterTypeTable, _
        "Region = 'CA'"

    repPartial.PopulatePartial "C:\Program Files\" & _
        "Microsoft Office\Office\Samples\Northwind.mdb"

    Set repPartial = Nothing
End Sub
```

You should call the **PopulatePartial** method under any of the following circumstances:

- Before synchronizing a full and a partial replica for the first time. Calling the **PopulatePartial** method ensures that the partial replica has all the system information required. If you simply create an empty partial replica and do not call the **PopulatePartial** method before synchronizing, you will see the message "the filters are not synchronized", even if no filters are set on the partial replica. The **PopulatePartial** method is required to initialize the replica with internal system information before the first synchronization.

  > **Note** The **PopulatePartial** method works only with direct connections; it does not support indirect synchronization such as dropbox (file transfer) synchronization or Internet synchronization by using an HTTP or FTP folder.

- Whenever you modify the filter for a partial replica, the **PopulatePartial** method removes any records from the partial replica that do not comply with the new filter. Simply using the **Synchronize** method after changing the filters does not remove these "orphaned" records.

Synchronizing between full and partial replicas may require careful attention. Say you create a replica set with three members: Full_1, Partial_1, and Full_2. Now assume Full_1 synchronizes with Partial_1, and because Partial_1 has a filter, it gets only a subset of the changes made at Full_1. Now Partial_1 synchronizes with Full_2. Obviously only the subset of the changes stored in Partial_1 can be sent to Full_2. It would be a mistake to assume that Full_2, having successfully completed the exchange, has all the updates from Full_1. The only way to guarantee that Full_2 has all the changes is to synchronize it with Full_1. When synchronizing full and partial replicas, the best process is to treat partial replicas as "leaf" nodes. That is, designate them as the end of a synchronization chain. Doing this also increases the efficiency of synchronizations, because the protocol that ensures correct propagation of updates between partial and full replicas may result in redundant data exchange if the partial replicas are not moved to the end of the synchronization chain.

Partial replicas introduce a number of subtleties for deleting or updating records when referential integrity and cascading updates or deletes are used in the same application. Consider what might occur in a simple database with two tables: Customers and Orders. Referential integrity is enforced between these tables, but cascading updates and deletes are not enabled. Consider Customer A, who has Orders records in the full replica. Now assume there is a partial replica with only the Customers table. If Microsoft Jet allowed the records pertaining to Customer A to be deleted at the partial replica, this delete would fail when it was sent to the full replica, because there would be existing records for Customer A.

To prevent this, Microsoft Jet traps attempts to update or delete primary keys records in the primary key table of a partial replica, and permits them only if:

- The primary key table has no foreign key records.

  –or–

- For each foreign key table, the relationship filter between the primary key table and the foreign key table is set.

  –or–

- The foreign key table's table filter is set to **True** (which indicates all records in the foreign key table are present).

For partial replicas, Microsoft Jet looks only at the immediate foreign key table to decide whether to permit an update or delete. However, it's important to remember the effect that cascading updates and deletes can have. Consider an example of three related tables (Customers, Orders, and OrderDetails). If cascading updates and deletes are enabled, an attempt to update a Customer record in a partial replica will also attempt to update any Order records, which in turn will attempt to update any OrderDetails records. If cascading updates and deletes are not enabled, an update to a Customer record would check the Order records and ignore the OrderDetails records.

Security

Replicated databases use the same security model as nonreplicated databases. The permissions assigned to a user's logon ID control the actions that user can take on the database.

The application designer must ensure that the same security information is available in each replica. There are two ways to do this:

- Make the *exact* same workgroup information file, System.mdw, available to each replica. The security file cannot be replicated, but it can be physically copied to each location.
- Re-create the entries for users and groups at each location in the *local* workgroup information file. To do this, copy the user and group names and associated personal identifiers (PIDs) from System.mdw into the local file. Make sure to copy the entries exactly.

A user with Administrator permission can do the following:

- Convert a nonreplicable database into a replicable database.
- Execute the **Move Replica** command for the Design Master.
- In the Design Master, make a local table or query replicable, or make a replicable table or query local.
