Advantage Database Server: The Official Guide

by Cary Jensen and Loy Anderson ISBN:0072230843
McGraw-Hill © 2003 (468 pages)

This resource delivers to-the-point information on using ADS to build outstanding databases with unparalleled stability, scalability, and performance.

Table of Contents

Back Cover

Finally! The official guide to Advantage Database Server has arrived to help you easily develop and deploy client/server applications. Manage large quantities of data and access that data in a multiuser environment with unbelievable speed. Learn how ADS gives you unparalleled reliability and data integrity with little or no database administration. Take advantage of features such as referential integrity, transaction processing, triggers, full text search, and much more. Technically reviewed by the Extended Systems Advantage Database Server R&D team, this one-of-a-kind resource will help you build and deploy reliable, scalable, relational database applications.

- Easily create robust, high-performance, client/server applications
- Leverage ADS's support for both set-based SQL and index-based navigation
- Enjoy nearly maintenance-free administration using self-tuning features
- Get the speed and reliability of transaction-based remote database server without the complexity
- Support Windows NT/2000/2003, Linux, NetWare, and Windows98/ME operating systems
- Replace the deprecated Borland Database Engine with practically no modifications
- Use virtually any development environment, including C# .NET, C#Builder, Visual C++, C++Builder, Delphi, Delphi for .NET, Java, JBuilder, Kylix, PHP, Perl, Visual Basic, and Visual Basic .NET

About the Authors

Cary Jensen, Ph.D., is President of Jensen Data Systems, Inc., a Houston-based company that provides training, consulting, and support. He is the best-selling co-author of nineteen books, and is a Web and magazine columnist. He is one of the world's leading authorities on database development, and presents award-winning training seminars in North America and Europe.

Loy Anderson, Ph.D., is Vice President of Jensen Data Systems, Inc., and best-selling, award-winning co-author of nineteen books. She has served as coordinating editor on three books, and manages development seminars.

Advantage Database Server—The Official Guide

Cary Jensen
Loy Anderson

McGraw-Hill/Osborne
2100 Powell Street, 10th Floor
Emeryville, California 94608
U.S.A.

1234567890 FGR FGR 019876543

Book p/n 0-07-223085-1 and CD p/n 0-07-223086-Xparts of ISBN 0-07-223084-3

Publisher: Brandon A. Nordin
Vice President & Associate Publisher: Scott Rogers

Editorial Director: Wendy Rinaldi
Project Editor: Jennifer Malnick
Acquisitions Coordinator: Athena Honore
Coordinating Editor: Loy Anderson
Technical Editor: Brad Schmidt
Contributing Technical Editors: Chris Franz, Patrick Harper, Jeremy “J.D.” Mullin, Lance Schmidt, Chuck Vertrees, Mark Wilkins, and Alex Wong
Copy Editors: Jan Jue, Dennis Weaver
Proofreader: Linda Medoff
Indexer: Loy Anderson, Cary Jensen
Composition: Carie Abrew, Tara Davis
Illustrators: Kathleen Edwards, Michael Mueller, Melinda Lytle
Series Designer: Roberta Steele
Cover Illustration: Pattie Lee

This book was composed with Corel VENTURA™ Publisher.

We dedicate this book to our dance instructors, Al Lena and Ronny Roch, for sharing with us this beautiful expression of life.

About the Authors

Cary Jensen

Cary Jensen is President of Jensen Data Systems, Inc. In addition to his consulting and development duties, he is an award-winning, best-selling author of 19 books, including *Building Kylix Applications* (McGraw-Hill/Osborne), *Oracle JDeveloper* (Oracle Press), *JBuilder Essentials* (McGraw-Hill/Osborne), *Delphi In Depth* (McGraw-Hill/Osborne), and *Programming Paradox 5 for Windows* (Sybex). Cary is a featured Web columnist for the Borland Developers Network (bdn.borland.com), where his column “The Professional Developer” appears, and is Contributing Editor of *Delphi Informant* Magazine. He is an internationally respected speaker and trainer, and was the author and principal speaker for the 2001–2003 Developer Days Seminars and Workshops, the 2000–2001 Delphi Development Seminars, the 1999–2000 Borland Developer Days, and the 1995–1999 Borland/Softbite Delphi World Tours. Cary has a Ph.D. from Rice University in Human Factors Psychology, specializing in human-computer interaction.

