CHAPTER 1

Introduction to Advantage Database Server

IN THIS CHAPTER:

Overview of Advantage Database Server

Advantage Database Server Versus Advantage Local Server

Advantage Tools

Building Database Applications Using ADS

Features Added in ADS Version 7
This chapter is designed to provide you with an overall picture of the Advantage Database Server (ADS), including what makes it special, a quick tour of the tools that you will likely use with it, as well as how to build database applications using ADS. If you are new to ADS, you will want to read this chapter carefully. Doing so will show you how you can use ADS in your database applications.

If you are already familiar with ADS, you probably already know much of what is discussed in this chapter. In that case, you might want to quickly skim this chapter before continuing on to Chapter 2. In particular, you might want to read the section “Features Added in ADS Version 7,” the latest release of ADS at the time of this writing.

Overview of Advantage Database Server

Advantage Database Server (ADS) is a relational database management system (RDBMS) marketed by Extended Systems, Inc., a company based in Boise, Idaho. ADS has been around since 1993, when it was introduced to provide a stable solution for Clipper developers who were tired of slow performance and corrupt indexes inherent to file server–based databases. Over the years, ADS has grown in both popularity and features. With the release of Advantage Database Server version 7.0, ADS finds itself a mature product with an impressive collection of features that rivals many of the more expensive and complicated database servers.

But what exactly is ADS? In a nutshell, the Advantage Database Server is a high-performance, low-maintenance, remote database server that permits you to easily build and deploy client/server applications. A lot of information is in that description, but what does it all mean? The following sections consider each of the points in this description.

ADS Is High Performance

First of all, ADS is a high-performance server. It permits you to manage very large quantities of data, and to access that data in a multiuser environment with unbelievable speed. For example, so long as you have designed your tables and indexes correctly, you can usually locate a particular record or subset of records in your database in a fraction of a second.

ADS’s performance derives from its underlying architecture. Unlike many of the more complicated and expensive database servers, such as Microsoft’s SQL Server and Oracle, ADS is not a traditional set-based relational database server based on SQL (structured query language). Instead, ADS is an ISAM (indexed sequential
access method) relational database server. ISAM databases use indexes extensively, permitting them to perform high-speed table searches, filtering, and table joins.

Even though it is an ISAM server, ADS provides extensive support for the SQL language. Indeed, with ADS you can use the industry standard SQL language to perform almost any task related to the management of your data. When it comes to data access, these SQL statements are translated by ADS into optimized, index-based operations, providing you with an unbeatable combination of speed and accessibility.

The ISAM architecture has a long history. It is the same architecture that is used by venerable databases such as dBase, FoxPro, and Clipper. However, those databases were file server databases, while ADS is a client/server database server. In other words, ADS provides the unbeatable combination of proven performance and client/server reliability.

Unlike traditional ISAM databases, however, ADS supports many of the features that you find in high-end, set-based SQL database servers. For example, ADS supports views, stored procedures, triggers, referential integrity, and domain integrity constraints.

**NOTE**

Extended Systems also has an Advantage Replication server, a separate product from ADS, for companies who need to keep multiple databases synchronized. Information about this replication server can be found at www.AdvantageDatabase.com.

Another performance-related ISAM feature distinguishes ADS from set-based SQL databases. ISAM databases support a navigational model of data access, whereas set-based SQL databases do not. In the set-based database model, in theory at least, there is no record order. As a result, the SQL language does not support the concept of navigating a database. While some set-based SQL databases know that record B follows record A, the only way to move to a record that is 100 records after record A is to retrieve the record that follows A, then retrieve the record that follows that one, and again, and again, until this task is performed 100 times. Consider the Delphi language, which supports a navigational model of data access, a legacy of the BDE (Borland Database Engine). Imagine that a Delphi DBGrid (a grid-like component used for displaying a result set) displays data from a `SELECT * FROM CUSTOMER` query against a set-based SQL server where the CUSTOMER table contains a million records. If the end user presses CTRL-END while the DBGrid displaying this result set is active, the DBGrid will navigate to the last record—but in order to do so, it must fetch every single record. Anyone who has seen this knows that it will take a very, very long time before the user arrives at the last record. Furthermore, both the server and the network will be kept busy by this operation.
By comparison, records in an ISAM database have a record order, based on a selected index. If you point a Delphi DBGrid to an ADS table with a million records, and press CTRL-END, you will move immediately to the last record. This is because ADS can use the current index or the table’s natural order to go to the last record, and then return only those last records needed to fill the display of the DBGrid.