- Modify the retention period.
- Create replicas with a higher priority than the replica they are being created from.

In addition, a user with Administrator permission can execute the **Recover Design Master** command from any replica.

> **Note** Make sure there is always at least one user with Administrator permission on the database. If the Design Master and the associated System.mdw file are destroyed (for example, through a hard disk failure), it is possible to designate another replica as the new Design Master—but only a user with Administrator permission can do this.

Registry Entries

Parameters for Microsoft Jet 4.0/Microsoft Access 2000 replication components are stored in the system registry. In addition to the entries listed here, the registry includes many entries for the Synchronizer (formerly known as the Transporter) and Replication Manager, such as the log file location, the last viewed replica, the security database, and so on, under the following subkeys:

HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\4.0\Transporter

HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\4.0\Replication Manager

Synchronizer Transport Order Registry Entry

When you execute an indirect synchronization by using the **Synchronize** method in JRO or by clicking **Synchronize Now** in Replication Manager, a "you pick" synchronization will be executed by using the Synchronizer. This means the Synchronizer will first read the registry for the order in which it should attempt to execute each type of synchronization. The default order is file system (dropbox) (1), Internet (2), then direct synchronization (3), if the first two methods fail.

These attempts do not cause any significant performance hit on synchronizations. Users can control the order of the attempts and even disable a particular synchronization type by using these new registry values.

For example, if you are always using a dial-up network, you would want to disable direct synchronization. To disable a synchronization transport type, set the corresponding registry value to 0 (zero). The default registry values are as follows:

HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\4.0\Transporter

"Priority_FS" = 1

"Priority_Internet" = 2

"Priority_Direct" = 3

> **Note** You must be very careful when disabling different synchronization transport types. If you disable direct synchronizations, Replication Manager cannot synchronize any replica of the replica set with an unmanaged replica, and you cannot use the **Synchronize Locally Managed Replicas** option, even if the replicas are located on the same machine. Special attention should also be taken when disabling indirect or Internet synchronizations. This feature is for advanced replication users who have designed their applications accordingly.

Structure of a Replicable Database

Before you can use replication, you must convert the original database to replicable format. Any of the tools listed earlier can help you do this. A database in replicable format includes a number of tables and fields that are not typically present in a nonreplicable database. When you use the tools listed earlier, Microsoft Jet automatically adds the fields and tables it needs to manage your replicated application. A database in replicable format includes:

- **User tables**—The tables an application developer constructs (for example, Products, Prices, Customers, and so on). When you convert the database to replicable format, Microsoft Jet retains your table definitions and adds additional fields it needs to manage the application.
- **System tables**—The tables Microsoft Jet requires to manage the application. These tables are generally hidden from users. When a Microsoft Jet database is converted into a replicable database, Microsoft Jet adds new system tables that record the history of exchanges between replicas, the location of other replicas in the replica set, and other information required by replication.
- **GUID**—A field that provides a 16-byte globally unique identifier (GUID) for each row in each table. A GUID is guaranteed to be unique, even if two users simultaneously construct a new row at different locations. If Microsoft Jet adds this field to your replicated tables, the default name for the field is s_GUID. If your table already includes a numeric field whose data type is Replication ID, Microsoft Jet will use the existing field instead of adding a new one.
- **Generation**—A field used to ensure that only changed data is exchanged when two replicas are synchronized. Microsoft Jet adds this field to each replicated table in your application. The default name for the field is s_Generation.
- **Lineage**—A field that tracks the version and replica number of each record in each table. Microsoft Jet uses this field to keep track of changes that have already been processed and to ensure that changes to a record are not forever sent in a circle between replicas. Microsoft Jet adds this field to each replicated table in your application. The default name for this field is s_Lineage.
- **Column Lineage**—For each table that is column-level tracked, Microsoft Jet adds the Column Lineage field. This field tracks the version and replica number of each column (field) in each replicated table. Microsoft Jet uses this field to keep track of changes and to determine whether a data conflict has occurred when multiple users have updated the same row. The default name for this field is s_ColLineage.

