



# Introducing SQL Anywhere<sup>®</sup> Studio

Part number: DC38120-01-0902-01  
Last modified: October 2004

---

Copyright © 1989–2004 Sybase, Inc. Portions copyright © 2001–2004 iAnywhere Solutions, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of iAnywhere Solutions, Inc. iAnywhere Solutions, Inc. is a subsidiary of Sybase, Inc.

Sybase, SYBASE (logo), AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, ASEP, AvantGo, AvantGo Application Alerts, AvantGo Mobile Delivery, AvantGo Mobile Document Viewer, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client Services, Client-Library, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, Dynamo, e-ADK, E-Anywhere, e-Biz Integrator, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, E-Whatever, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Channel, M-Business Network, M-Business Server, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, ML Query, MobiCATS, My AvantGo, My AvantGo Media Channel, My AvantGo Mobile Marketing, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASiS, OASiS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open Client/Server, Open Client/Server Interfaces, Open ClientConnect, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Orchestration Studio, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power Through Knowledge, power.stop, Power++, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, Powersoft Portfolio, Powersoft Professional, PowerStage, PowerStudio, PowerTips, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, QAnywhere, Rapport, Relational Beans, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report Workbench, Report-Execute, Resource Manager, RW-DisplayLib, RW-Library, S.W.I.F.T. Message Format Libraries, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILLS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL Server SNMP SubAgent, SQL Server/CFT, SQL Server/DBM, SQL SMART, SQL Station, SQL Toolset, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase User Workbench, Sybase Virtual Server Architecture, SybaseWare, Syber Financial, SyberAssist, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Versacore, Viewer, VisualWriter, VQL, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, WarehouseArchitect, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, and XP Server are trademarks of Sybase, Inc. or its subsidiaries.

All other trademarks are property of their respective owners.

---

# Contents

<b>About This Manual</b>	<b>ix</b>
SQL Anywhere Studio documentation . . . . .	x
Accessing information online . . . . .	xiii
Documentation conventions . . . . .	xvi
The Adaptive Server Anywhere sample database . . . . .	xviii
The CustDB sample database . . . . .	xix
Finding out more and providing feedback . . . . .	xx
<b>I SQL Anywhere Studio Overview</b>	<b>1</b>
<b>1 Introducing SQL Anywhere Studio</b>	<b>3</b>
Welcome to SQL Anywhere Studio . . . . .	4
Installing SQL Anywhere Studio . . . . .	6
<b>2 Introducing Adaptive Server Anywhere and UltraLite</b>	<b>9</b>
Introduction to Adaptive Server Anywhere . . . . .	10
Intended uses . . . . .	11
Hallmarks . . . . .	12
The database server . . . . .	14
Applications . . . . .	15
Introduction to UltraLite . . . . .	16
Comparing Adaptive Server Anywhere and UltraLite . . . . .	18
<b>3 Introducing Replication Technologies</b>	<b>21</b>
Introduction . . . . .	22
Consolidated and remote databases . . . . .	23
Propagation methods . . . . .	26
Sybase replication technologies . . . . .	29
<b>II Platform Support</b>	<b>35</b>
<b>4 SQL Anywhere Studio for Windows CE</b>	<b>37</b>
Getting started . . . . .	38
Using the sample applications . . . . .	45
Configuring Windows CE databases . . . . .	53
Using the database server on Windows CE . . . . .	62
Using the administration utilities . . . . .	69

---

SQL Anywhere Studio features not supported on Windows CE 80

<b>5</b>	<b>SQL Anywhere Studio Supported Platforms</b>	<b>95</b>
	Introduction . . . . .	96
	Windows and NetWare operating systems . . . . .	98
	UNIX, Linux, and Macintosh operating systems . . . . .	103
	MobiLink synchronization consolidated databases . . . . .	107
	iAnywhere Solutions ODBC drivers supported platforms . . . . .	108
	UltraLite target platforms . . . . .	109
	Operating system versions . . . . .	111
<b>III</b>	<b>Working with Databases</b>	<b>115</b>
<b>6</b>	<b>The Architecture of Database Applications</b>	<b>117</b>
	Relational database concepts . . . . .	118
	The pieces of a database system . . . . .	123
	How the pieces fit together . . . . .	125
	Multi-tier computing architecture . . . . .	128
	Using multiple databases . . . . .	129
	Application programming interfaces . . . . .	131
	Inside Adaptive Server Anywhere . . . . .	136
<b>7</b>	<b>Selecting Data from Database Tables</b>	<b>139</b>
	SQL and database computing . . . . .	140
	SELECT statement . . . . .	142
	Selecting a complete table . . . . .	143
	Selecting columns from a table . . . . .	145
	Ordering query results . . . . .	148
	Selecting rows from a table . . . . .	151
<b>8</b>	<b>Selecting Data from Multiple Tables</b>	<b>157</b>
	Introduction . . . . .	158
	Joining tables using the cross product . . . . .	159
	Using the ON phrase to restrict a join . . . . .	160
	Joining tables using key joins . . . . .	162
	Joining tables using natural joins . . . . .	164
	Joining tables using outer joins . . . . .	166
<b>9</b>	<b>Selecting Data Using Subqueries</b>	<b>167</b>
	Introducing subqueries . . . . .	168
	Single-row and multiple-row subqueries . . . . .	170
	Using subqueries instead of joins . . . . .	172

---

<b>10</b>	<b>Selecting Aggregate Data</b>	<b>175</b>
	Summarizing data . . . . .	176
	A first look at aggregate functions . . . . .	177
	Applying aggregate functions to grouped data . . . . .	178
	Restricting groups . . . . .	180
<b>11</b>	<b>Updating the Database</b>	<b>183</b>
	Introduction . . . . .	184
	Adding rows to a table . . . . .	185
	Modifying rows in a table . . . . .	186
	Deleting rows . . . . .	187
	Grouping changes into transactions . . . . .	188
	Integrity checking . . . . .	191
<b>IV</b>	<b>SQL Anywhere Studio Tutorials</b>	<b>195</b>
<b>12</b>	<b>The Sample Database</b>	<b>197</b>
	About the sample database . . . . .	198
	Lesson 1: Make a copy of the sample database . . . . .	200
	Lesson 2: Start the Adaptive Server Anywhere database server	201
	Lesson 3: Displaying the database server window . . . . .	203
	Lesson 4: Shut down the database server . . . . .	205
	Summary . . . . .	206
<b>13</b>	<b>Making the Connection</b>	<b>207</b>
	About connections . . . . .	208
	About ODBC data sources . . . . .	209
	Lesson 1: Create an ODBC data source . . . . .	210
	Lesson 2: Connect using the Sample ODBC data source . . . . .	213
	Lesson 3: Other means of connecting . . . . .	214
	Lesson 4: Disconnecting from the sample database . . . . .	215
	Summary . . . . .	216
<b>14</b>	<b>Using Interactive SQL</b>	<b>217</b>
	About Interactive SQL . . . . .	218
	Lesson 1: Start Interactive SQL . . . . .	219
	Lesson 2: The Interactive SQL interface . . . . .	220
	Lesson 3: Displaying data using Interactive SQL . . . . .	226
	Lesson 4: Working with SQL statements . . . . .	232
	Summary . . . . .	239
<b>15</b>	<b>Managing Databases with Sybase Central</b>	<b>241</b>
	About Sybase Central . . . . .	242

---

Lesson 1: Start Sybase Central and connect . . . . .	243
Lesson 2: The Sybase Central interface . . . . .	244
Lesson 3: Viewing the sample database . . . . .	246
Lesson 4: Create and edit tables . . . . .	251
Lesson 5: Manage users and groups . . . . .	253
Lesson 6: View and edit stored procedures . . . . .	255
Lesson 7: Back up your database . . . . .	258
Restore the sample database . . . . .	259
Summary . . . . .	260
<b>16 Designing and Building a Database</b>	<b>261</b>
About designing a new database . . . . .	262
Designing and building a simple database . . . . .	263
Lesson 1: Create a database file . . . . .	264
Lesson 2: Connect to your database . . . . .	266
Lesson 3: Design and create a table . . . . .	267
Lesson 4: Identify and create primary keys . . . . .	269
Lesson 5: Design column properties . . . . .	271
Lesson 6: Design and create relationships between tables . . . . .	274
Summary . . . . .	277
<b>17 Synchronizing Databases with MobiLink</b>	<b>279</b>
About MobiLink . . . . .	280
Introduction . . . . .	281
Lesson 1: Create your databases . . . . .	282
Lesson 2: Prepare the databases for synchronization . . . . .	286
Lesson 3: Start the MobiLink synchronization server . . . . .	289
Lesson 4: Run the MobiLink synchronization client utility . . . . .	290
Summary . . . . .	291
<b>18 Replicating Data with SQL Remote</b>	<b>293</b>
About SQL Remote . . . . .	294
Lesson 1: Getting Started . . . . .	295
Lesson 2: Set up the consolidated database . . . . .	296
Lesson 3: Set up the remote database . . . . .	300
Lesson 4: Replicate data . . . . .	302
Lesson 5: Restore the database and database settings . . . . .	305
Summary . . . . .	307
<b>19 Designing Databases with PowerDesigner</b>	<b>309</b>
About PowerDesigner . . . . .	310
Lesson 1: Getting Started . . . . .	312
Lesson 2: Add a column . . . . .	317
Lesson 3: Check your work . . . . .	320

---

Lesson 4: Save changes and generate database . . . . .	321
Summary . . . . .	324
<b>20 Creating Reports with InfoMaker</b>	<b>327</b>
About InfoMaker . . . . .	328
Lesson 1: Getting Started . . . . .	329
Lesson 2: Create a basic report . . . . .	330
Lesson 3: Enhance your report . . . . .	333
Summary . . . . .	337
<b>21 Microsoft Visual Basic Quick Start</b>	<b>339</b>
Tutorial: Developing a Visual Basic application . . . . .	340
<b>V Appendix</b>	<b>343</b>
<b>22 Glossary</b>	<b>345</b>
<b>Index</b>	<b>363</b>





---

# About This Manual

Subject	<p>This book introduces SQL Anywhere Studio, the complete relational data-management and synchronization system for mobile, embedded, and workgroup computing.</p> <p>SQL Anywhere Studio is a set of software components for working with relational data. It includes the Adaptive Server Anywhere and UltraLite relational database-management systems, as well as data synchronization and replication technology. It also includes applications for designing and deploying databases, and for creating custom reports and data entry forms.</p>
Audience	<p>This book is for all application developers and database administrators using SQL Anywhere Studio.</p>
Before you begin	<p>This book assumes an elementary familiarity with relational databases and SQL.</p>

---

# SQL Anywhere Studio documentation

## The SQL Anywhere Studio documentation

This book is part of the SQL Anywhere documentation set. This section describes the books in the documentation set and how you can use them.