Loy Anderson

Loy Anderson is Vice President of Jensen Data Systems, Inc., a Houston- based consulting and training company. She is an award-winning, best-selling co-author of 19 books with Cary Jensen and has served as coordinating editor on several books. Loy manages the Delphi Developers Days Seminars and Workshops, which won the 2002 and 2003 Delphi Informant’s Readers Choice Awards for Best Training. Her company’s latest workshops include .NET development, Delphi development, and Advantage Database Server training. She was also an event coordinator for the 1999–2000 Borland Developer Days and the 2000–2001 Delphi Development Seminars. Loy has a Ph.D. from Rice University in Human Factors Psychology, specializing in human-computer interaction.

About the Lead Technical Editor

Brad Schmidt

Brad Schmidt is an R&D Project Manager at Extended Systems, Inc. Prior to becoming project manager for the Advantage Database Server product line, as well as other software products at Extended Systems, he was a lead developer on the Advantage Database Server. He has worked on the Advantage product line since its inception in 1992, and has been the lead architect and developer on many of its significant features, including the Transaction Processing System, the port of the server to Windows NT, the Advantage Local Server, the Advantage proprietary file format (ADT/ADI/ADM files), and the Advantage OLE DB Provider. Brad holds a patent on functionality available in the Advantage Transaction Processing System. In addition to his current project management responsibilities, he has significant influence on new product features, contributes to all facets of Advantage documentation, and delivers several developer training sessions a year on Advantage Database Server. Brad lives in Eagle, Idaho, with his wife, Judy, two sons, Kellen and Nicholas, and a daughter, Lindsey. He has a bachelor's degree in Computer Science from Montana State University.

Acknowledgments

We want to take this opportunity to express our heartfelt thanks to everyone who made this book possible. From Osborne Media Group, we want to thank Scott Rogers, who got this project rolling; Wendy Rinaldi, who made sure it happened; Jennifer Malnick, for her professionalism and dedication in moving this book quickly through production; as well as Dennis Weaver, our favorite copyeditor; and Jan Jue, for her meticulous work. We also want to thank Athena Honore.

We are also deeply indebted to the wonderful people at Extended Systems, Inc. In particular, we want to thank Katie Moser, Advantage Marketing Manager, and Chuck Vertrees, Advantage Product Manager, for inspiring this project and promoting it at Extended Systems. We are also deeply indebted to, and grateful for, the technical editing team, who gave us, hands-down, the best technical edit and support that we have ever had on a book. This team was lead by this book's Technical Editor, Brad Schmidt. During the course of this book, not only did Brad, the Advantage R&D Project Manager, review these chapters, but he also had each chapter reviewed by several, and sometimes all, of the following people (listed in alphabetical order): Chris Franz, Patrick Harper, Jeremy "J.D." Mullin, Lance Schmidt, Chuck Vertrees, Mark Wilkins, and Alex Wong. In addition, we appreciate the unflinching help of Lance, J.D., Chris, and Alex for their advice and assistance in writing the applications that appear in Part III of this book. Other people at Extended Systems we want to thank include Charles Jepson, Justin Borg, Joachim Dürr, Dirk Erickson, Peter Funk, Phil Ruebel, Lee Stigile, Bryce Twitchell, Jay Wendt, and last, but not least, master salesman Bill Schuler.

We also want to thank the many wonderful people who use and champion the Advantage Database Server. This is a really great community, and your enthusiasm for this product is nothing less than contagious. Thanks also to our colleague Bill Todd. Finally, we want to thank our friend Corey Wood, who provided us with technical assistance when we wrote our first training course on the Advantage Database Server.

Introduction

I have considered myself a database developer since Loy Anderson and I launched our business in 1988. During that time, I have had the pleasure of working on a large number of different and exciting database projects in almost every field imaginable, from medical to legal, from financial to aerospace, from manufacturing to distribution.

I also started training corporate software developers in 1988, and in the intervening years have trained thousands of database developers. In the early days, most of these developers used a local, file server–based database engine, such as Paradox, dBase, FoxPro, Clipper, or MS Access, though over time an increasingly greater proportion used a client/server architecture. But even to this day there are a surprisingly large number of developers still using a local database engine.