Additional fields are inserted in replicated tables when there are OLE links to embedded graphics or Memo fields. This is an exception to the rule that data changes are tracked on a record level for tables whose **RowLevelTracking** property has been set to **True**. Because OLE objects may be of a significant size (a bitmap image may easily be 1 megabyte [MB] or more), and therefore expensive to send over a communications line, Microsoft Jet replication sends the OLE Object field only if it has been modified.

System Tables

System tables support code within Microsoft Jet. You should not rely upon the format remaining the same between releases. Treat the descriptions of these tables as "for your information only," which may assist you in debugging certain applications.

- **MSysConflicts** stores information related to each and every conflict that has been added to its appropriate conflict table. This table is replicated to all members of the replica set.
- **MSysExchangeLog** is a local table that appears in each member of the replica set and stores information about synchronizations that have taken place between this member and other members of the replica set.
- **MSysGenHistory** stores a history of generations. It contains a record for each generation that a replica knows about. It is used to avoid sending common generations during synchronizations and to resynchronize replicas that are restored from backups. This table appears in all members of the replica set, but a process slightly different from that used with normal replicated tables merges it.

- **MSysOthersHistory** stores a record of generations received from other replicas. It contains one generation from every message seen from other replicas.
- **MSysRepInfo** stores information relevant to the entire replica set, including the identity of the Design Master. It contains a single record. This table is replicated to all members of the replica set.
- **MSysReplicas** stores information about all replicas in the replica set. This table is replicated to all members of the replica set.
- **MSysSchChange** stores design (schema) changes that have occurred in the Design Master so that they can be dispersed to any member of the replica set. The records in this table are deleted periodically to minimize the size of the table.
- **MSysSchedule** stores information for scheduled synchronization. The Synchronizer for a local replica set member uses this table to determine when the next synchronization with another Synchronizer should take place, and how to synchronize data and design changes with the other Synchronizer.
- **MSysSidetables** identifies the tables that experienced a conflict and the name of the table that contains the conflicting records. This table is visible only if a conflict has occurred between the user's replica and another in the replica set. This is a special replicable table, similar to a conflict table, which is replicated to all replicas in the replica set.
- **MSysTableGuids** relates table names to GUIDs. Table GUIDs are used in tables such as MSysTombstone as a reference to a table name stored in this table. This allows efficient renaming of tables. In addition, this table includes the level number used for ordering tables so that updates can be processed efficiently. This is a local table that is updated by the tracking layer at the Design Master and, as part of the processing of design changes, at all other members of the replica set.
- **MSysTombstone** stores information about deleted records, and allows deletes to be dispersed to other replicas. This table appears in all members of the replica set.
- **MSysTranspAddress** stores addressing information for Synchronizers and defines the set of Synchronizers known to this replica set. This table appears in all members of the replica set.
- **MSysContents** stores the information regarding records that should be included in partial replicas. This table appears only in partial replicas.
- **MSysFilters** stores the information regarding the filters that are to be applied to partial replicas. This table appears only in partial replicas.
- **MSysTranspCoords** stores the display layout of the Synchronizers and replicas used by Replication Manager.

GUIDs

Just as fingerprints distinguish one person from all other people, every object in a set of data must have a unique identifier to distinguish it from all other objects. This identifier is called a GUID (globally unique identifier) or UUID (universally unique identifier). Within a database, the primary key serves a similar purpose; it ensures that every record in a table has a unique identifier.

When you replicate a database, Microsoft Jet adds several fields to each table in the database. One of those fields, s_GUID, contains a GUID that uniquely identifies a single record.

You can use the s_GUID field as the primary key in the database. The advantage of doing so is that it virtually eliminates the possibility of duplicate keys; the potential disadvantage is that the GUID field does not convey any meaning to the user.

If you choose AutoNumber or Number as the data type for a field, you can select Replication ID as the setting for the **FieldSize** property. The result is that a GUID is assigned for the row and stored in the field. If a table already includes a field with a GUID, the s_GUID field is not added when you replicate the table. Microsoft Jet uses the existing GUID instead.

GUID generation

The question "How do I know a GUID is really unique?" is a common one. The process of generating a GUID includes numerous checks to ensure its uniqueness. GUIDs are created from:

- The network node ID.