This is an important point, especially if you are coming to ADS from a file server database, such as Paradox, dBase, or Access. File server databases permit a navigational approach. If you want to migrate one of these databases to a set-based SQL database server, unless your database is very small, you will likely have to reprogram your user interface to remove any navigational features. Otherwise, users’ attempts to navigate can have serious consequences for your application’s performance.

The problem is that end users love the navigational interface. Having a grid that displays some records from a table, and having the impression that they can easily jump to somewhere in the middle of that table (or anywhere in the table they want to) is very appealing to end users. With ADS, you can provide that feature, but with set-based SQL servers, you should not.

Here is another way to look at it. With ADS you have a choice. You can write your applications using the portable and more or less standardized SQL language, or you can use a navigational model, or you can use both. With SQL-based remote database servers, you are limited to the set-oriented SQL language.

A number of the Advantage data access mechanisms permit you to build client applications that use the navigation model. These include the Advantage TDataSet Descendant, which can be used by Borland’s Delphi, Kylix, and C++Builder, as well as languages that can make direct calls to the Advantage Client Engine, which include Borland’s products as well as Microsoft’s Visual Basic and Visual C++.

NOTE

Development environments that can use ADO (Active X data objects) can leverage most of the navigational model through the Advantage OLE DB Provider. What is missing with ADO is that you cannot set ranges.

ADS Is Low Maintenance

Most database servers require a database administrator to keep them running smoothly and efficiently. And database administrators often require advanced training and certification. But with ADS, most applications require little or no maintenance. In most cases, once the ADS server is installed, you can pretty much forget about it, other than ensuring that your data gets backed up regularly.

“How can this be?” you ask. Once again, this is largely due to the underlying architecture of ADS. For example, so long as you use the Advantage proprietary
format for your data tables, space previously occupied by deleted records is automatically recovered when new records are added. Similarly, ADS’s legendary stability makes it unnecessary to rebuild indexes, in most cases. That ADS-based databases require little or no maintenance makes this server particularly attractive for applications that are distributed to many different locations. For example, if you license your application to many different clients, the less you or they have to worry about managing the database server, the better. This is why ADS is a good choice for vertical market applications and for applications that are deployed to many different systems.

**ADS Is a Remote Database Server**

A remote database server like ADS is an application specifically designed to provide other applications with access to one or more databases. Those other applications are referred to as clients, and most client applications are end user applications. This configuration, where clients request data from a remote server, is referred to as client/server architecture.

The client/server architecture offers many advantages, but the most important is reliability. With ADS, you have confidence that your data is secure and accessible.

**Advantage Database Server Versus Advantage Local Server**

In addition to ADS, Extended Systems also provides a database engine called the Advantage Local Server (ALS). ALS is very similar to Microsoft’s Jet Engine, which is used by MS Access, and Borland’s BDE (Borland Database Engine) when the BDE is using local tables. All of these database engines are file server database engines. Unlike a database server, which is a stand-alone application, a database engine is essentially a library of data access routines that runs in the same process as the client application. Each client application will load its own copy of these database engine files, and each client application is responsible for all data manipulation.

Extended Systems makes ALS available to developers free of charge. Developers can use ALS with their stand-alone and small, multiuser applications, and can even distribute these applications to their clients without paying any royalty fees. The only applications that you are prohibited from using ALS with are those applications that run on the Internet, such as CGI (common gateway interface) Web server extensions. For those types of applications, you must use ADS.
Why, you are probably wondering, should you use ADS if you can use ALS for free? The answer is straightforward. ADS is better.

Actually, the benefits of a database server like ADS over database engines like ALS are associated with four factors. These are reduced network traffic, improved performance, enabled transactions, and unparalleled stability. Each of these factors is examined in the following sections.

Reducing Network Traffic

In short, the Advantage Local Server is not a database server, in the true client/server sense. With file server–based systems, all data processing is performed on the individual workstations. For example, to select all records from a customer table for customers in a particular city, the entire customer table index must be transferred across the network to the workstation, which then finds the record based on the index locally. The located records are then retrieved from the file server.

The problem is worse if an appropriate index does not exist for the table. In that case, the entire table must be transmitted across the network.

While this overhead is negligible if the customer table includes several dozen records, the strain that it places on the network increases with the size of the table. For example, selecting the hundred or so records for customers from Des Moines, Iowa, from a million record table requires that the entire index for all one million records be transmitted across the network.

The problem is particularly bad when you consider that the network load increases in direct proportion to the number of people who are using the application simultaneously. Every single client needs to read indexes and tables across the network in order to get its work done.

By comparison, Advantage Database Server performs its processing on the database server, which is typically the machine on which the shared data files are located. There is no need to copy entire indexes or tables across the network when the client application is interested in only a few records. Specifically, in a client/server environment, the client requests the records that it needs, and the server uses the indexes and tables to locate the needed data. Once the data is identified, that data, and only that data, is transmitted across the network to the client.