The SQL Anywhere Studio documentation is available in a variety of forms: in an online form that combines all books in one large help file; as separate PDF files for each book; and as printed books that you can purchase. The documentation consists of the following books:

- ◆ **Introducing SQL Anywhere Studio** This book provides an overview of the SQL Anywhere Studio database management and synchronization technologies. It includes tutorials to introduce you to each of the pieces that make up SQL Anywhere Studio.
- ◆ **What's New in SQL Anywhere Studio** This book is for users of previous versions of the software. It lists new features in this and previous releases of the product and describes upgrade procedures.
- ◆ **Adaptive Server Anywhere Database Administration Guide** This book covers material related to running, managing, and configuring databases and database servers.
- ◆ **Adaptive Server Anywhere SQL User's Guide** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.
- ◆ **Adaptive Server Anywhere SQL Reference Manual** This book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.
- ◆ **Adaptive Server Anywhere Programming Guide** This book describes how to build and deploy database applications using the C, C++, and Java programming languages. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by those tools. It also describes the Adaptive Server Anywhere ADO.NET data provider.
- ◆ **Adaptive Server Anywhere SNMP Extension Agent User's Guide** This book describes how to configure the Adaptive Server Anywhere SNMP Extension Agent for use with SNMP management applications to manage Adaptive Server Anywhere databases.
- ◆ **Adaptive Server Anywhere Error Messages** This book provides a complete listing of Adaptive Server Anywhere error messages together with diagnostic information.

- 
- ◆ **SQL Anywhere Studio Security Guide** This book provides information about security features in Adaptive Server Anywhere databases. Adaptive Server Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. This book may be of interest to those who wish to run the current version of Adaptive Server Anywhere in a manner equivalent to the C2-certified environment.
  - ◆ **MobiLink Administration Guide** This book describes how to use the MobiLink data synchronization system for mobile computing, which enables sharing of data between a single Oracle, Sybase, Microsoft or IBM database and many Adaptive Server Anywhere or UltraLite databases.
  - ◆ **MobiLink Clients** This book describes how to set up and synchronize Adaptive Server Anywhere and UltraLite remote databases.
  - ◆ **MobiLink Server-Initiated Synchronization User's Guide** This book describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization from the consolidated database.
  - ◆ **MobiLink Tutorials** This book provides several tutorials that walk you through how to set up and run MobiLink applications.
  - ◆ **QAnywhere User's Guide** This manual describes MobiLink QAnywhere, a messaging platform that enables the development and deployment of messaging applications for mobile and wireless clients, as well as traditional desktop and laptop clients.
  - ◆ **iAnywhere Solutions ODBC Drivers** This book describes how to set up ODBC drivers to access consolidated databases other than Adaptive Server Anywhere from the MobiLink synchronization server and from Adaptive Server Anywhere remote data access.
  - ◆ **SQL Remote User's Guide** This book describes all aspects of the SQL Remote data replication system for mobile computing, which enables sharing of data between a single Adaptive Server Anywhere or Adaptive Server Enterprise database and many Adaptive Server Anywhere databases using an indirect link such as e-mail or file transfer.
  - ◆ **SQL Anywhere Studio Help** This book includes the context-sensitive help for Sybase Central, Interactive SQL, and other graphical tools. It is not included in the printed documentation set.
  - ◆ **UltraLite Database User's Guide** This book is intended for all UltraLite developers. It introduces the UltraLite database system and provides information common to all UltraLite programming interfaces.

- 
- ◆ **UltraLite Interface Guides** A separate book is provided for each UltraLite programming interface. Some of these interfaces are provided as UltraLite components for rapid application development, and others are provided as static interfaces for C, C++, and Java development.

In addition to this documentation set, PowerDesigner and InfoMaker include their own online documentation.

Documentation formats SQL Anywhere Studio provides documentation in the following formats:

- ◆ **Online documentation** The online documentation contains the complete SQL Anywhere Studio documentation, including both the books and the context-sensitive help for SQL Anywhere tools. The online documentation is updated with each maintenance release of the product, and is the most complete and up-to-date source of documentation.

To access the online documentation on Windows operating systems, choose Start ► Programs ► SQL Anywhere 9 ► Online Books. You can navigate the online documentation using the HTML Help table of contents, index, and search facility in the left pane, as well as using the links and menus in the right pane.

To access the online documentation on UNIX operating systems, see the HTML documentation under your SQL Anywhere installation.

- ◆ **PDF books** The SQL Anywhere books are provided as a set of PDF files, viewable with Adobe Acrobat Reader.

The PDF books are accessible from the online books, or from the Windows Start menu.

- ◆ **Printed books** The complete set of books is available from Sybase sales or from eShop, the Sybase online store at <http://eshop.sybase.com/eshop/documentation>.

---

## Accessing information online

The SQL Anywhere Studio online documentation features a number of tools to help you navigate through the information.

### Online books (PDF)

Online books (PDF) are exact copies of the printed documentation in PDF format, viewable with Adobe Acrobat. The presence and appearance of some of the following features depends on which version of Acrobat or Acrobat Reader you are using.

- ◆ **Bookmarks tab** The Bookmarks tab displays a list of all the chapters in the current book. You can click any chapter or page to view the expand or collapse the headings and view the structure of the document. Clicking a heading displays the page in the right side of the window. The Bookmarks tab provides an overview of the structure of the book and quick access to specific chapters and pages.
- ◆ **Active links** Active links, such as see also references, are highlighted in blue. When you move your cursor over the link, the hand turns to a pointing finger. Clicking the link takes you to the place in the documentation the link points to, either in the current book or in other books. Links between books do not work properly when Acrobat is run as a plugin to certain browsers. They do work properly in Internet Explorer, or when using Acrobat directly.
- ◆ **Forward/Back buttons** The Forward and Back buttons at the top left side of the help window let you retrace your steps, and revisit the pages in the order you visited them. Each time you click the Back button, the topic moves to the most recently visited page prior to the one currently displayed. The Forward button becomes enabled after you click the Back button at least once, and remains enabled until you return to the most recently visited page.
- ◆ **Go to start/Go to end buttons** Clicking the Go to start button returns you to the very first page of the book, and clicking the Go to end button advances you to the very last page of the book.

### Online books

Online books are exact copies of the printed documentation in Microsoft HTML Help format.

- ◆ **Contents tab** The Contents tab displays a list of all the books in the SQL Anywhere Studio collection. You can double click any book to view the chapters within that book, and you can click each chapter to view its pages. The Contents tab provides an overview of the structure of the documentation and quick access to specific chapters and pages.

- 
- ◆ **Page header** The page header is located at the top of each page and describes the location of the current page. The current location is described as book title ► chapter title ► page title. You can click the book or chapter titles to move to that part of the book. Notice that as you navigate through the pages on the right side of the window, the placeholder on the Contents tab on the left side of the window moves accordingly.
  - ◆ **Index tab** The index contains keywords associated with chapters and pages. You can type a word or phrase in the top box of the Index tab to jump to a topic in the index in the scroll box below. When you find the entry you are looking for, simply double-click the entry to display the page.
  - ◆ **Search tab** Type a word or phrase in the top box of the Search tab, and click List Topics to display all topics that contain that word or phrase in the scroll box below. If you want to find an exact match, put the word or phrase in double quotation marks (“”). You can further refine your selection by selecting the checkboxes at the bottom of the page, or using Boolean search terms by clicking the arrow next to the search box. Search previous results lets you narrow your next search to the topics displayed in your previous search. Match similar words looks for words that are close in spelling to the searched word. Search titles only will search only the titles for the desired word. When you find the entry you are looking for, simply double-click the entry to display the page.
  - ◆ **Favorites tab** When you find a page that you know you want to access again, add it to your Favorites list for quick access. Display the page you want to save, click the Favorites tab, then click the Add button at the bottom of the window. Clicking Add adds the topic listed in the Current topic box (the topic to the Topics list. You can modify your favorites list by selecting a topic and clicking Remove. Favorites entries are preserved between upgrades of the online books.
  - ◆ **Previous/Next buttons** The Previous and Next buttons at the top right side of the help window are equivalent to turning the pages of a book. Previous turns the page back one topic (moves to the previous page in chronological order). Next turns the page forward one topic (moves to the subsequent page in chronological order).
  - ◆ **Forward/Back buttons** The Forward and Back buttons on the menu bar let you retrace your steps, and revisit the pages in the order you visited them. Each time you click the Back button, the topic moves to the most recently visited page prior to the one currently displayed. The Forward button becomes enabled after you click the Back button at least once, and remains enabled until you return to the most recently visited page.

- 
- ◆ **Web Links button** The Web Links button at the top of the help window offers quick access to iAnywhere Solutions resources, including the iAnywhere web site, tech support, registration, downloads, and the iAnywhere Solutions developer community.
  - ◆ **Browse button** Click the Browse button for quick access to tutorials, a list of actions you can perform, and a glossary of terms.
  - ◆ **Collapse button** Some pages contain headings that are collapsible, as indicated by the plus (+) or minus (-) sign immediately to the left of the heading. You can click the Collapse button to either close all the headings, or click the Expand button to open all the headings.
  - ◆ **PDF button** Clicking the PDF button opens the current book in PDF format.

---

## Documentation conventions

This section lists the typographic and graphical conventions used in this documentation.

### Syntax conventions

The following conventions are used in the SQL syntax descriptions:

- ◆ **Keywords** All SQL keywords appear in upper case, like the words ALTER TABLE in the following example:

```
ALTER TABLE [ owner.]table-name
```

- ◆ **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown like the words *owner* and *table-name* in the following example:

```
ALTER TABLE [ owner.]table-name
```

- ◆ **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots), like *column-constraint* in the following example:

```
ADD column-definition [ column-constraint, ... ]
```

One or more list elements are allowed. In this example, if more than one is specified, they must be separated by commas.

- ◆ **Optional portions** Optional portions of a statement are enclosed by square brackets.

```
RELEASE SAVEPOINT [ savepoint-name ]
```

These square brackets indicate that the *savepoint-name* is optional. The square brackets should not be typed.

- ◆ **Options** When none or only one of a list of items can be chosen, vertical bars separate the items and the list is enclosed in square brackets.

```
[ ASC | DESC ]
```

For example, you can choose one of ASC, DESC, or neither. The square brackets should not be typed.

- ◆ **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces and a bar is used to separate the options.

```
[ QUOTES { ON | OFF } ]
```

If the QUOTES option is used, one of ON or OFF must be provided. The brackets and braces should not be typed.