- A time value.
- A clock sequence value.
- A version value.

Here is a sample GUID:

2fac1234-31f8-11b4-a222-08002b34c003

The hyphens make it easier to read and are used only when the GUID is displayed; they are not part of the GUID. A GUID is derived from the following components:

- *Time* is a 60-bit timestamp representing the number of 100ns ticks since October 15, 1582 AD. This means time values are valid until approximately AD 3400.
- *Version* identifies the version of the algorithm used to generate the GUID.
- *Clock sequence* accounts for loss of continuity of the clock, for example when a clock is reset.

A GUID includes the following fields

<time_low>-<time_mid>-<time_hi_and_version>-<clock_seq_hi_and_reserved>-<clock_seq_low>-<node>

where:

- The *time_low* field is set to the least-significant 32 bits of the timestamp.
- The *time_mid* field is set to bits 32 through 47 of the timestamp.
- The 12 least-significant bits of the *time_hi_and_version* field are set to bits 48 through 59 of the timestamp. The four most-significant bits are set to the 4-bit version number of the GUID algorithm being used.
- The six least-significant bits of the *clock_seq_hi_and_reserved* field are set to the six most-significant bits of the clock sequence. The most-significant two bits are set to "0" and "'1", respectively.
- The *clock_seq_low* field is set to the eight least-significant bits of the clock sequence.
- The *node* field stores the node ID. The construction of the node ID depends on whether a network card is present. If a network card is present, the node ID is retrieved from NetBIOS. The first six bytes are extracted from the synchronous adapter status NCB. This is the IEEE 802 48-bit node address.

  If a network card is not installed, the node ID is set to a 48-bit number (a 47-bit random number plus 1 bit for local usage). This number is not guaranteed to be unique, even on the generating machine, but is unlikely to be duplicated on another machine. However, because GUIDs are time + sequence, this is a reasonable approximation for a local GUID. The node ID returned is explicitly made into a multicast IEEE 802 address so that it will not conflict with a "real" IEEE 802-based node address. The LocalOnly bit contains 1 if the address was generated, 0 if it's a real IEEE 802 address. The 48-bit number will be composed of:

- The computer's name.
- The value of the performance counter.
- The system memory status.
- The total bytes and free bytes on drive C.
- The stack pointer (value).
- A LUID (locally unique ID).
- Whatever random data was in the node ID buffer at create time.

GUID usage

There can be more than one field with coltypeGUID in a table, but there can be only one autogenerated column of coltypeGUID in a table, regardless of whether the table is replicable.

If an autogenerated GUID field is added to a table, GUID values are generated for all records in the table. The value of an autogenerated GUID cannot be changed or deleted.

Generations

When you convert a table to replicable format, Microsoft Jet adds a new field, s_Generation, to every replicable table in the replicated database.

The s_Generation field controls which records are sent during an exchange. When a record is modified, its generation is set to zero (0). In general, all records with generation 0 are sent during an exchange and the generation for the record is incremented to one more than the last generation, which now becomes the new highest generation.

When an exchange occurs, the sending replica knows the last generation sent to that specific receiving replica. Only records with generations higher than the previous generations or generation 0 are sent.

The receiving replica will not apply generations received out of sequence.

In some cases, Microsoft Jet replication may determine that there are too many records to be sent in a single exchange message. In these situations, the first set of records for the exchange will be of one generation and the following sets of records will be of higher generations. Therefore, it is possible that a single exchange may contain records with different generations.

Generally, there is one generation field per record. To optimize exchanges for databases that contain Memo or OLE Object fields (sometimes referred to as a BLOB, or binary large object) an extra generation field is associated with each BLOB. This generation value is set to 0, ensuring it is sent during the next exchange, but only if the BLOB is modified. If other fields in a record are modified, but not the BLOB field, the BLOB generation is not set to 0 and the BLOB is not sent with the exchange.

Lineage

The s_Lineage field is added to every replicated table in the database.