Improved Performance

Performance is another area where file server–based systems suffer. Specifically, the individual workstations are responsible for all processing. In other words, all user interface interaction as well as database manipulation is performed on each client workstation in a file server–based system. The file server itself performs no processing, other than transferring files from the server to the workstation.
By comparison, client/server systems perform nearly all data manipulation on the database server, efficiently distributing the processing across multiple machines. When using ADS, your workstations are primarily responsible for the user interface, while the remote server takes care of data manipulation.

Most database applications are multiuser, and it is in these situations where the performance advantages of the client/server architecture are most profound. However, note that in stand-alone applications, where the data is used by only one workstation, ALS-based applications might actually outperform client/server applications. In short, if the data is used by only one workstation, and the data is stored on that workstation, ALS often provides the greatest level of performance. For these applications, ADS provides you with a seamless, high-performance migration path when these applications are converted to multiuser use.

Available Transactions

A transaction is an operation that treats two or more changes to a database as a discrete unit, saving those changes in an all-or-none fashion. If all of the operations involved in the transaction can be completed, the transaction itself is committed. However, if even one operation cannot be successfully completed, all operations within the transaction are canceled, ensuring that your data remains consistent.

Transactions require centralized control of data access, and file server–based systems cannot provide that. Every change in a file server–based system is independent of every other change. As a result, it is entirely possible for a file server–based system to complete some, but not all, operations.

One very important feature of a transaction comes into play if the database server, the client workstation, or the network across which they communicate, fails before the transaction is committed. In those cases, the edits of the transaction are rolled back automatically by the Advantage Database Server once the problem is detected, or the server is restarted (if the server crashed).

Though ALS does not support transactions, you can write your applications that use ALS as if it did. Specifically, when using ALS, your code can include calls to begin, commit, and roll back transactions. These statements will be ignored. However, if you then migrate that application to use ADS, no changes are necessary in your code, and the transactions will be observed.

Improved Stability

The final drawback to file server–based systems is stability. Because there is no centralized control over locking and transactions, the failure of a single workstation can corrupt parts of the database. Developers familiar with BDE-based local table applications occasionally encounter errors such as corrupt or out-of-date indexes.
The issue of stability disappears almost completely when the Advantage Database Server is involved. In these applications, ADS itself is managing the data, and failure of an individual workstation or the network simply cannot harm the data.

When to Use the Advantage Local Server

If ADS is so much better than a database engine solution like ALS, why does Extended Systems bother making ALS available? Well, there are several, but they are all related to a simple fact. Not all applications require the benefits of ADS.

Imagine that you have written a database application that you plan on licensing to many different clients. Your software is probably pretty expensive, and pricing may be an issue for some of your customers. For example, consider a customer who will have only one or two users on the system. The additional cost of an ADS license may make the difference between selling the software or not. For those customers, you can deploy the application using ALS.

Imagine this same customer at some later time. Maybe they now have four simultaneous users, and they cannot tolerate the occasional corruption of an index due to a workstation crash. For a relatively small amount of money, you can upgrade their application to use the Advantage Database Server, and your customer can say “goodbye” to corrupt indexes.

For other customers, especially those who have a large number of users, there is no question about it—you will deploy their applications using ADS.

What is so great about this scenario is that there is no difference, from a development standpoint, between your application using ALS or ADS. The API (application programming interface) for ALS and ADS is exactly the same. And most of the time, upgrading an ALS application to use ADS is simply a matter of installing the server (a process that can be automated).

To put this another way, whether an application uses ALS or ADS is a deployment issue, not a development issue. You will write your application the same way regardless of which deployment option you choose.

Actually, there are a couple of differences, but they really serve more to underscore how similar the two interfaces are. As you have already learned, you can write your application to employ transactions, but only when you deploy with ADS will those statements actually do something.

There is another difference. If you are using the Java language and want to use the Advantage JDBC (Java database connectivity) Driver, you can only use ADS. This is because the Advantage JDBC Driver is a class 4 driver, which is to say that it communicates directly with the server, without going through a client API. If you want to use Java with ALS, you must use the JDBC-ODBC bridge (a class 1 driver), in conjunction with the Advantage ODBC (open database connectivity) Driver.
The second difference is related to triggers and extended stored procedures. With ADS, both triggers and stored procedures execute on the server. Since ALS is not a remote database server, any triggers or stored procedures will actually execute on the client workstation, so you lose the major benefits of distributed computing and reduced network traffic.

The bottom line is that ALS is a wonderful option for database developers. It provides you with a low-cost deployment option that effortlessly scales to client/server when client/server features are required.