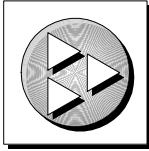


---

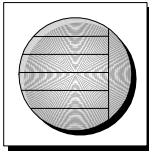
## Graphic icons

The following icons are used in this documentation.

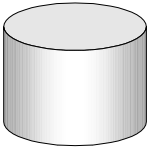
- ◆ A client application.



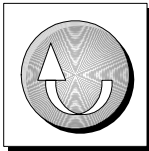
- ◆ A database server, such as Sybase Adaptive Server Anywhere.



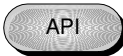
- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ◆ Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink Synchronization Server and the SQL Remote Message Agent.



- ◆ A programming interface.



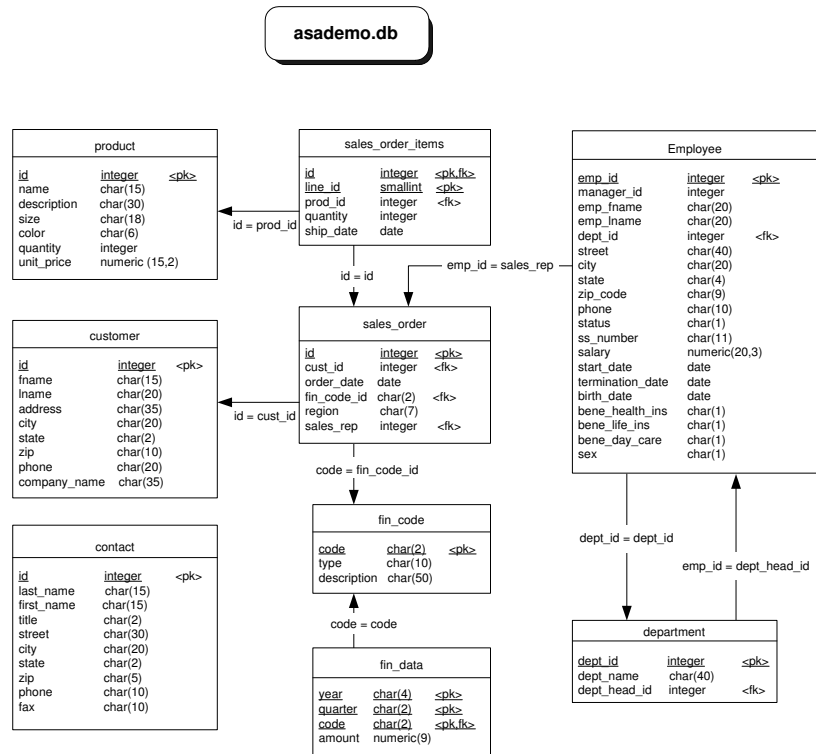
# The Adaptive Server Anywhere sample database

Many of the examples throughout the documentation use the Adaptive Server Anywhere sample database.

The sample database is held in a file named *asademo.db*, and is located in your SQL Anywhere directory.

The sample database represents a small company. It contains internal information about the company (employees, departments, and finances) as well as product information and sales information (sales orders, customers, and contacts). All information in the database is fictional.

The following figure shows the tables in the sample database and how they relate to each other.



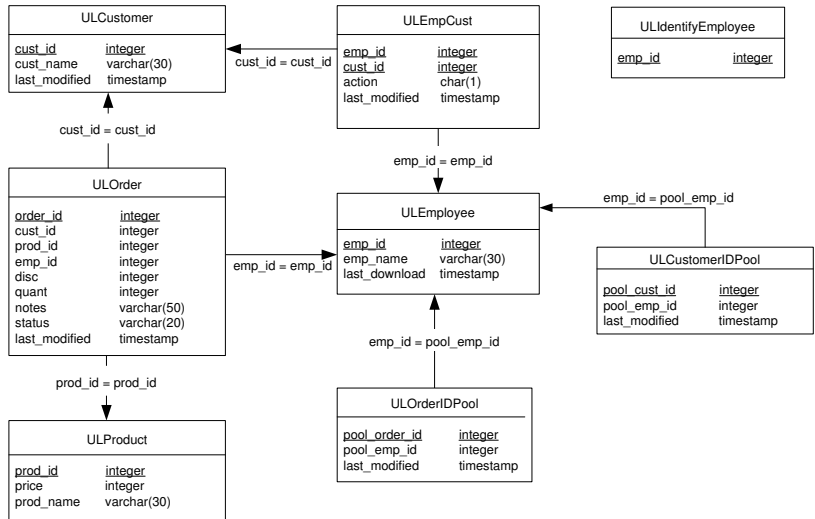
# The CustDB sample database

Many of the examples in the MobiLink and UltraLite documentation use the UltraLite sample database.

The reference database for the UltraLite sample database is held in a file named *custdb.db*, and is located in the *Samples\UltraLite\CustDB* subdirectory of your SQL Anywhere directory. A complete application built on this database is also supplied.

The sample database is a sales-status database for a hardware supplier. It holds customer, product, and sales force information for the supplier.

The following figure shows the tables in the CustDB database and how they are related to each other.



---

## Finding out more and providing feedback

We would like to receive your opinions, suggestions, and feedback on this documentation.

You can provide feedback on this documentation and on the software through newsgroups set up to discuss SQL Anywhere technologies. These newsgroups can be found on the *forums.sybase.com* news server.

The newsgroups include the following:

- ◆ [sybase.public.sqlanywhere.general](#)
- ◆ [sybase.public.sqlanywhere.linux](#)
- ◆ [sybase.public.sqlanywhere.mobilink](#)
- ◆ [sybase.public.sqlanywhere.product\\_futures\\_discussion](#)
- ◆ [sybase.public.sqlanywhere.replication](#)
- ◆ [sybase.public.sqlanywhere.ultralite](#)
- ◆ [ianywhere.public.sqlanywhere.qanywhere](#)

### **Newsgroup disclaimer**

iAnywhere Solutions has no obligation to provide solutions, information or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and insure its operation and availability.

iAnywhere Solutions Technical Advisors as well as other staff assist on the newsgroup service when they have time available. They offer their help on a volunteer basis and may not be available on a regular basis to provide solutions and information. Their ability to help is based on their workload.

You can e-mail comments and suggestions to the SQL Anywhere documentation team at [iasdoc@ianywhere.com](mailto:iasdoc@ianywhere.com). Although we do not undertake to reply to e-mails at that address, you can be sure we will read your suggestions with interest.

PART I

# SQL ANYWHERE STUDIO OVERVIEW

This part introduces SQL Anywhere Studio and its database management and replication technologies.



---

## CHAPTER 1

# Introducing SQL Anywhere Studio

About this chapter

This chapter introduces the pieces that make up SQL Anywhere Studio.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Welcome to SQL Anywhere Studio</a>	4
<a href="#">Installing SQL Anywhere Studio</a>	6

---

# Welcome to SQL Anywhere Studio

With SQL Anywhere Studio, you can deliver information to workgroup, mobile, and embedded database systems throughout an entire organization.

SQL Anywhere Studio includes the following components.

Relational database systems

◆ **Adaptive Server Anywhere** The relational database at the core of the product is a transaction-based SQL database designed for personal and workgroup use. Adaptive Server Anywhere runs on a wide range of operating systems, including many flavors of Windows and UNIX, as well as on Novell NetWare. It runs on hardware ranging from multiple-CPU workgroup servers to the most modest PCs, as well as on Windows CE devices.

◆ **UltraLite** For building and deploying relational database applications on small devices, including the Palm Computing Platform and Windows CE. UltraLite has built-in support for MobiLink synchronization.

UltraLite lets you build relational databases with less than 50 kb of disk space, and is specifically intended for small devices.

Data synchronization technologies

◆ **MobiLink** For two-way synchronization of data between a central database and many remote UltraLite or Adaptive Server Anywhere databases. The central database can be Adaptive Server Anywhere, Adaptive Server Enterprise, or another DBMS such as Oracle, Microsoft SQL Server, or IBM DB2.

◆ **SQL Remote** For two-way, message-based replication of data between a central database and many remote databases. With SQL Remote, you can replicate data between laptop computers and a central database, using e-mail or dial-up access.

◆ **Replication Agent** For replicating data from Adaptive Server Anywhere databases to other databases via Sybase Replication Server.

Development, design, and administration tools

◆ **InfoMaker** For querying databases and creating sophisticated and effective custom reports of data. InfoMaker is also a personal data assistant that lets you work with data in many ways.

◆ **PowerDesigner** For designing, generating, documenting, and maintaining databases.

◆ **Management and development tools** SQL Anywhere Studio includes the Sybase Central database management tool and the Interactive SQL database utility, as well as a query editor and a stored procedure debugger.

SQL Anywhere Studio includes an optionally-installable accessibility enablement module. This component provides the Sun AccessBridge



module, which is loaded whenever you use Sybase Central or Interactive SQL. Third-party software such as screen readers make use of this module to provide access to software features.

InfoMaker and PowerDesigner are available only on Windows operating systems. However, they can be used as clients of a server running on any supported operating system.

Separately-licensable components

The following components are separately licensable and must be ordered before you can install them.

- ◆ **Java option** The Java virtual machine and runtime classes that enable the use of Java in the database must be ordered separately.
- ◆ **SQL Anywhere Studio security option** The software for strong encryption must be ordered separately. Strong encryption can be used in the following cases:
  - Transport-layer security between an Adaptive Server Anywhere database server and client.
  - Transport-layer security between a MobiLink synchronization server and client.
  - AES\_FIPS database encryption. (AES database file encryption other than AES\_FIPS is included in the base package and does not require separate licensing.)

To order the Java and security components, you have the following options:

- ◆ Visit <http://www.sybase.com/detail?id=1015780>.
- ◆ See the card in your SQL Anywhere Studio package.
- ◆ Call iAnywhere at (800) 801-2069.

---

# Installing SQL Anywhere Studio

How you install SQL Anywhere Studio depends on the operating system you are using. You must ensure that you are properly licensed before installing the software.

## ❖ To install SQL Anywhere Studio (Windows operating systems)

1. Start the installation by running the *setup.exe* program in the root directory of the SQL Anywhere Studio CD-ROM. Follow the instructions in the setup wizard.

The setup program allows you to choose which of the components you wish to install.

## ❖ To install SQL Anywhere Studio (Novell NetWare or Windows CE)

1. You must install Adaptive Server Anywhere for NetWare from a machine connected to the NetWare server and running a Microsoft Windows operating system. Run the SQL Anywhere Studio setup program and choose NetWare or Windows CE installation.

## ❖ To install SQL Anywhere Studio (UNIX)

1. The installation instructions depend on which UNIX operating system you are using. For more information, consult the separate *Read Me First* booklet, which is included in SQL Anywhere Studio for UNIX.