Over the years, as the limitations of the file server–based architecture became more widely understood, a specific interaction repeated itself over and over, and does so to this day. A developer in one of my seminars will approach me during a break and relate that they are still using a local database engine. Realizing that they need greater stability and performance, they will tell me that they are switching to a particular database server, almost always one of the big-name database servers. When I ask them how many users are on their system, they often reply that they have three or five or ten. Rarely more than twenty.

This exchange always leaves me feeling a little shocked. Do they know what they are getting themselves into? Sure, the client/server architecture offers greater stability and increased performance, but are they going to be able to allocate the resources that most of the high-end database servers require? Do they have a DBA (database administrator) already? If not, can they afford to hire a DBA? Can they afford the continuing training and certification? Is the expense worth it when they have only five users?

I'm sharing this background with you so that you will understand why I am such a huge fan of Advantage Database Server (ADS). For almost every developer with whom I have had the preceding interaction, ADS is a better solution for implementing their application using a client/server architecture, one that can easily handle hundreds of users. Not only does ADS offer the reliability and performance of a remote database server, it does so with a very minimum of administration. In a world where we often confuse sophistication with complexity, ADS is the perfect exception to the rule. But isn't this the way it should be? Simple sophistication. Wouldn't it be great if all software were like this?

But there is something else about the Advantage Database Server that originally attracted my attention—the company that makes it. From my very first contacts with the Advantage group at Extended Systems, all the way through the writing of this book, I have been continuously impressed with the level of professionalism and commitment by the people behind the product. From the individuals in marketing and sales, through the management and development team, there is a culture of pride that permeates this group. Ask anyone who uses ADS and you'll likely hear the same thing. This product is great, and so is the company behind it. What a combination.

Don't get me wrong: My admiration for the people behind ADS is not intended as a slight to other software companies. Indeed, I have had in the past, and continue to have, great respect for a number of the software companies that I deal with, the people who work at them, and an appreciation for the culture of excellence that they engender. Borland Software Corporation is an outstanding example of what I am talking about. But the fact remains that ADS is backed by a first-rate company that respects and supports its customers. You can't say that about every software company.

—*Cary Jensen*

About This Book

This book is designed to get you started using the Advantage Database Server, organized into three parts. In Part I, you learn how ADS gives you the performance of a world-class, relational, remote database server without the complexity normally associated with the client/server architecture. Topics discussed here include what ADS is, what your data access options are, how to build tables and indexes, and how to set up a data dictionary in order to create a secure and feature-rich database.

In Part II, you learn the ins and outs of Advantage SQL, the particular dialect of ANSI/92 SQL used by the Advantage Database Server. And in Part III, you learn how to access your ADS data from the most popular development environments, including Delphi, Java, Visual Basic, C#, and more.

The CD-ROM that accompanies this book includes code samples as well as a set of database files. As you progress through the chapters of Part I, you will make modifications to this database, adding tables, indexes, a data dictionary, constraints, stored procedures, triggers, and more. Each chapter contains step-by-step instructions that walk you through the process of creating these various objects. Consequently, in most cases, you will want to start this book from the beginning and work your way through the chapters in sequence.

There is one thing to note about this database that you will be working with in this book: The examples described do not add all of the objects that you would probably want to add to this database if you were creating a production application. For example, the step-by-step instructions only have you add the indexes that are necessary for the other objects you will create later in the book, such as referential integrity definitions.

In other words, if you were actually going to deploy this database, you would likely add many more indexes than we have you add here. But that point, and other similar points related to the creation of objects, is made repeatedly throughout. For a real database, you create those objects that support the features that you want to implement in your client applications. Importantly, the chapters in Part I explain what these features are and show you how to define them.

Who This Book Is For

This book is for anyone who wants to create better database applications without the complexity normally associated with the client/server architecture. If you are not currently using a remote database server, you are cheating yourself if you do not give ADS a look. If you are currently using another database server, and are unhappy with the level of maintenance that it requires, you too will benefit from considering the advantages of ADS.

The CD-ROM that accompanies this book includes everything you need to give ADS a try; it includes a single-user license for the Windows NT/2000/2003, Linux, Novell NetWare, and Windows 98/ME platforms. It also includes all utilities and associated client drivers, permitting you to access this server from almost every development environment imaginable. You will also find a link on the CD-ROM that permits you to download a 5-user trial version of the ADS server, in case you want to test ADS in a multiuser environment.