Throughout this book we will assume that you are going to deploy your applications using ADS. But we will try to point out when a feature we are describing works differently between ADS and ALS.

**Advantage Tools**

There is more to Advantage than just ADS and ALS. You will also use several support tools to configure your database server, as well as to create your actual databases. Among these are the Advantage Configuration Utility, the Advantage Data Architect, and the Advantage ANSI Collation Utility. Each of these tools is described in the following sections.

**The Advantage Configuration Utility**

When ADS is installed on your server on Windows NT/2000/2003, the Advantage Configuration Utility is also installed. This utility provides you with several important capabilities. It permits you to view statistics about your server’s operation, and it also permits you to manually set many of the server’s configurable properties. Because the Windows NT/2000/2003 server is the most popular server version, the Advantage Configuration Utility is described in this section.

Windows 98/ME, Linux, and Novell NetWare installations of ADS do not include the Advantage Configuration Utility. Windows 98/ME installations are configured from the server application itself, and Linux and NetWare installations are configured from command-line parameters and/or configuration files. For these installations, refer to the documentation for information on configuring ADS.

**ON THE CD**

The ADS documentation is located on this book’s CD-ROM. The ADS documentation can also be downloaded in PDF format from http://devzone.AdvantageDatabase.com.
All OS (operating system) versions of ADS have an Advantage Database Server Management Utility available from within the Advantage Data Architect. (A stand-alone version of this utility is also available with ADS 7.0 and later.) The Management Utility contains information similar to the Advantage Configuration Utility described in this section. See the ADS documentation for additional information on the Advantage Database Server Management Utility.

**NOTE**

The purpose of the following descriptions is to show you where you can get information about the server. This section is not designed to provide a detailed description of how these settings are used. For that information, please refer to the ADS documentation.

To use the Advantage Configuration Utility in Windows NT/2000/2003, select the Start button, and then select Programs | Advantage Database Server | ADS Configuration Utility. The Advantage Configuration Utility shown in Figure 1-1 is displayed. There are three main tabs: Database Info, Installation Info, and Configuration Utility. These pages are described in the following sections.

**Database Info Page**

The Database Info page, shown in Figure 1-1, contains basic statistics about attached users, connections, work areas, as well as open tables and indexes. The Current column
on the Database Info page displays current usage statistics, while the Max Used column shows the highest value for each statistic since the server was started.

This information can help you determine whether you need to make adjustments to any of the configurable parameters of the server. For example, if you find that the maximum number of configured connections were used, and that some connections were rejected, you can use the Configuration Utility pages to increase the number of available connections. The Configuration Utility pages are described later in this section.

**The Installation Info Page**

The Installation Info page, shown in Figure 1-2, contains information about the installed server, including the licensed number of users, your serial number, and the server version you are running.

The values on the Installation Info page are not configurable. If you need to change these values, you can either reinstall your server, or use a special utility that ships with ADS. This utility, called adsstamp.exe, permits you to install a new license key and to change your character sets and a few other settings. For information on using adsstamp.exe, refer to the ADS help.

---

**ON THE CD**

Many of the settings that you can control from the Advantage Configuration Utility can also be adjusted using command-line parameters and/or configuration files. For information about these options, see the Advantage Database Server documentation on this book’s CD-ROM.
The remaining pages of the Advantage Configuration Utility are used to set the parameters used by the server. You display these additional pages by clicking on the Configuration Utility tab of the Advantage Configuration Utility.

In most cases, once you have adjusted the settings of the Advantage Database Server, you will not need to make further changes. However, after the server has been running for a while, you should inspect the Database Info page of the Advantage Configuration Utility to ensure that your initial settings are sufficient. If you find that you have to make adjustments to one or more of the settings found on the Configuration pages of the Advantage Configuration Utility, you will need to stop and then restart your server before your changes take effect.

**The Database Settings Page**

Use the Database Settings page, shown in Figure 1-3, to adjust the maximum permitted connections, work areas, and simultaneously opened tables and indexes. Note that the maximum number of connections is not the same as the maximum number of users. Each machine that is currently accessing the server counts as a user. Each user can have multiple connections. You should set Number of Connections to the maximum user count times the average number of connections each user requires.

**The File Locations Page**

Use the File Locations page, shown in Figure 1-4, to change the location of the error and transaction log files, as well as the semaphore connection file path.
Chapter 1: Introduction to Advantage Database Server

15

The error file, ads.err.dbf, is updated when ADS encounters a problem at runtime. This log file is a simple DBF file that you can open as a free table using the Advantage Data Architect, or using any other utility capable of viewing DBF files.