## The SQL Anywhere program group

For Windows operating systems, after the software is installed, you will have a SQL Anywhere **program group**. You can reach the program group by clicking the Start button and choosing Programs ► SQL Anywhere 9.

Installing SQL Anywhere Studio under UNIX does not provide a program group.

### Program group items

The program group contains some or all of the following items. The items you see depend on the choices you made when installing the software.

- ◆ **Adaptive Server Anywhere** Contains the items:
  - **Interactive SQL** Starts the Interactive SQL utility for sending SQL statements to a database.
  - **Network Server Sample** Starts the network database server and loads the sample database.

- **ODBC Administrator** Starts the ODBC Administrator program for setting up and editing ODBC data sources.
- **Personal Server Sample** Starts the personal server and loads the sample database.
- ◆ **MobiLink program group** To access MobiLink synchronization programs and samples.
- ◆ **UltraLite program group** To access UltraLite programs and samples.
- ◆ **Check for updates** To access a web page with information about the latest updates available for your version of SQL Anywhere Studio.
- ◆ **iAnywhere Online Resources** Opens a web page with information about iAnywhere Solutions.
- ◆ **Online Books** Opens the online documentation for Adaptive Server Anywhere.
- ◆ **Sybase Central** Starts Sybase Central, the database management tool.

In addition, you may have items for InfoMaker and PowerDesigner.



---

## CHAPTER 2

# Introducing Adaptive Server Anywhere and UltraLite

About this chapter

This chapter introduces the Adaptive Server Anywhere relational database system, as well as the UltraLite, the small-footprint Adaptive Server Anywhere database. It describes their uses, features, and differences.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction to Adaptive Server Anywhere</a>	10
<a href="#">Intended uses</a>	11
<a href="#">Hallmarks</a>	12
<a href="#">The database server</a>	14
<a href="#">Applications</a>	15
<a href="#">Introduction to UltraLite</a>	16
<a href="#">Comparing Adaptive Server Anywhere and UltraLite</a>	18

---

## Introduction to Adaptive Server Anywhere

This chapter describes the particular hallmarks of Adaptive Server Anywhere and describes the uses to which it can be put.

Adaptive Server Anywhere provides a series of tools for storing and managing data. You can use these tools to enter data into your database, to change your database structure, and to view or alter your data.

The Adaptive Server Anywhere relational database-management system is the heart of SQL Anywhere Studio. Adaptive Server Anywhere is designed for tasks that require a full-featured SQL database. It is designed to operate in varied environments. It takes advantage of available memory and CPU resources, providing good performance in environments with ample resources. It also operates well in environments with limited physical and database administration resources, including mobile computing environments, embedded database use, and use as a database server for small and medium businesses.

## Intended uses

Roles for which Adaptive Server Anywhere is ideally suited include the following:

- ◆ **A small and medium business database server** Adaptive Server Anywhere is built to handle the requirements of small and medium businesses, with anywhere from a few users to several hundred users. It provides a high-performance database for workgroups and companies, well-suited for (but not limited to) environments where administration and hardware resources are limited.

Adaptive Server Anywhere can employ multiple CPUs and use up to 64 Gb of memory. Our customers have Adaptive Server Anywhere databases with tens of gigabytes of data in production use.

- ◆ **An embedded database** Many applications require a database “behind the scenes”. Personal Information Managers, document management systems, network monitoring applications—just about any application that stores information. Adaptive Server Anywhere is intended to be the database for these applications. The UltraLite deployment option is intended for embedded environments that have very restricted resources.

A key feature of embedded databases is that they be able to run entirely without administration. Adaptive Server Anywhere has demonstrated this facility in many demanding commercial applications.

- ◆ **Mobile computing** Laptop and notebook computers are now common in many workplaces. Adaptive Server Anywhere is intended to be the SQL database for these computers. With MobiLink synchronization and SQL Remote replication, Adaptive Server Anywhere extends transaction-based computing throughout the enterprise. The UltraLite deployment option and MobiLink synchronization technology provide full database functionality on devices with limited resources, such as hand-held computers.

---


## Hallmarks

Adaptive Server Anywhere is built around the following technological hallmarks:

- ◆ **Full SQL relational database-management system** Adaptive Server Anywhere is a transaction-processing relational database-management system (RDBMS), with full recovery capabilities, online backup, referential integrity actions, stored procedures, triggers, row-level concurrency control, schedules and events, a rich SQL language, and all the features you expect in a full SQL RDBMS.
- ◆ **Economical hardware requirements** Adaptive Server Anywhere requires fewer memory and disk resources than other database-management systems.
- ◆ **Easy to use** Adaptive Server Anywhere is self-tuning and easy to manage. You can use Adaptive Server Anywhere without the extensive database administration efforts usually associated with relational database-management systems.
- ◆ **Standalone and network use** Adaptive Server Anywhere can be used in a standalone manner, for example as an embedded database in a data-centric application, or as a network server in a multi-user client/server or three-tier environment. As an embedded database system, it can be started automatically by an application when required.
- ◆ **High performance** While Adaptive Server Anywhere is designed with simple administration and modest resource requirements in mind, it is a scalable, high-performance DBMS. Adaptive Server Anywhere can run on multiple CPUs, has an advanced query optimizer, and provides performance monitoring and tuning tools.
- ◆ **Industry standard interfaces** Adaptive Server Anywhere provides a native ODBC 3.5 driver for high performance from ODBC applications, and an OLE DB driver for use from ActiveX Data Object (ADO) programming environments. It has an ADO.NET data provider for Adaptive Server Anywhere and it also comes with Sybase jConnect for JDBC, as well as an iAnywhere JDBC driver, and supports embedded SQL and Sybase Open Client interfaces.
- ◆ **A cross-platform solution** Adaptive Server Anywhere can be run on many operating systems, including Windows, Novell NetWare, Sun Solaris, and Linux.

☞ The components available on each platform may differ. For information, see [“SQL Anywhere Studio Supported Platforms” on page 95](#).



System requirements  For more information on supported operating systems for components in SQL Anywhere Studio, see [“SQL Anywhere Studio Supported Platforms” on page 95](#).

## Network software requirements

If you are running an Adaptive Server Anywhere network server, you must have appropriate networking software installed and running.

The Adaptive Server Anywhere network server is available for Windows, Novell NetWare, Linux, and UNIX operating systems.

Adaptive Server Anywhere supports the TCP/IP network protocol and the SPX protocol for Novell NetWare.

---

## The database server

There are two versions of the Adaptive Server Anywhere database server included in the product:

- ◆ **The personal database server** This server is provided for single-user, same-machine use: for example, as an embedded database server. It is also useful for development work.

The name of the personal server executable is as follows:

- On UNIX operating systems, it is *dbeng9*.
- On Windows, except Windows CE, it is *dbeng9.exe*.

- ◆ **The network database server** In addition to the features of the personal server, the network server supports client/server communications across a network. It is provided for multi-user use.

The name of the network server executable is as follows:

- On UNIX operating systems, it is *dsrv9*.
- On Windows, including Windows CE, it is *dsrv9.exe*.
- On Novell NetWare, the server is a NetWare Loadable Module (NLM) called *dsrv9.nlm*.

Same SQL features in each version

The request-processing engine is identical in the two versions of the server. They support exactly the same SQL language, and exactly the same database features. The personal server does not support communications across a network, more than ten concurrent connections, or the use of more than one CPU. Applications developed against a personal server work unchanged against a network server.

## Applications

This section introduces some of the database applications that are supplied with Adaptive Server Anywhere in the SQL Anywhere Studio product. These applications help you design, build, and administer your databases.

Which components you have installed depends on which operating system you are using, what choices you made when installing the software, and whether you received the Adaptive Server Anywhere product itself, or installed Adaptive Server Anywhere as part of another product.

For example, if you have received Adaptive Server Anywhere as part of another product, you may not have both versions of the database server. Similarly, not all components are available on all operating systems. For example, there is no personal server for NetWare, only a network server.

### Sybase Central

Sybase Central provides a graphical user interface for creating and modifying databases and database objects, for inspecting the structure of databases, and for database server administration. You can use it to perform such tasks as creating a new database, adding a table, or adding a column to a table.

### Interactive SQL

Interactive SQL is an application for typing and sending SQL statements to a database. Interactive SQL allows you to query and alter data in your database, as well as modify the structure of your database. Everything that can be done in Sybase Central can be done in Interactive SQL, but administration tasks are easier in Sybase Central.

### Utilities

A set of utilities is available for carrying out administration tasks such as backing up a database. Utilities are useful for including in batch files for repeated use.

☞ For more information about administration utilities, see “[Database Administration Utilities](#)” [*ASA Database Administration Guide*, page 493].

---

## Introduction to UltraLite

The appearance of small computing devices such as handheld computers, pagers, and mobile phones creates a demand for a database with even more modest memory requirements than Adaptive Server Anywhere. An obvious option is to produce a trimmed down relational database engine, but our experience has shown that each application and each customer has a distinct set of features that are, for them, essential. Further, such an approach would mean that customers would have to learn two different databases, inevitably different in some ways.

Instead, we developed UltraLite, a novel technology that uses a reference database and your application source code to generate a relational database engine containing only those features of Adaptive Server Anywhere used by your application. Each query is stored with a complete access plan for fast execution; the code needed to execute just those tasks you need are built into your UltraLite database engine. Each UltraLite database engine is different, but many are only a few tens of kilobytes, and can easily be run in a device as small as a pager.

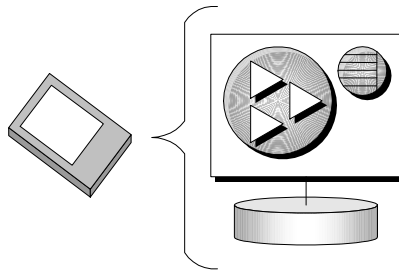
Adaptive Server Anywhere serves as a reference database when you build your UltraLite application, and so your SQL statements, data types, and so on are exactly those of Adaptive Server Anywhere. UltraLite is a deployment technology for Adaptive Server Anywhere, not a different database system. The tasks that each UltraLite database engine is built to perform are carried out in a manner completely compatible with Adaptive Server Anywhere.

UltraLite has built-in MobiLink synchronization technology so that your application is linked into the information network.

UltraLite supports the Palm Computing Platform, Windows CE, and other operating systems used in small devices, such as Java.

### UltraLite architecture

If you want to provide a database application for a small device such as a handheld organizer, you may want to use the UltraLite deployment technology. In UltraLite, the database server and the application are part of the same process, and the database server is specific to the application.



In this case, the database may not be a file on disk. The storage method for the database depends on the deployment platform.

---

# Comparing Adaptive Server Anywhere and UltraLite

This section highlights the differences between Adaptive Server Anywhere and UltraLite, to help you decide which technology is suited to your needs.