The lineage is used to determine which replicas have already received a specific update and also to determine the winner when conflicts occur. The lineage consists of a series of entries representing each replica that has changed this record. Each lineage entry consists of a shortened form (4 bytes) of the replica ID and a version number (4 bytes). The shortened version of the replica ID is the replica's nickname. The version number starts at 1 and is incremented each time the record is modified.

Column Lineage

The s_ColLineage field is added to every replicated table that has column-level change tracking specified in the database. (Note that, in Microsoft Jet 4.0, the default behavior when a table is made replicable is for changes to the table to be tracked at the column level.)

The column lineage is used to determine when conflicts occur with the granularity of the column level. The column lineage consists of a series of entries representing each column. Each column lineage entry consists of a shortened form (4 bytes) of the replica ID and a version number (4 bytes) that has made a change to that column. The shortened version of the replica ID is the replica's *nickname*. The version number starts at 1 and is incremented each time the record is modified.

Design Considerations

Microsoft Jet enforces a 4096-byte and 255-field limit to any record. These limits include fields and data you add to tables yourself, as well as any fields and data Microsoft Jet adds in the course of replication. When you design tables in an application you plan to replicate, make sure you allow for the fields Microsoft Jet is likely to add.

At a minimum, every record in a replicated table will receive s_GUID, s_Lineage, s_ColLineage (for tables where column-level tracking has been set), and s_Generation fields, which collectively require approximately 32 bytes per record. This means you should design tables with no more than 251 fields (255 maximum, minus 4) and 4,064 bytes (4,096 maximum, minus 32). This is not generally a hindrance to well-designed applications, because very few well-designed applications use either the maximum allowable fields or bytes in a single record. If your application uses Memo or OLE Object fields (BLOB fields), replication will add an additional 4-byte field per BLOB. This is the result of an exchange optimization, whereby only the long binary fields that have been modified are sent in an exchange.

Microsoft Jet replication does not support CHECK constraints. CHECK constraints are a new feature supported by Microsoft Jet 4.0, which allow more complex constraints on and across tables.

The topology of your application should be designed around the real-time load and estimated size of your database. For example, you should have more than one hub database if many replicas are synchronizing at the same time and many data changes are being exchanged during the synchronization. It is a good idea to prototype your application with real-time data and load levels.

Replica Retention Period

The replica set retention period setting controls the number of days nonsynchronized system data is retained in system tables. The retention period is established when the database is initially made replicable. If you replicate the database by using Replication Manager, JRO, or DAO, the default retention period is 60 days. If you replicate the database by using Microsoft Access or Briefcase replication, the default retention period is 1,000 days. The purging of old information happens during a database compact and also periodically if a Synchronizer is managing the replica. The retention period can be changed in the Design Master by using Replication Manager or the **RetentionPeriod** property in JRO. A replica set should have a large retention period if replicas do not synchronize frequently. However, if replicas synchronize frequently and you want to keep replica size smaller, specify a shorter retention period.

For More Information

Here is a list of publications you can refer to if you want more information about database replication, the Microsoft Jet database engine, or using replication in Microsoft products.

- *Microsoft Jet Database Engine Programmer's Guide, Third Edition* (Microsoft Press®, 1999). A detailed analysis of the Microsoft Jet database engine, including extensive information about Microsoft Jet replication.
- "Implementing Database Replication with JRO"

Glossary

ActiveX Data Objects (ADO)
: The programming language-independent object-based data access language designed to be a universal data access approach.

Bidirectional exchange
: An exchange between two replicas, in which both replicas send and receive updates.

Cascade update/cascade delete
: Referential integrity options that specify that changes or deletions to the primary key in one table will be propagated to any other tables that reference that primary key value.

Column-level tracking
: The tracking of data updates at the column (field) level, so that during synchronization updates to different columns

in the same row (record) can be merged.

Column lineage
: A record of each nickname/generation pair for each column. Used when changes to a table are tracked at the column level in order to optimize exchanges between replicas and resolve conflicts.

Conflict
: In some situations, an update received from one replica causes a conflict in the receiving replica if both replicas simultaneously update the same column or row in a table, or if new data violates a unique key rule, or a table-level validation rule. One replica "wins" the conflict and its value is propagated to the rest of the replica set. The "losing row" is logged in a conflict table, which is replicated to all replicas, and every replica is offered the chance to resubmit its update.