If you do not need a remote database server just yet, this CD-ROM also includes ALS, the Advantage Local Server. (ALS is installed when you install any of the client drivers, with the exception of the Java client. Read the license.txt file located in the directory in which you install a client driver for further information.) ALS is a file server–based database engine that supports up to five simultaneous users, and can be deployed royalty-free. (Note, however, that the free ALS cannot be used in Internet applications or with the Advantage JDBC Java Driver.) Importantly, ALS and ADS have identical APIs (application programming interfaces), permitting you to effortlessly convert an ALS client to an ADS client. The CD-ROM also includes the sample code listings and applications described in this book.

In short, the CD-ROM has everything you need to get started with ADS today.

So buckle your seatbelt and return your seat to its full, upright position. The fun is about to begin

Part I: ADS and the Advantage Data Architect

Chapter List

Part Overview

Part I of this book shows you how to build and configure the tables, indexes, and data dictionaries that you access from ADS (Advantage Database Server). Chapter 1 begins with a broad overview of ADS, providing you with a detailed understanding of the power, performances, and flexibility of ADS.

Chapters 2 and 3 show you how to create the basic building blocks of a high- performance database. In Chapter 2, you discover how to define tables, the data structures in which your data is stored. And in Chapter 3, you explore indexes, the means to fast and efficient data access.

In Chapter 4, you learn the power of ADS data dictionaries, including the many advanced features that data dictionaries provide. This chapter also shows you how to implement security using a data dictionary, including how to add users and groups to control access to your data.

Chapter 5 provides you with an in-depth look at constraints, record and field-level definitions that improve the accuracy of the data managed by ADS. And in Chapter 6, you discover how to create custom views of your tables. Included in this discussion is how you can modularize your access to data using views, simplifying what would otherwise require sophisticated data manipulations.

The final two chapters in this section provide you with everything you need to know in order to use two of ADSs most advanced features: stored procedures and triggers. Not only do these chapters provide you with a detailed examination of the role that these features play in your applications, but they also show you how to implement these objects using some of today's hottest development languages, including Delphi, Visual C# .NET, and Visual Basic .NET.

Chapter 1: Introduction to Advantage Database Server

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.

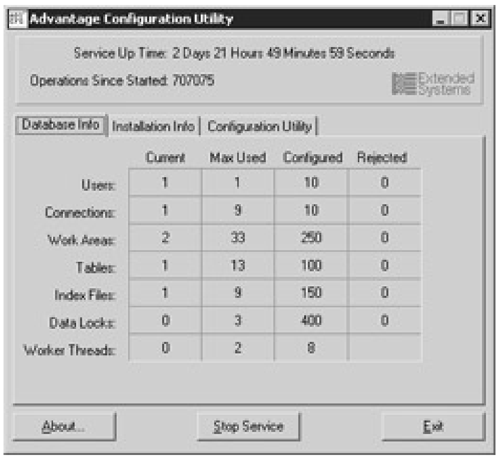
Windows 98/ME, Linux, and Novell NetWare installations of ADS do not include the Advantage Configuration Utility. Windows 98/ME installations are configured from the server application itself, and Linux and NetWare installations are configured from command-line parameters and/or configuration files. 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.



Service Up Time: 2 Days 21 Hours 49 Minutes 59 Seconds

Operations Since Started: 707075

	Current	Max Used	Configured	Rejected
Users:	1	1	10	0
Connections:	1	9	10	0
Work Areas:	2	33	250	0
Tables:	1	13	100	0
Index Files:	1	9	150	0
Data Locks:	0	3	400	0
Worker Threads:	0	2	8	

Figure 1-1: Use the Advantage Configuration Utility to view statistics for and to configure your ADS server.

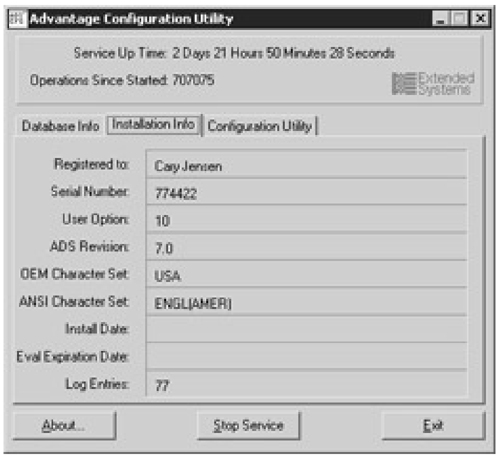
Database Info Page

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.