If you know what your resources are already, choosing between Adaptive Server Anywhere and UltraLite may be an easy decision. If, however, you are planning to deploy on a platform supported by both technologies, such as Windows CE, consider the following issues.

## Adaptive Server Anywhere

For many years, Adaptive Server Anywhere has provided relational database technology designed specifically for the needs of workgroup, mobile and embedded computing. The product has been designed from the ground up with this market in mind. The advantages of choosing Adaptive Server Anywhere include:

- ◆ Adaptive Server Anywhere is designed to operate efficiently with limited memory, CPU power, and disk space. Core features such as the query optimizer and the data caching mechanism are designed specifically to operate without extravagant use of resources. At the same time, Adaptive Server Anywhere contains the features needed to take advantage of workgroup servers, including support for many users, scalability over multiple CPUs, and advanced concurrency features. If you are deploying mainly to PCs, Adaptive Server Anywhere is built to meet your needs.
- ◆ Adaptive Server Anywhere is a cross-platform solution. The same database runs on Windows (Windows 95 and its successors, Windows NT and its successors, and Windows CE), UNIX including Linux, and Novell NetWare. You can move a database file from one operating system to another.
- ◆ Adaptive Server Anywhere is designed to operate without administration, making it ideal for use as an embedded database. Adaptive Server Anywhere provides a self-tuning query optimizer, built-in scheduling and event-handling capabilities, as well as autostart and autostop mechanisms.
- ◆ Many years of experience working with successful customers have led to a rich set of field-tested features. Not only the standard checklist features of stored procedures, triggers, declarative referential integrity, full transaction processing, and recovery, but all the little extras that can make the difference between a successful project and a failure.

- ◆ SQL Anywhere synchronization technologies (SQL Remote and MobiLink) mean that you can integrate Adaptive Server Anywhere databases into your organization's infrastructure.
- ◆ Adaptive Server Anywhere can be a benefit if you are happier using an interface other than Embedded SQL or Java (such as ODBC or OLE DB) and your target platform is not so resource-constrained as to require UltraLite.

## UltraLite: the “small fingerprint” database

UltraLite is a novel technology that uses a reference database and your application source code to generate a relational database engine containing only those features of Adaptive Server Anywhere used by your application. The advantages of choosing UltraLite include:

- ◆ UltraLite database engines include only the code necessary to execute the tasks specified when your application is compiled, so you cannot use dynamic SQL to execute ad hoc queries against an UltraLite database.  
Each UltraLite database and database engine is for use by only a single application: if you want to use more than one application against a single database, you should choose Adaptive Server Anywhere.
- ◆ The programming model for UltraLite enables platform-independent database access code so that you can port the user interface to new platforms and devices without having to alter the underlying data access layer.
- ◆ If you are deploying to small devices such as the Palm Computing Platform, UltraLite will fit the bill.





---

## CHAPTER 3

# Introducing Replication Technologies

### About this chapter

Data **replication** is the sharing of data among physically distinct databases. When an application modifies shared data at any one database, the changes are propagated to the other databases in the replication setup. Changes can be propagated by various means and through a variety of channels, allowing flexible replication setups while preserving data integrity. Data replication is also referred to as data **synchronization**.

Sybase has three replication technologies. **MobiLink** and **SQL Remote** are designed for replication between a central database and a large number of remote databases. **Replication Server** is intended for near-real-time replication between a relatively small number of databases.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	22
<a href="#">Consolidated and remote databases</a>	23
<a href="#">Propagation methods</a>	26
<a href="#">Sybase replication technologies</a>	29

---

# Introduction

This section introduces basic concepts in data replication.

## Benefits of data replication

- Data availability** One of the key benefits of a data replication system is that data is made available locally, rather than through potentially expensive, less reliable, and slow connections to a single central database. Data is accessible locally even in the absence of any connection to a central server, so you are not cut off from data in the event of a failure of a long-distance network connection.
- Response time** Replication improves response times for data requests for two reasons. Retrieval rates are faster because requests are processed on a local server, without accessing a wide area network. Also, local processing offloads work from a central database server so that competition for processor time is decreased.

## Challenges for replication technologies

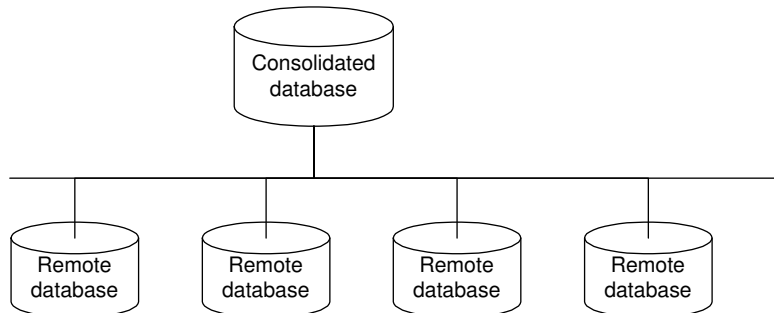
- Any replication technology must address several challenges that arise as a result of the increased flexibility permitted by replication.
- Transactional integrity** One of the challenges of any replication system is to ensure that each database retains transactional integrity at all times.
- Replication Server and SQL Remote replicate portions of the transaction log in such a way that transactions are maintained during replication: either a whole transaction is replicated, or none of it is replicated. This ensures transactional integrity at each database in the setup.
- MobiLink consolidates changes made in multiple, committed transactions. These changes are applied to another database in a single transaction.
- Data consistency** Another challenge to replication systems is to maintain data consistency throughout the setup. Replication systems maintain a **loose consistency** in the setup as a whole: that is, all changes are replicated to each site over time in a consistent manner, but different sites may have different copies of data at any instant.

## Consolidated and remote databases

Both MobiLink and SQL Remote provide data replication between a consolidated database and a set of remote databases.

A **consolidated database** is a database that contains all the data to be replicated. A **remote database** is a copy of the consolidated database that may be running either at the same site as the consolidated database or at a physically distant site.

The figure displays a schematic illustration of a small installation.



Remote users

A replication installation includes many remote databases. Each remote database contains a subset of the information in the consolidated database. Each remote database is a physically separate database, usually on a separate computer. All remote databases must stay consistent with the consolidated database.

The entire replication setup may be considered a single dispersed database, with the master copy of all shared data being kept at the consolidated database.

Each remote site that submits replications to the consolidated database is considered to be a **remote user** of the consolidated database. In the case that a remote site is a multi-user server, the entire site is considered to be a single remote user of the consolidated database.

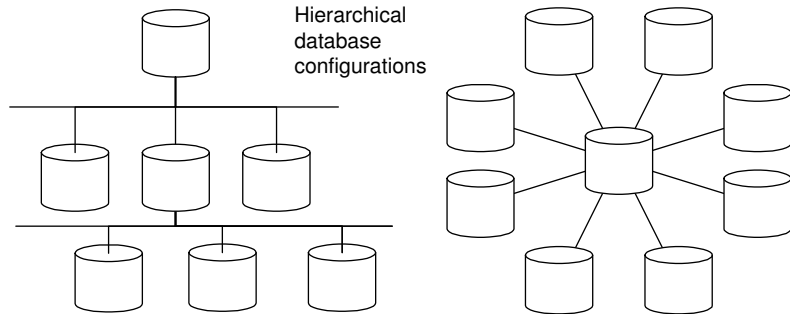
## Hierarchical database configurations

For databases in a **hierarchical configuration**, every database has a single parent database, except the consolidated database, which has no parent.

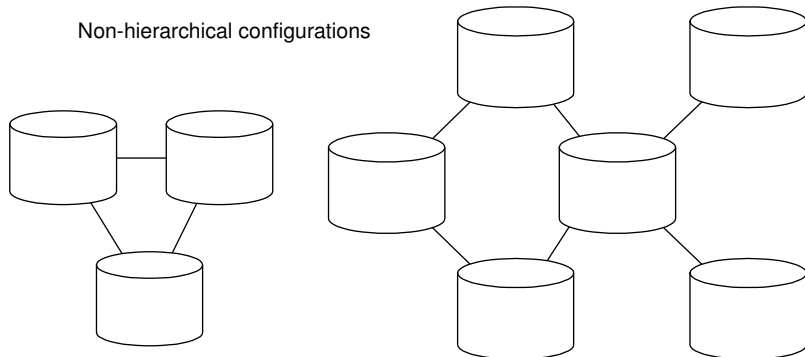
SQL Remote supports hierarchical configurations of databases; it does not support peer-to-peer replication or other non-hierarchical configurations. MobiLink is also normally used with a hierarchical configuration, but can also be used in other configurations.

---

For any two databases directly sharing data in a hierarchical configuration, one is always above or below the other in the hierarchy.



Databases in a non-hierarchical configuration do not have a well-defined notion of above or below.



In a MobiLink or SQL Remote installation, each database contains all of, or a subset of, the data replicated by the database above it in the hierarchy.

Remote databases can contain tables that are not present at the consolidated database as long as they are not involved in replication. SQL Remote requires that the table and column names in the remote databases match the ones in the consolidated database. In contrast, MobiLink allows data to be stored in different columns and tables in the remote databases than in the consolidated database, allowing greater flexibility.

## Two-way replication

All Sybase replication technologies provide two-way replication: changes made at the consolidated database are propagated to remote databases and changes made at remote databases are propagated to the consolidated database and, hence, to other remote databases. Sybase Replication Server requires that a particular piece of data can be modified at only one location.

Both SQL Remote and MobiLink allow the same data to be changed simultaneously at multiple locations and provide a means of resolving any conflicts.

---

## Propagation methods

When a transaction modifies shared data at any one database, the transaction or changes must be replicated to the other databases in the replication setup. There are various means by which this task may be accomplished.

### Session-based replication: MobiLink

In a session-based replication scheme, synchronization occurs in real time over some sort of direct communications link. For example, the connection could be over a modem, network, or radio modem. Remote sites connect at intervals of minutes, hours, days, or weeks.

A session-based synchronization process is analogous to a telephone conversation in which all outstanding issues at both ends are resolved. The process follows a particular format. A MobiLink remote site begins by opening a connection to a MobiLink synchronization server and uploading a complete list of all the changes made to the remote database since the previous synchronization. Upon receiving this data, the server updates the consolidated database, then sends back all relevant changes. The remote site incorporates the entire set of changes, then sends back a confirmation and closes the connection.

### Message-based replication: SQL Remote

SQL Remote exchanges data between databases using **messages**. Messages are typically files, placed in a particular directory, or specially formatted e-mail messages. A **message agent**, attached to each database, sends messages regarding changes to its own data. The same agent also receives messages from one or more other databases and modifies the database, according to the contents of the received messages. This system allows replication between databases that have no direct connection: an occasional message-based connection such as e-mail or a periodic dial-up link is sufficient.