Transaction log files are files that the ADS TPS (transaction processing system) creates while updates to a database are being performed within the context of a transaction. These important files permit ADS to either commit or roll back changes if there is a failure somewhere in the system before the transaction is complete. These files, which have the .tps extension, are deleted by ADS when the associated transaction is complete.

Unlike the ads.err.dbf files, TPS files are intended for the internal use of ADS. You should not manually delete or attempt to view these files.

The Communications Page

If you want to permit applications to connect to your server over the Internet, use the Communications page to define how this communication will be permitted. The Communications page of the Advantage Configuration Utility is shown in Figure 1-5.

IP Port permits you to set the UDP and TCP ports on which the server will listen for client connections. (“IP,” “UDP,” and “TCP” stand for Internet protocol, universal datagram protocol, and transmission control protocol, respectively.) If IP Port is set to the default 0, it will try to use UDP and TCP port 6262. If you want to specify the UDP/TCP IP Port, set IP Port to a port that is available for both UDP and TCP.
If you want to connect to ADS over the Internet, you set Internet Port to the UDP port that applications (Windows and Linux clients) will use to connect to your server, or to the TCP/IP (transmission control protocol/Internet protocol) port that Java applications will use to connect to your server. This port (TCP/IP for Java clients or UDP/IP for Windows and Linux clients) must also be opened in any firewalls separating the server and the applications that need to access it. Leaving Internet Port set to the default of 0 disables Internet access.

NOTE
There are plans to include full TCP/IP support with Windows and Linux clients in an ADS release sometime after 7.0.

You also use this page to configure whether to use a semaphore file, the client connection timeout, and level of data compression.

NOTE
To enable Internet access to the Advantage Database Server, you must also configure the data dictionary used to access the data to accept Internet connections. Depending on the data access mechanism or the specifics of your connection, each client machine may also require an ADS configuration file that contains additional information about the connection. For information on setting up client applications to connect over the Internet, see the topic “Advantage Internet Server” in the ADS documentation.
The Misc. Settings Page

Use the final page of the Advantage Configuration Utility, Misc. Settings, shown in Figure 1-6, to set the maximum number of worker threads, as well as the maximum size of the error log.

Advantage Data Architect

The Advantage Data Architect, whose main screen is shown in Figure 1-7, is a graphical tool that simplifies the creation and configuration of your tables and databases. The Advantage Data Architect is also referred to as ARC. The Advantage Data Architect also includes a wide variety of support features, including the ability to import and export data, execute ad hoc queries, perform checks on your development environment, and much, much more. As a developer, you are likely to use the Advantage Data Architect more than any other tool that ships with ADS.

While the Advantage Data Architect makes it easy to design your tables and data dictionaries, there are alternatives. Specifically, all of the database-related configuration capabilities provided by Advantage Data Architect can also be performed using Advantage SQL, as well as by programming directly to the ADS API using ACE (the Advantage Client Engine).

The Advantage Data Architect provides another benefit, albeit one that is not immediately obvious. It can provide you with insight on how to create and manage...
your tables and data dictionaries programmatically. The Advantage Data Architect was written in the Delphi language, and the source code for this Delphi project is available on the CD-ROM that accompanies this book. The Delphi language is easy to read, so even if you are not a Delphi developer, you can probably figure out how specific operations were performed.

Because the Advantage Data Architect is such a valuable tool for working with tables and data dictionaries, a large portion of this book discusses its use. In other words, there is no chapter that specifically discusses the Advantage Data Architect. Instead, almost all examples in this book where tables and data dictionaries are manipulated use the Advantage Data Architect.

**Advantage ANSI Collation Utility**

ADS ships with the Advantage ANSI Collation Utility, shown in Figure 1-8, a tool that permits you to create custom ANSI collations. The ANSI (American National Standards Institute) character set is a standard mapping of characters to numeric values. These characters include printable characters and special control characters, such as tabs and carriage returns. An ANSI collation sequence defines the order, or precedence, of characters for the purpose of making string comparisons.
Few developers will ever need to use the Advantage ANSI Collation Utility. For most applications that use the English language, you will install one of the provided English collation sequences. Similarly, for most non-English applications, Extended Systems provides localized character sets that ensure proper string comparisons.

For those non–English speaking developers for whom there is no localized character set, the Advantage ANSI Collation Utility is an essential tool for defining character precedence.

When installing ADS or ALS, you will choose either a localized character set or an ANSI collation sequence. Which options you have depends on the version of the Advantage Database Server that you are installing. For example, if you are installing the domestic version of ADS, you will be able to select only between an American English collation, a Canadian English collation, and a French Canadian collation.