Service Up Time: 2 Days 21 Hours 50 Minutes 28 Seconds

Operations Since Started: 707075

Registered to:	Cary Jensen
Serial Number:	774422
User Option:	10
ADS Revision:	7.0
OEM Character Set:	USA
ANSI Character Set:	ENGL(AMER)
Install Date:	
Eval Expiration Date:	
Log Entries:	77

Figure 1-2: The Installation Info page of the Advantage Configuration Utility

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.

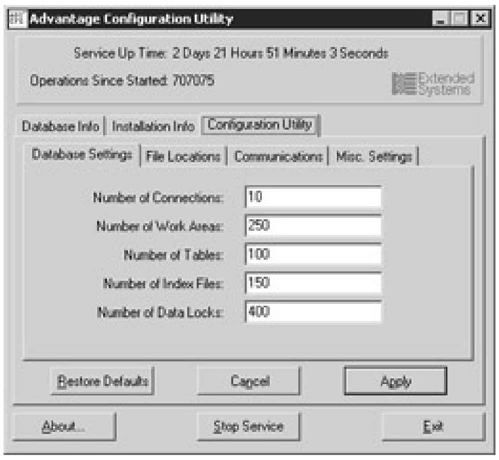
On the CD Many of the settings that you can control from the Advantage Configuration Utility can also be adjusted using command-line parameters and/or configuration files. For information about these options, see the Advantage Database Server documentation on this book&"para">The remaining pages of the 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.


Advantage Configuration Utility

Service Up Time: 2 Days 21 Hours 51 Minutes 3 Seconds
Operations Since Started: 707075

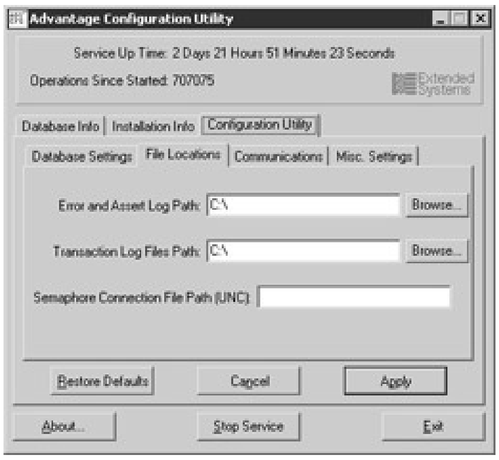
Database Info | Installation Info | Configuration Utility
Database Settings | File Locations | Communications | Misc. Settings

Number of Connections: 10
Number of Work Areas: 250
Number of Tables: 100
Number of Index Files: 150
Number of Data Locks: 400

Figure 1-3: The Database Settings page of the Advantage Configuration Utility

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.


Advantage Configuration Utility

Service Up Time: 2 Days 21 Hours 51 Minutes 23 Seconds
Operations Since Started: 707075

Database Info | Installation Info | Configuration Utility
Database Settings | File Locations | Communications | Misc. Settings

Error and Assert Log Path: C:\
Transaction Log Files Path: C:\
Semaphore Connection File Path (UNC):

Figure 1-4: The File Locations page of the Advantage Configuration Utility

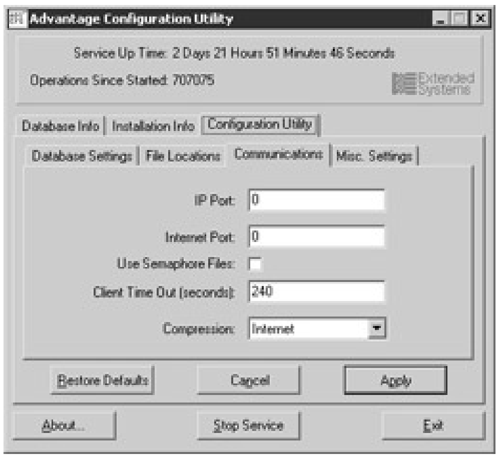
The error file, ads_err.dbf, is updated when ADS encounters a problem at runtime. This log file is a simple DBF file that you can open as a free table using the Advantage Data Architect, or using any other utility capable of viewing DBF files.

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.