Data Access Objects (DAO)
: The programming language-independent object-based data access language used to manipulate Microsoft Jet data.

Database engine
: A program, or part of a program, that serves as the link between a database-management system (DBMS) or application and the data. Specifically, the part of a DBMS program that reads and writes data records.

Design Master
: The single member of a replica set in which design (schema) changes may be made for propagation around the whole replica set.

Exchange
: The process of sending design and data updates between replicas.

Foreign key
: A reference to the primary key in another table.

Generation
: A counter, specific to each replica, that is incremented each time an exchange occurs with any other replica.

Global object
: An object that is replicated around the replica set. Global objects can be created only in the Design Master.

GUID
: Globally unique identifier. A primary key in a database that is always guaranteed to be unique. Also known as a UUID (universal unique identifier).

Jet and Replication Objects (JRO)
: A component of ADO that exposes functions exclusively for the Microsoft Jet database engine. These functions support creating and managing replica sets.

Lineage
: A record of each nickname/generation pair. Used to optimize exchanges between replicas and resolve conflicts.

Local object
: An object that is not replicated around the replica set. A local table and/or query can be created in any replica. Local Microsoft Access objects can be created only if the **ReplicateProject** database property is set to **False** prior to making the database replicable.

Method
: An action that can be applied to an object. For example, to move to the first record of a recordset, you apply the **MoveFirst** method to the **Recordset** object.

Multimaster
: The ability to modify any data at any replica.

Nickname
: A shortened name for of the replica ID GUID, used in the lineage.

OLE
: A standard method of linking and embedding objects created by one program to another program. Also, a type of field (the OLE Object field) in Microsoft Jet used to store complex objects created by other programs.

Objects
: A package of things created and manipulated by programs. In Microsoft Jet, tables, users, and query definitions are all examples of objects.

Open database connectivity (ODBC)
: A connectivity standard to read and write records from other databases, usually server databases.

Primary key
: A field (or multiple fields) in a table that ensures that a record can be uniquely identified.

Property
: An attribute of an object that can be retrieved and (sometimes) set. For example, the **Index** property of a table can be set to the name of one or more fields in the table.

Pull exchange
: An exchange between two replicas where one replica only receives, or pulls, update from the other replica.

Push exchange
: An exchange between two replicas where one replica only sends, or pushes, its update to the other replica.

Read-only replica
: A type of replica that is not permitted to update either data or the design. Read-only replicas can be created only through Replication Manager, JRO, or DAO.

Referential integrity
: A relational database rule that requires that all foreign-key values reference valid primary-key values.

Replica
: A special copy of a database that is created in such a way as to allow changes made in the replica to be exchanged at

a later time with other replicas in the replica set, eventually bringing all the replicas in the replica set into a consistent state.

Replication
: The process of creating special copies of a database, where the copies have a special relationship to each other.

Replica ID
: A GUID that uniquely identifies a replica. The replica ID value is stored in the **ReplicaId** property.

Replica set
: Replicas that share a common heritage and are able to synchronize their data and schema. Replicas can synchronize only with other replicas in the same set.

Retention period
: The amount of time, measured in days, that a replica set retains details of deleted records, design changes, and other system-specific information. This value can be modified only within the Design Master, since it is considered a design change, by using Microsoft Replication Manager or the **Retention** property value in JRO.

Row-level tracking
: The tracking of data updates at the row (record) level; therefore, during synchronization, updates to different columns in the same row will cause a conflict.

Synchronization
: The process of bringing two replicas into a consistent state.

Transaction
: A sequence of actions that must occur as a single unit.

Transaction processing
: A mode of database processing that supports the creation, and saving (**CommitTrans**) or undoing (**Rollback**) of transactions.

Two-phase commit protocol
: A system used in some distributed database systems whereby each database either agrees to or rejects a proposed change, and only if every database in the system agrees is the change actually made.

Validation rule
: An expression that can be linked to a change of data so that it is always evaluated when a certain type of data modification is made.