Regardless of which character set or collation sequence you install, keep one very important issue in mind. The server, and all client applications that use it, must use the same character set or collation sequence. Doing so ensures that both clients and server agree on how strings are compared.
Because both client and server must use the same character set or ANSI collation sequence, it is particularly important for non-English applications to use the ANSI character set provided by Extended Systems, instead of choosing “default on machine.” This is especially important for Advantage Local Server users. Each version of Windows (95, 98, NT, 2000, and so on) has different “default on machine” ANSI collation sequences, even when the same language is configured. Consequently, if one ALS client is using Windows 98 and another is using Windows 2000, collation mismatch errors will result.

**CAUTION**

If you change the character set or collation sequence being used by your clients and server, you must rebuild all indexes before you access any of your tables.

---

**Building Database Applications Using ADS**

The Advantage Database Server is a RDBMS (relational database management system), but it is not a full-scale database development environment. While ADS includes a varied set of tools for creating and configuring the files that will be used by a database application, you do not use it directly to build the actual client applications. For that purpose you use one of a wide range of development environments.

This section is designed to provide you with a broad understanding of the role that ADS, and the tools that come with it, play in application development. To meet this end, this discussion is broken down into two parts: creating the database and creating the client applications.

In reality, this discussion could have been organized in a number of different, yet equally correct, ways. For example, database development is often described as a four-part process of design, development, testing, and deployment. While true, that description focuses on the process, and not the tools. Since this book is specifically about the Advantage Database Server, we felt that a tool-based view is more appropriate.

**Creating the Database**

A *database*, as the term is used here, is a collection of files used by the Advantage Database Server. With ADS, these files include tables, memo files, indexes, and in most cases, data dictionaries.

**NOTE**

If you are unfamiliar with what the parts of a database are, you might want to take a quick look at Chapter 2 (creating tables), Chapter 3 (defining indexes), and Chapter 4 (using data dictionaries). The introductions to each of these chapters define what these various files are, and what roles they play in a database.
In most cases, you use the Advantage Data Architect to create your database. Specifically, you either import existing data into one or more tables, or you design the tables from scratch. You then consider how your data will be accessed by the client applications, and design the indexes that will make that access fast.

In many cases, you also design a data dictionary. A data dictionary, as far as ADS is concerned, is a special file that is used to access all of the tables within a given application. Data dictionaries provide your client applications with a number of powerful features, such as security, constraints, and referential integrity, to name just a few. Here again, you will probably use the Advantage Data Architect to create your data dictionary.

Instead of using the Advantage Data Architect to create your database, you can also create your database at runtime from one or more of your client applications. To do this, your client applications would need to include code that defines new tables, indexes, and data dictionaries, as needed, on-the-fly. Your client applications would not explicitly create memo files. Memo files are created automatically when data is inserted into memo or BLOB (binary large objects) fields.

Creating databases on-the-fly from within a client application requires a lot of programming, compared with creating databases with the Advantage Data Architect. Fortunately, creating a database at runtime is normally only necessary in special situations. For example, some applications need to store separate data in separate databases. If these applications need to be able to create a new database at any time, then you will probably need to take the extra effort to create the new database at runtime.

Before a client application can access the tables, indexes, and data dictionaries of an application, the client workstation must be able to send IP packets to the server upon which Advantage Database Server is running. Client applications don’t even need the rights to access the individual files. The server does this. The exception is when the client is using ALS, in which case the client application must have rights to the shared table, index, and memo files.

The server on which you install your database and the Advantage Database Server can be running one of the following operating systems: Windows 98, Windows ME, Windows NT 4.0 (Service Pack 6 or later), Windows 2000, Windows 2003, Novell NetWare 4 or later, or Linux.

Actually, there are four versions of the ADS server. The most popular version is designed for Windows NT 4.0 or later, which includes 2000 and 2003. For Novell networks, there is a NLM (NetWare loadable module), which requires NetWare version 4 or later. The Linux version of ADS requires glibc 2.1.2-11 or newer and kernel version 2.2 or newer. Finally, there is a Windows 98/ME version. However, Windows 98 and ME are poor hosts for a server, so you should consider one of the other versions of ADS.
NOTE

Windows XP is not listed as a supported OS since XP is marketed by Microsoft as a client operating system only, not as a server operating system. ADS for Windows NT/2000/2003 runs on XP, although XP is not a supported OS.

Some final comments about creating a database are in order before continuing on to the discussion about building client applications. First, while creating tables, indexes, and data dictionaries is not hard, the design of the tables, indexes, and data dictionaries is often the result of research and thoughtful consideration.

For example, you need to take into account what kind of data your need to store and how it will be used. This will lead you to consider how many tables you will need to create, and what indexes you will add to them. Likewise, how you set up a data dictionary, including what groups and users to add, what views to define, and whether to use referential integrity, can have a big impact on the success of your database application.