Service Up Time: 2 Days 21 Hours 51 Minutes 46 Seconds
Operations Since Started: 707075

IP Port: 0
Internet Port: 0
Use Semaphore Files: ☐
Client Time Out (seconds): 240
Compression: Internet

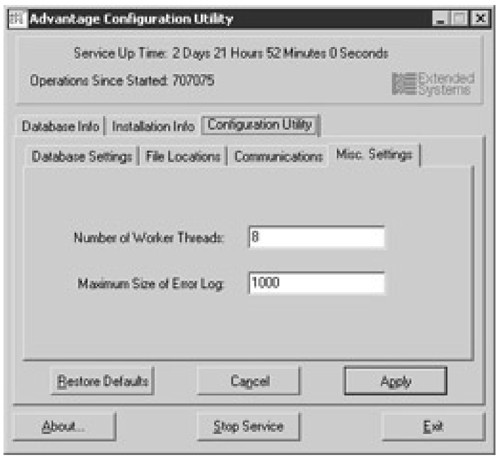
Figure 1-5: The Communications page of the Advantage Configuration Utility
IP Port permits you to set the UDP and TCP ports on which the server will listen for client connections. (&"para">If you want to connect to ADS over the Internet, you set Internet Port to the UDP port that applications (Windows 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.



Service Up Time: 2 Days 21 Hours 52 Minutes 0 Seconds
Operations Since Started: 707075
Number of Worker Threads: 8
Maximum Size of Error Log: 1000

Figure 1-6: The Misc. Settings page of the Advantage Configuration Utility

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.

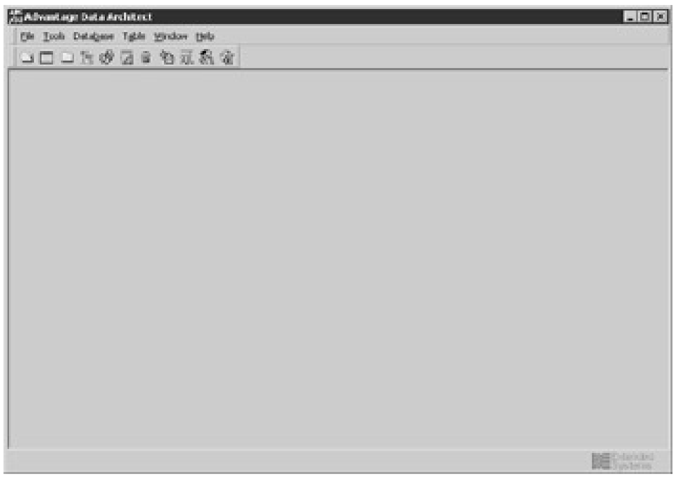


Figure 1-7: The main screen of the Advantage Data Architect

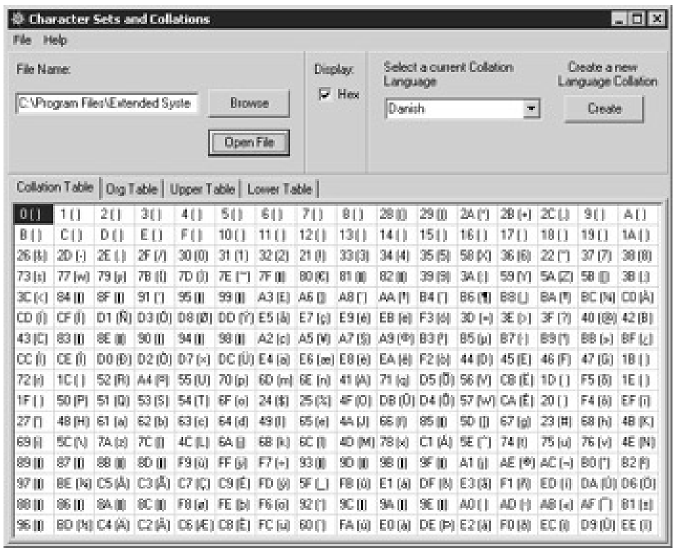
While the Advantage Data Architect makes it easy to design your tables and data dictionaries, there are alternatives. Specifically, all of the database-related configuration capabilities provided by Advantage Data Architect can also be performed using Advantage SQL, as well as by programming directly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



File Name: C:\Program Files\Extended Syste

Display: [✓ Hex]