In message-based communications, each message carries its destination address and other control information so that no direct connection is necessary between applications exchanging information. For example, an e-mail message contains the destination address; there is no direct connection between the sending server and the recipient.

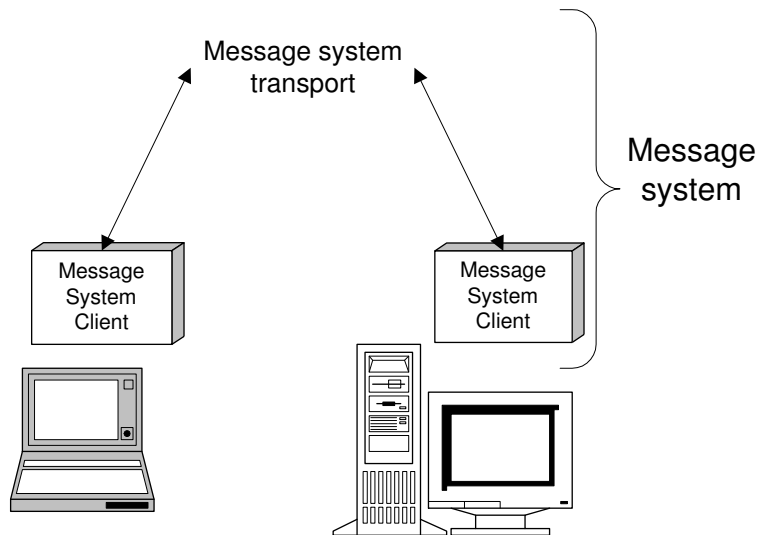
Message services use store and forward methods

Just as session-based client/server applications rely on network communication protocol stacks, such as TCP/IP or Novell NetWare's SPX, so message-based applications rely on message services such as Internet Simple Mail Transfer Protocol (SMTP), Microsoft's Messaging API

(MAPI), Lotus' Vendor Independent Messaging (VIM), or a simple shared file link.

Message services use **store-and-forward** methods to get each message to its destination: for example, e-mail systems store messages until the recipient opens their mail folder to read their mail, at which time the e-mail system forwards the message.

Building a replication system on top of a message system means that a message-based replication system, such as SQL Remote, does not need to implement a store-and-forward system to get messages to their destination. Just as session-based client/server applications do not implement their own protocol stacks to pass information between client and server, so SQL Remote uses existing message systems to pass the messages.



#### Guaranteed delivery

To work reliably, a message-based replication system must both guarantee that all messages reach their destination and that the messages are applied in the same order that they are sent. SQL Remote incorporates a protocol to guarantee application of replication updates in the correct order.

### Connection-based replication: Replication Server

Some replication technologies rely on the presence of a continuous, or at least almost continuous, connection between the databases. Through this connection, the two databases conduct an ongoing dialogue. These types of systems excel at replicating changes quickly. Indeed, given sufficient

---

resources and channel capacity, replication can occur reliably with a lag time of no more than a few seconds.

Replication server is a near real-time replication system designed primarily for replication between a small number of databases. It is normally used with a continuous, reliable, high-speed connection. It incorporates store-and-forward techniques that allow replication to continue automatically when a connection is lost and later re-established.

The main drawback of this type of system is that a reliable, continuous connection can be expensive to maintain. This restriction makes connection-based technologies suited to replication between two large, fixed databases. In environments where the remote machines are mobile or are only occasionally connected, message-based or session-based technologies provide more flexible solutions.



## Sybase replication technologies

Sybase provides three replication technologies:

- ◆ **MobiLink** is a session-based technology intended for the two-way replication of data between a central, consolidated database and a large number of remote databases. It supports a variety of consolidated database servers, including non-Sybase databases. Administration and resource requirements at the remote sites are minimal, making it well suited to a variety of mobile applications. At the end of each synchronization session, the databases are consistent.
- ◆ **SQL Remote** is a message-based technology intended for the two-way replication of transactions. It is designed for two-way replication involving a consolidated data server and large number of remote databases. Administration and resource requirements at the remote sites are minimal, making it well suited to mobile databases. This system is message based. Depending on the setup, typical lag times between the consolidated and remote databases can be on the order of seconds, minutes, or hours.
- ◆ **Replication Server** is a connection-based technology intended for the two-way replication of transactions. It is well suited to replication between a small number of enterprise databases connected by a high-speed network, generally with an administrator at each site. In such a setup, it is possible to achieve lag times as low as a few seconds.

### Choosing a replication technology

Each Sybase replication technology lends itself to particular applications. The following descriptions differentiate the technologies and let you select the one best suited to your needs.

You should consider which of the following elements are important in your application

Your consolidated database system

In a typical replication environment, a large database serves as a central repository for information. Sometimes you can choose a database system that suits your needs. Other times, a central database already exists and you must adapt the replication system to work with it.

MobiLink can work with many popular database servers, including Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, Oracle, Microsoft SQL Server, and IBM DB2.

In a SQL Remote installation, the central database must be either Adaptive Server Anywhere or Sybase Adaptive Server Enterprise.

---

Your remote database system	<p>Sybase replication technologies also differ in the types of remote databases that they can support. MobiLink allows your remote database to be either Adaptive Server Anywhere or UltraLite.</p> <p>SQL Remote supports only Adaptive Server Anywhere remote databases.</p>
Network characteristics	<p>MobiLink and SQL Remote are both well suited to occasionally-connected environments, where remote sites must operate for hours or days in isolation, although more frequent synchronization is possible whenever a network connection is available. In contrast, Replication Server is designed for a continuous connection to allow large amounts of data to be replicated promptly.</p> <p>MobiLink is session based. A real-time connection is required during synchronization. If this connection is interrupted before synchronization is complete, the process will not complete until the next synchronization. In contrast, SQL Remote relays information via messages, which can be sent or received asynchronously. These messages may take the form of files on a hard disk, or e-mail messages. These messages can be processed whenever they are received, allowing replication to occur incrementally.</p>
Frequency of synchronization	<p>In some situations, it may be important that your information is replicated immediately. In others, replication once or twice a day may suffice. In fact, more frequent replication may be impossible when no network connection is available.</p> <p>Both MobiLink and SQL Remote are primarily intended for situations where replication occurs infrequently; for example every few hours or days. MobiLink and SQL Remote can both handle more frequent synchronization, but resource and network requirements are greater. However, given sufficient resources, MobiLink synchronizations can occur every few minutes. SQL Remote, when run in continuous mode, allows replication to occur every few seconds.</p> <p>Replication Server is designed for setups requiring near real-time replication.</p>
The number of remote sites	<p>If you have a very large number of remote users, the best options are MobiLink or SQL Remote. The SQL Remote message-based design allows a typical installation to handle thousands of remote users. MobiLink scalability is limited only by the scalability of the consolidated database-management system. Replication Server is designed for only a few sites.</p> <p>While these numbers are guidelines, there is no hard limit on the maximum number of remote sites with any of these systems. The actual number depends on the amount of information replicated, the frequency of synchronization, and the design of your application.</p>

**Transaction ordering** SQL Remote replicates data by scanning the transaction log and preparing messages, as appropriate, for each transaction. It orders these messages and sends them to the remote or consolidated site. When processing receives messages, SQL Remote always processes them in the same order as they were applied to the other database. When necessary, it automatically delays processing a message until all earlier messages have been applied.

MobiLink, in contrast, works by grouping the results of multiple transactions on the remote database into one set of changes to be applied to the consolidated database. Since synchronization always occurs at a transaction boundary, referential integrity is preserved. The order of the individual changes made during the component transactions is not preserved. However, since uncommitted data is never synchronized, data integrity is preserved.

**Achieving data consistency at a particular time** Immediately following each MobiLink synchronization session, the data in the two databases is consistent. The ability to guarantee the consistency of the data at a remote site at a particular point in time is an advantage of MobiLink session-based replication. For example, if it is important that the data at a remote site accurately reflect the data in the consolidated database at a particular time, say 10 o'clock in the morning, this objective can be achieved by synchronizing just prior to this time. As long as the synchronization completes successfully, the currency of the data at the remote site is assured.

When changes to the data are replicated through an exchange of messages, it is difficult to guarantee that the data in a particular remote site is completely consistent with the data in the consolidated site at any particular point in time. For example, sometimes a message is lost in transit. SQL Remote automatically recognizes this fault and resends the message, but such interruptions can cause unexpected delays.

### Replication technology characteristics summary

The following table summarizes the characteristics of each replication technology. Following sections expand on the entries in this table.

Replication technology	Number of databases	Connection	Frequency	Volume	Database types
MobiLink	Large	Occasional	Medium	Medium	Heterogeneous
SQL Remote	Large	Occasional	Low	Medium	Homogeneous
Replication Server	Small	Continuous	Low	High	Heterogeneous

---

## MobiLink characteristics

MobiLink is designed for replication installations with the following requirements:

- ◆ **Large numbers of databases** MobiLink is designed to support large numbers of remote databases. It can support thousands of remote databases in a single installation.
- ◆ **Occasionally connected** MobiLink supports databases that are occasionally connected or indirectly connected to the network on which the server is running. MobiLink scalability is limited only by the scalability of the consolidated database-management system.
- ◆ **Flexible synchronization schedule** Applications typically connect and synchronize at intervals of minutes, hours, or days.
- ◆ **Low to medium volume** Download information for remote sites is prepared for one remote site at a time. Large amounts of data in a MobiLink system can cause long connection times, since the remote site cannot disconnect until synchronization is complete.
- ◆ **Heterogeneous databases** MobiLink supports many of the most popular relational-database systems for use as a consolidated database. The schema of the remote sites can be different from that of the consolidated database because you control the synchronization process by writing scripts.

## SQL Remote characteristics

SQL Remote is designed for replication installations with the following requirements:

- ◆ **Large numbers of databases** SQL Remote is designed to support a large number of remote databases. It can support thousands of remote databases in a single installation because the messages for many remote sites can be prepared simultaneously.
- ◆ **Occasionally connected** SQL Remote supports databases that are occasionally connected or indirectly connected to the network on which the server is running.
- ◆ **Low to high latency** High latency means a long lag time between data being entered at one database and being replicated to each database in the installation. With SQL Remote, replication messages are sent typically at periods of seconds, minutes, hours, or days.

- ◆ **Low to moderate volume** As replication messages are delivered occasionally, a high transaction volume at each remote site can lead to a very large volume of messages. SQL Remote is best suited to systems with a relatively low volume of replicated data per remote database. At the consolidated site, SQL Remote can, however, prepare messages efficiently by preparing messages for multiple sites simultaneously.
- ◆ **Homogeneous databases** SQL Remote supports Adaptive Server Enterprise and Adaptive Server Anywhere databases. Each database in the system must have a very similar schema.