A second consideration is that the design of your database is something that will likely change over time. In short, database design is often an iterative process.

As your design begins to take shape, you often will discover that a particular table is missing one or more fields, or that additional indexes need to be created, or that entirely new tables must be added.

These changes, when they happen, will often affect both the server and the client parts of the application. For instance, if you find that you have to add a new field to a table, you might use the Advantage Data Architect to update the table file. In most cases you will then also need to change your client applications so that they can use the newly added field.

Creating the Client Applications

The Advantage Data Architect is a pretty powerful piece of software. You will learn as you work through the later chapters in this book, that not only does the Advantage Data Architect provide you with the tools to design a database, but also that it permits you to work with the database. Specifically, using the Advantage Data Architect, you can add new records to your tables, change existing records, and delete records. It also provides you with a number of tools to sort the data in your tables, as well as to filter your data to display only a subset of the records in a given table. Indeed, it even permits you to export your data to HTML (hypertext markup language), which you can then print, using this feature as a simple reporting mechanism.

Here is another way to look at it. The Advantage Data Architect is an ADS client application. However, it is a very general ADS client whose primary purpose is to give you the ability to easily design and test your database. In most cases, you will create one or more custom client applications to work with your ADS databases.
Custom client applications are all about making your data accessible to your end users. Sure, an end user could use the Advantage Data Architect to view just a subset of records, but that would require that your end user know SQL. Similarly, while an end user could export the results of a SQL query to HTML, and then print this from a Web browser, the output would lack the quality and sophistication that most people want to see in a report.

With a custom client application, your end users are presented with data options that make sense in the context of the data. For example, your application can display a form from which the end user enters a customer account number. Once the number is entered, your application can perform a query to select the data for that customer, and display the customer data in an easy to view format. Similarly, your application can include a menu item that, when selected, creates and prints an attractive report that makes the data easily digestible.

You can create custom client applications that use ADS from a wide variety of application development platforms. Examples of languages from which you can access ADS include Borland products Delphi, C++Builder, C#Builder, Kylix, and JBuilder; Microsoft products Visual Basic, Visual C++, Visual C#, FoxPro, Visual J#, and Visual Studio.NET; IBM’s VisualAge for Java; Sun Microsystems’ Sun ONE Studio; Computer Associates’ Clipper; Grafx Software’s Visual Objects; and Corel’s Paradox. For Web-based development, many of the preceding products can be used to create CGI or ISAPI (Internet server application programming interface) Web server extensions. In addition, you can access ADS from PHP and Perl.

In short, you can use any application development environment for which Extended Systems supplies a data access mechanism. Fortunately, Extended Systems provides a large number of data access mechanisms. These are listed in Table 1-1, which includes a description of the development environments best suited for each mechanism.

Several of the data access mechanisms listed in Table 1-1 refer to groups of drivers. For example, there are two Advantage ODBC drivers, one for 32-bit Windows and another for Linux. Similarly, there are Windows and Linux versions of both the Perl and PHP drivers.

As you can see from this list, you have an impressive selection of data access options with the Advantage Database Server. As a result, you can access your data from Advantage Database Server with nearly every modern software development environment.

There is another point that deserves mention here. Each of these data access options can access any version of the Advantage Database Server. Specifically, even though the Advantage OLE DB Provider is only available for Windows clients, it can access ADS running under Linux as well as Novell NetWare. Similarly, a Java client using the Advantage JDBC Driver running on a Macintosh can access any ADS server, regardless of the operating system the server is running on.
## Data Access Mechanism

<table>
<thead>
<tr>
<th>Supported Development Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantage TDataSet Descendant</td>
</tr>
<tr>
<td>Advantage ODBC Driver</td>
</tr>
<tr>
<td>Advantage OLE DB Provider</td>
</tr>
<tr>
<td>Advantage Client Engine (ACE) API</td>
</tr>
<tr>
<td>Advantage Clipper RDDs (replaceable database drivers)</td>
</tr>
<tr>
<td>Advantage Visual Objects RDDs</td>
</tr>
<tr>
<td>Advantage JDBC Driver</td>
</tr>
<tr>
<td>Advantage .NET Data Provider</td>
</tr>
<tr>
<td>Advantage DBI Driver</td>
</tr>
<tr>
<td>Advantage PHP Extension</td>
</tr>
<tr>
<td>Advantage Crystal Reports Driver</td>
</tr>
</tbody>
</table>

### Table 1-1  Data Access Mechanisms for the Advantage Database Server

**ON THE CD**

The drivers listed in Table 1-1 can be found on the CD-ROM accompanying this book.
Chapter 1: Introduction to Advantage Database Server

Features Added in ADS Version 7

The Advantage Database Server is a mature product, and version 7.0 is the latest release at the time of this writing. With this release, Extended Systems has continued the tradition of adding powerful new features to the server, while maintaining the ease of use that is its hallmark. These features are discussed in the following sections.