Select a current Collation Language: Danish

Create a new Language Collation

0 [ ]	1 [ ]	2 [ ]	3 [ ]	4 [ ]	5 [ ]	6 [ ]	7 [ ]	8 [ ]	28 [(]	29 [)]	2A [*]	2B [+]	2C [,]	9 [ ]	A [ ]
B [ ]	C [ ]	D [ ]	E [ ]	F [ ]	10 [ ]	11 [ ]	12 [ ]	13 [ ]	14 [ ]	15 [ ]	16 [ ]	17 [ ]	18 [ ]	19 [ ]	1A [ ]
26 [&]	2D [-]	2E [.]	2F [/]	30 [0]	31 [1]	32 [2]	21 [!]	33 [3]	34 [4]	35 [5]	58 [X]	36 [6]	22 ["]	37 [7]	38 [8]
73 [s]	77 [w]	79 [y]	7B [{]	7D [}]	7E [~]	7F []	80 [€]	81 []	82 [‚]	39 [9]	3A [:]	59 [Y]	5A [Z]	5B [[]	3B [;]
3C [<]	84 [„]	8F []	91 [‘]	95 [•]	99 [™]	A3 [£]	A6 [¦]	A8 [¨]	AA [ª]	B4 [´]	B6 [¶]	B8 [¸]	BA [º]	BC [¼]	C0 [À]
CD [Í]	CF [Ï]	D1 [Ñ]	D3 [Ó]	D8 [Ø]	DD [Ý]	E5 [å]	E7 [ç]	E9 [é]	EB [ë]	F3 [ó]	3D [=]	3E [>]	3F [?]	40 [@]	42 [B]
43 [C]	83 [ƒ]	8E [Ž]	90 []	94 [”]	98 [˜]	A2 [¢]	A5 [¥]	A7 [§]	A9 [©]	B3 [³]	B5 [µ]	B7 [·]	B9 [¹]	BB [»]	BF [¿]
CC [Ì]	CE [Î]	D0 [Ð]	D2 [Ò]	D7 [×]	DC [Ü]	E4 [ä]	E6 [æ]	E8 [è]	EA [ê]	F2 [ò]	44 [D]	45 [E]	46 [F]	47 [G]	1B [ ]
72 [r]	1C [ ]	52 [R]	A4 [¤]	55 [U]	70 [p]	6D [m]	6E [n]	41 [A]	71 [q]	D5 [Õ]	56 [V]	CB [Ë]	1D [ ]	F5 [õ]	1E [ ]
1F [ ]	50 [P]	51 [Q]	53 [S]	54 [T]	6F [o]	24 [$]	25 [%]	4F [O]	DB [Û]	D4 [Ô]	57 [W]	CA [Ê]	20 [ ]	F4 [ô]	EF [ï]
27 [']	48 [H]	61 [a]	62 [b]	63 [c]	64 [d]	49 [I]	65 [e]	4A [J]	66 [f]	85 […]	5D []]	67 [g]	23 [#]	68 [h]	4B [K]
69 [i]	5C [\]	7A [z]	7C [|]	4C [L]	6A [j]	6B [k]	6C [l]	4D [M]	78 [x]	C1 [Á]	5E [^]	74 [t]	75 [u]	76 [v]	4E [N]
89 [‰]	87 [‡]	88 [ˆ]	8D []	F9 [ù]	FF [ÿ]	F7 [÷]	93 [“]	9D []	9B [›]	9F [Ÿ]	A1 [¡]	AE [®]	AC [¬]	B0 [°]	B2 [²]
97 [—]	BE [¾]	C5 [Å]	C3 [Ã]	C7 [Ç]	C9 [É]	FD [ý]	5F [_]	FB [û]	E1 [á]	DF [ß]	E3 [ã]	F1 [ñ]	ED [í]	DA [Ú]	D6 [Ö]
8B [‹]	86 [†]	8A [Š]	8C [Œ]	F8 [ø]	FE [þ]	F6 [ö]	92 [’]	9C [œ]	9A [š]	9E [ž]	A0 [ ]	AD [-]	AB [«]	AF [¯]	B1 [±]
96 [–]	BD [½]	C4 [Ä]	C2 [Â]	C6 [Æ]	C8 [È]	FC [ü]	60 [`]	FA [ú]	E0 [à]	DE [Þ]	E2 [â]	F0 [ð]	EC [ì]	D9 [Ù]	EE [î]

Figure 1-8: The ANSI Collation Utility

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&"para">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.