### Replication Server characteristics

Replication Server is designed for replication installations with the following requirements:

- ◆ **Small numbers of databases** Replication Server is designed to support replication among servers, with installations typically involving fewer than one hundred servers.
- ◆ **Continuously connected** Connections between primary sites and replicate sites may be over a wide area network, but Replication Server is designed for situations where there is a near-continuous connection path for data exchange among the servers in the installation.
- ◆ **Low latency** Low latency means a short lag time between data being entered at one database and being replicated to each database in the installation. With Replication Server, replication messages are sent typically within seconds of being entered at a primary site.
- ◆ **High volume** With near-continuous connections and high performance, Replication Server is designed for a high volume of replication messages.
- ◆ **Heterogeneous databases** Replication Server supports several leading DBMSs, and allows mapping of object names during replication, so that support for heterogeneous databases is provided.



## PART II

# PLATFORM SUPPORT

This part introduces SQL Anywhere Studio supported platforms, as well as platform-specific information.





---

## CHAPTER 4

# SQL Anywhere Studio for Windows CE

### About this chapter

This chapter includes Windows CE platform-specific information for SQL Anywhere Studio.

The information contained within this chapter is intended to supplement information contained throughout the remainder of the documentation set.

☞ For information about security and databases running on Windows CE, see “[Keeping your Windows CE database secure](#)” [*SQL Anywhere Studio Security Guide*, page 23].

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Getting started</a>	38
<a href="#">Using the sample applications</a>	45
<a href="#">Configuring Windows CE databases</a>	53
<a href="#">Using the database server on Windows CE</a>	62
<a href="#">Using the administration utilities</a>	69
<a href="#">SQL Anywhere Studio features not supported on Windows CE</a>	80

---

## Getting started

This section describes the tasks required to set up SQL Anywhere Studio on your Windows CE device. Completion of these tasks ensures that you can use the features of SQL Anywhere Studio for Windows CE.

### Installing SQL Anywhere Studio on a Windows CE device

#### Requirements

- ◆ SQL Anywhere Studio for Windows CE must be installed from a machine running a supported Microsoft Windows operating system.  
☞ For a list of Microsoft Windows operating systems supported by Adaptive Server Anywhere, see [“Operating system versions” on page 111](#).
- ◆ ActiveSync for your Windows CE device must be installed on the desktop machine.

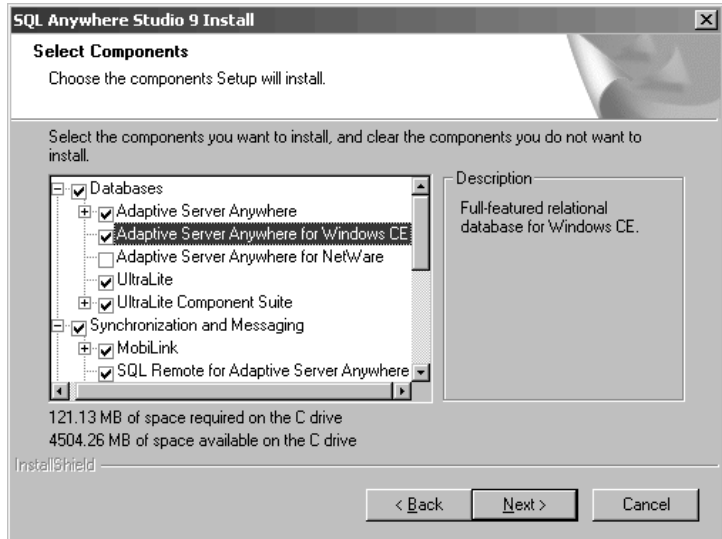
#### Windows CE file locations

On Windows CE, all files are installed to the installation directory `\Program Files\Sybase\ASA9`. No subdirectories are created. The only exception is that all DLLs are installed into the `\Windows` directory.

#### ❖ To install SQL Anywhere Studio for Windows CE

1. Connect your Windows CE device to a PC running a supported Microsoft Windows operating system.
2. Start the installation by running the SQL Anywhere Studio `setup.exe` program on your PC.
3. Follow the instructions in the SQL Anywhere Studio 9 Install wizard.
4. On the Select Components page, select the Adaptive Server Anywhere for Windows CE checkbox under Databases.

Optionally, if you already have Adaptive Server Anywhere installed on your PC, clear the Adaptive Server Anywhere checkbox under Databases to avoid reinstalling to your PC.



5. Follow the remaining instructions in the wizard.

## Configuring a PC to connect to your Windows CE device

This section describes two methods to configure a PC to connect to an Adaptive Server Anywhere database running on your Windows CE device. This is done to allow you to administer your Windows CE database from the administration utilities on a PC.

☞ For more information on Windows CE administration, see [“Using the administration utilities”](#) on page 69.

### Requirements

- ◆ Microsoft ActiveSync 3.5 or higher.
- ◆ A Windows CE device supported by Adaptive Server Anywhere.
  - ☞ For a list of Windows CE devices supported by Adaptive Server Anywhere, see [“Operating system versions”](#) on page 111.
- ◆ Adaptive Server Anywhere for Windows CE installed on your Windows CE device.
- ◆ A PC running a supported Microsoft Windows operating system.

### Determining the IP address of your Windows CE device

You can connect from applications running on a desktop machine, such as Sybase Central or Interactive SQL, to a database server running on a

---

Windows CE device. The connection uses TCP/IP over the ActiveSync link between the desktop machine and the Windows CE device.

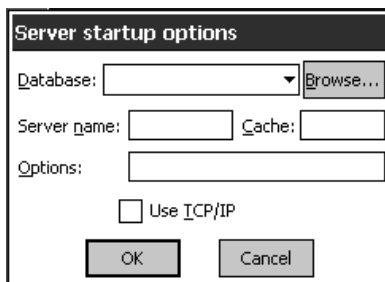
If you are using an Ethernet connection between your desktop machine and the Windows CE device the following procedure works. Connecting from the desktop to the Windows CE device is not supported if ActiveSync is using a USB connection.

☞ For information about using a serial cable connection and ActiveSync 3.x, see [“Using ActiveSync 3.0 and a serial cable” \[ASA Database Administration Guide, page 60\]](#).

### ❖ To determine the IP address of your Windows CE device

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. Tap the *dsrv9* icon.

The Server Startup Options dialog appears.



4. Specify the database file that you want to start.  
Type **/Program Files/Sybase/ASA9/asademo.db** to start the sample database running on the device.
5. Select the Use TCP/IP checkbox.  
A TCP/IP connection is necessary to connect from a desktop machine to the database running on your Windows CE device.
6. In the Options field of the Server Startup Options dialog, type **-z**.  
With the **-z** option, the server writes out its IP address during startup. The address may change if you disconnect your Windows CE device from the network and then re-connect it.

☞ For more information, see [“-z server option” \[ASA Database Administration Guide, page 166\]](#).

7. Tap OK to start the sample database running on the network server.
8. Navigate to the Today screen on your device.
9. Tap the Server icon located in the bottom right corner of the screen.  
The Server Messages window appears.
10. Determine the IP address of the server.  
The IP address appears in the Server Messages window. To see the IP address, you may have to scroll up using the scroll bar on the side of the Server Messages window.

Now you can create an ODBC data source to connect from the desktop machine to your Windows CE device.

☞ For more information, see [“Creating an ODBC data source to connect to your Windows CE device”](#) on page 42.

## Editing the registry

### **Caution**

*Modifying your registry is dangerous. Modify your registry at your own risk. It is recommended that you back up your system before modifying the registry.*

### ❖ To configure your desktop machine

1. Stop any Adaptive Server Anywhere database servers running on both the desktop machine and your Windows CE device.
2. Remove your Windows CE device from its cradle.
3. Open the Registry Editor on the desktop machine.  
At a command prompt, type **regedit**.
4. In the Registry Editor, open the following key:  
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows CE Services\ProxyPorts`.
5. From the Edit menu, select New ► DWORD Value.  
Enter the name **ASA**.
6. Double-click the ASA key that you just created.

- 
7. Give it the Decimal value 2638.

This is the value of the default TCP/IP port for Adaptive Server Anywhere.

**Note**

Every time your Windows CE device is cradled, ActiveSync forwards traffic on port 2638 to the device. Thus, if you start a server on your desktop machine while your Windows CE device is cradled, it cannot use the default port 2638. If this is problematic, you can choose another port to dedicate to Windows CE traffic.

8. Click OK.

Now you can create an ODBC data source to connect from the desktop machine to your Windows CE device.

☞ For more information, see [“Creating an ODBC data source to connect to your Windows CE device”](#) on page 42.

## Creating an ODBC data source to connect to your Windows CE device

The **Open Database Connectivity (ODBC)** interface is defined by Microsoft Corporation, and is a standard interface for connecting client applications to database-management systems in Windows environments. Connections are made by specifying connection parameters. It is often convenient to collect a set of connection parameters together and store them in an **ODBC data source**. ODBC data sources are a convenient way of saving connection parameters for repeated use.

☞ For more information, see [“Working with ODBC data sources”](#) [*ASA Database Administration Guide*, page 53].

This section describes how to create an ODBC Data Source on your Windows desktop machine to connect to a database running on your Windows CE device.

### ❖ To create an ODBC data source to connect to your Windows CE device

1. Open the ODBC Administrator on the desktop machine.

From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.

The ODBC Administrator appears.

2. On the User DSN tab, click Add.

The Create New Data Source dialog appears.

3. Select Adaptive Server Anywhere 9.0, and then click Finish.

The ODBC Configuration dialog appears.

4. On the ODBC tab, in the Data Source Name field, type a name for the data source.

For example, type **CEdevice**.

5. On the Login tab, select Supply User ID and Password and leave the User ID and Password fields blank.

Each time you connect to a database, you need to supply a user ID and password.

**Tip**

When connecting to a database, the default user ID is **DBA**. The default password is **SQL**.

6. On the Database tab, leave the Server Name field blank.

Each time you connect from a desktop machine, you will have to specify the server name. This name appears in the title bar of the Server Messages window on your Windows CE device.

7. On the Network tab, select the TCP/IP checkbox.

8. In the adjacent field, type the connections parameters.

For example, type **Host=127.0.0.1;DoBroadcast=none**.

**Host** this parameter specifies the IP address that the Windows CE device listens on.

If you had to edit the registry to connect to your Windows CE device, use the default IP address of **127.0.0.1**.

☞ For more information, see [“Editing the registry” on page 41](#).

Otherwise, use the IP address of your Windows CE device.

☞ For more information, see [“Determining the IP address of your Windows CE device” on page 39](#).