Triggers

A trigger is a compiled routine that executes on the server in response to a data-related event. ADS supports record-level triggers on insert, update, and delete events. For each of these events there are three trigger event types: a before, an instead of, and an after event type. As a result, there are a total of nine trigger events.

Like AEPs (Advantage Extended Procedures), ADS’s stored procedure mechanism, triggers can be implemented in a wide variety of languages. Specifically, Advantage triggers can be written as standard Windows DLLs (dynamic link libraries), COM (component object model) servers, Linux SO (shared object) libraries, or .NET managed assemblies. Unlike AEPs, Advantage triggers can also be implemented using SQL scripts.

Creating and installing triggers is discussed in Chapter 8.

Full Text Search

Full text search (FTS) permits you to conduct high-speed, index-based searches of text, memo, and BLOB fields. Using full text search, you can add sophisticated search-engine functionality to your client applications. Full text search is discussed in detail in Chapter 3.

Data Dictionary Upgrade Functionality

Two versions of a data dictionary can be compared. The result of this comparison is a script of SQL statements that you can use to transform one data dictionary to another.

The purpose of these generated SQL scripts is to permit you to easily upgrade a data dictionary from within a client application. Once an upgraded SQL script has been generated, you can customize it, if necessary, and then include that script in your application. For example, within your application you can determine the current version of a data dictionary, and execute the appropriate upgrade if you find it to be an older version. This capability is particularly beneficial to developers who deploy their applications to many different sites and who need to provide a mechanism to upgrade older data dictionaries.

Comparing data dictionaries is discussed in Chapter 4.
The Advantage .NET Data Provider

While .NET developers can use the Advantage OLE DB Provider with the classes of the System.Data.OleDb third-level namespace to connect to ADS, Extended Systems has created a custom .NET Data Provider for optimized access to ADS using ADO.NET. These classes can be found in the AdsProvider namespace and include AdsConnection, AdsCommand, AdsDataReader, AdsParameter, AdsParameterCollection, AdsCommandBuilder, and AdsDataAdapter.

The Advantage .NET Data Provider is discussed in Chapter 15.

The Advantage JDBC Driver

Prior to the release of ADS 7.0, Java developers had to rely on the JDBC class 1 JDBC-ODBC bridge in order to use ADS from their Java applications. Although this solution works, it is limited to clients running on ODBC-enabled operating systems. It also creates a more complicated client install, since not only does the Advantage ODBC driver need to be installed, but so does the ACE (Advantage Client Engine). In ADS 7.0 and later, Java developers have a class 4 JDBC driver for ADS, the Advantage JDBC Driver. A class 4 JDBC driver is one that is written entirely in Java, and that does not require a client API. As a result, Java client applications using a class 4 driver are very convenient to install.

The Advantage JDBC Driver is discussed in Chapter 13.

Extended SQL Support

With each release of ADS, more of ADS’s functionality has been exposed through Advantage SQL. With ADS 7.0, support has been extended to almost every aspect of the product. Now you can obtain metadata about databases and data dictionaries, create and remove users, set user access rights, and more, all using the SQL language. Now there is almost nothing you can do with ADS that cannot be done using SQL.

To learn more about the syntax and semantics of SQL with Advantage, please refer to Chapters 9, 10, and 11.

Other Enhancements

While not as spectacular as triggers or full text search, a number of additional enhancements to ADS are sure to be welcomed by pre-7.0 ADS users. The first of these is a new compression algorithm that will improve the speed of data transfer over networks. While this feature will improve the already fast performance of ADS over local area networks (LANs), it will be most welcome by developers needing to access ADS from client applications across the Internet and in other WAN (wide area network) environments.
Another welcome addition is a MONEY field type in ADS tables. The money field is similar to the CURDOUBLE field. However, unlike CURDOUBLE fields, which are internally stored as IEEE (Institute of Electrical and Electronics Engineers, a standards body) double values, all calculations with those fields are done using doubles. IEEE doubles can easily lose precision. With the MONEY data type, all storage and calculations are performed using 64-bit integer fields, which lose no precision.

Another notable upgrade applies only to the NetWare version of ADS. With ADS 7.0 and later, NetWare ADS servers will no longer be limited to files smaller than 4GB. The maximum file size is now 16 exabytes for tables, 35 terabytes for index files, and 4 terabytes for memo files. You will also need to be using NetWare 5 or later for this feature.

**NOTE**

An exabyte is a billion gigabytes.

In the next chapter, you will learn how to use ADS to create tables.