☞ For more information, see [“Using the TCP/IP protocol” \[ASA Database Administration Guide, page 87\]](#).

**DoBroadcast** this parameter controls how the TCP/IP connection is made.

When **DoBroadcast=none** is specified, the TCP/IP connection is made directly with the port specified. Use this setting if you had to edit the registry to connect to your Windows CE device.

☞ For more information, see [“Editing the registry” on page 41](#).

---

When **DoBroadcast=direct** is specified, no broadcast is performed to the local subnet to search for a database server. Instead, the Host IP address is required.

☞ For more information, see [“Determining the IP address of your Windows CE device” on page 39](#).

☞ For more information, see [“DoBroadcast protocol option \[DOBROAD\]” \[ASA Database Administration Guide, page 211\]](#).

9. Click OK to create the data source.

You can now use the data source you just created to connect from a desktop machine to a database running on your Windows CE device.

☞ For more information, see [“Using the administration utilities” on page 69](#).



## Using the sample applications

This section introduces you to the sample database and the sample applications provided with SQL Anywhere Studio for Windows CE.

### The sample database

The sample database represents a small company that makes a limited range of sports clothing. All information in the sample database is fictional.

☞ For more information, see [“About the sample database” on page 198](#).

Where is the sample database stored?

The sample database is stored in a file called *asademo.db*. This file is located in the `|Program Files|Sybase|ASA9` directory on your Windows CE device.

How do I access the sample database?

The following sample applications are included with your SQL Anywhere Studio for Windows CE installation:

- ◆ ASA Server Example
- ◆ ADO.NET Sample
- ◆ ADOCE Sample
- ◆ ESQL Sample
- ◆ ODBC Sample

You can use these applications to access the sample database and examine the capabilities of SQL Anywhere Studio for Windows CE.

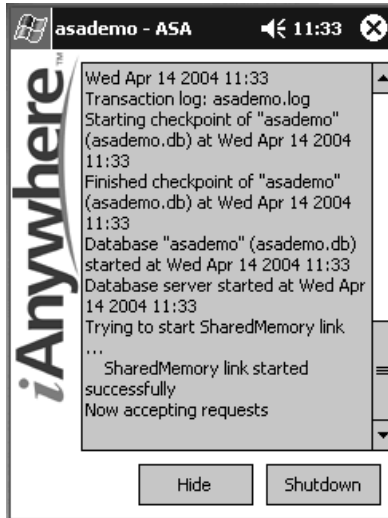
### The ASA Server Example

The ASA Server Example starts the sample database on a network server using preset server options and connection parameters.

#### ❖ To start the ASA Server Example

1. Navigate to the Adaptive Server Anywhere installation directory.  
From the Start menu, tap Programs ► Sybase ► ASA9.
2. Start the sample database on a network server by tapping the ASA Server Example icon.

The sample database starts on the network server. Once it is started, the server appears as an icon in the bottom right-hand corner of the Today screen on your Windows CE device. If you would like to view the Server Messages window, tap the icon.



The sample database is running on your Windows CE device.

You can now connect to the sample database on your Windows CE device from a desktop machine.

☞ For more information, see [“Using the administration utilities” on page 69](#).

When you are finished using the sample database, you must shut down the server.

#### ❖ To shut down the server

1. Tap the network server icon located in the bottom right-hand corner of the Today screen.

The Server Messages window appears.

2. Tap Shutdown.

## The ADO.NET Sample

In order to use the ADO.NET Sample, you must have the Microsoft.NET Compact Framework version 1.0 with Service Pack 2 or higher installed on your device.

You can download this component from the [Microsoft Download Center](#).

The ADO.NET Sample demonstrates a simple application that uses the ADO.NET programming interface. This application allows you to start the

sample database running on the network server, and access and modify data using SQL statements.

The source code for this sample can be found in  
*asa\_install\_directory\samples\asa\ce\_ado\_net\_sample*.

You can load this project in eMbedded Visual C++ from:  
*asa\_install\_directory\samples\asa\ce\_ado\_net\_sample*.

☞ For more information, see “[The ADO.NET programming interface](#)” [*ASA Programming Guide*, page 3].

**Note**

In the ADO.NET Sample user interface, SQL statements must be entered on a single line.

❖ **To use the ADO.NET Sample**

1. Start the ADO.NET Sample.

From the Start menu, tap Programs ► Sybase ► ASA9 ► ADO.NET Sample.

2. Tap Connect.

The sample database starts on a network server. The Server Messages window appears briefly, then disappears.

3. Tap Exec SQL to execute the default SQL statement, `SELECT * FROM employee`, and then tap Exec SQL.

All of the data from the Employee table appears in the data window.

4. Navigate through the data in the employee table using the scroll bars on the side and bottom of the data window.

5. Access a more specific range of data.

Type `SELECT emp_id, emp_lname FROM employee`.

Tap Exec SQL to execute the SQL statement.

6. The specified range of data replaces the data that was in the data pane.

7. Edit data in the table.

Type `SELECT * FROM employee ORDER BY emp_id` and tap Exec SQL.

Notice the employee Matthew Cobb, with the `emp_id` of 105.

8. Type `UPDATE employee SET emp_lname = 'Jones' WHERE emp_lname = 'Cobb'`.

- 
9. Tap Exec SQL to execute the SQL statement.
  10. Type `SELECT * FROM employee ORDER BY emp_id` and tap Exec SQL.  
Notice that Matthew's last name has been changed from Cobb to Jones.
  11. Reverse the changes.  
Type `UPDATE employee SET emp_lname = 'Cobb' WHERE emp_lname = 'Jones'` and tap Exec SQL.
  12. Verify that the changes were reversed.  
Type `SELECT * FROM employee ORDER BY emp_id` and tap Exec SQL.  
Notice that Matthew's last name has been changed back to Cobb.
  13. Access data from another table.  
In the Exec SQL field, type `SELECT * FROM customer`.  
Tap Exec SQL.
  14. All of the data from the Customer table appears in the data window, replacing the data from the Employee table.
  15. Shut down the network server by tapping Disconnect.  
The ADO.NET Sample disconnects, and the network server automatically shuts down.
  16. Close the ADO.NET Sample by tapping the × in the top right-hand corner of the window.

## The ADOCE Sample

The ADOCE Sample demonstrates a simple application that uses ADOCE, the ADO programming interface for Windows CE development. This application allows you to start the sample database running on the network server, and access data using SQL statements.

The source code for this sample can be found in `asa_install_directory\samples\asa\ce_adoce_sample`.

You can load this project in eMbedded Visual C++ from `asa_install_directory\samples\asa\ce_adoce_sample`.

☞ For more information on the ADO programming interface, see “[The OLE DB and ADO Programming Interfaces](#)” [*ASA Programming Guide*, page 327].

**Note**

In the ADOCE Sample user interface, SQL statements must be entered on a single line.

❖ **To use the ADOCE Sample**

1. Start the ADOCE Sample.  
From the Start menu, tap Programs ► Sybase ► ASA9 ► ADOCE Sample.
2. Tap Connect.  
The sample database starts on a network server. The Server Messages window appears briefly, then disappears.
3. Tap Execute SQL to execute the default SQL statement, `SELECT * FROM employee.`  
All of the data from the Employee table appears in the data window.
4. Navigate through the data in the employee table using the scroll bar on the side of the data window.
5. Access a more specific range of data.  
Type `SELECT emp_id, emp_lname FROM employee.`  
Tap Execute SQL to execute the SQL statement.
6. Scroll down to view the data, which appears below the data that is already displayed in the data window.
7. Access data from another table.  
In the Execute SQL field, type `SELECT * FROM customer.`  
Tap Execute SQL.
8. Scroll down to view the data from the Customer table, which appears below the data that is already displayed in the data window.
9. Shut down the network server by tapping Disconnect.  
The ADOCE Sample disconnects, and the network server automatically shuts down.
10. Close the ADOCE Sample by tapping the × in the top right-hand corner of the window.

---

## The ESQL Sample

The ESQL Sample demonstrates a simple application that uses the Embedded SQL programming interface. This application allows you to start the sample database running on the network server, and access data using SQL statements.

The source code for this sample can be found in `asa_install_directory\samples\asa\ce_esql_sample`.

You can load this project file in eMbedded Visual C++ from: `asa_install_directory\samples\asa\ce_esql_sample\esql_sample.vcw`.

☞ For more information, see “[The Embedded SQL programming interface](#)” [*ASA Programming Guide*, page 5].

### Note

In the ESQL Sample user interface, SQL statements must be entered on a single line.

### ❖ To use the ESQL Sample

1. Start the ESQL Sample.  
From the Start menu, tap Programs ► Sybase ► ASA9 ► ESQL Sample.
2. Tap Connect to connect to the sample database using the default connection string.  
The sample database starts on a network server. The Server Messages window appears briefly, then disappears.
3. Tap ExecSQL to execute the default SQL statement, `SELECT * FROM employee`.  
All of the data from the Employee table appears in the data window.
4. Navigate through the data in the employee table using the scroll bars on the bottom and side of the data window.
5. Access data in another table.  
In the ExecSQL field, type `SELECT * FROM customer`, and then tap ExecSQL.  
All of the data from the Customer table appears in the data window, replacing the data from the Employee table.
6. Shut down the network server by tapping Disconnect.

The ESQL Sample disconnects and the network server automatically shuts down.

7. Close the ESQL Sample by tapping the × in the top right-hand corner of the window.

## The ODBC Sample

The ODBC Sample demonstrates a simple application that uses the ODBC programming interface. This application allows you to start the sample database running on the network server, and access data using basic SQL statements.

The source code for this sample can be found in  
*asa\_install\_directory\samples\asa\ce\_odbc\_sample*.

You can load this project file in eMbedded Visual C++ from:  
*asa\_install\_directory\samples\asa\ce\_odbc\_sample\odbc\_sample.vcw*

☞ For more information, see “[The ODBC programming interface](#)” [*ASA Programming Guide*, page 2].

### Note

In the ODBC Sample user interface, SQL statements must be entered on a single line.

### ❖ To use the ODBC Sample

1. Start the ODBC Sample.  
 From the Start menu, tap Programs ► Sybase ► ASA9 ► ODBC Sample.
2. Tap Connect.  
 The sample database starts on a network server. The Server Messages window appears briefly, then disappears.
3. Tap ExecSQL to execute the default SQL statement, `SELECT * FROM employee`.  
 All of the data from the Employee table appears in the data window.
4. Navigate through the data in the employee table using the scroll bars on the bottom and side of the data window.
5. Access data in another table.  
 In the ExecSQL field, type `SELECT * FROM customer`.  
 All of the data from the Customer table appears in the data window, replacing the data from the Employee table.

---

6. Shut down the network server by tapping Disconnect.

The ODBC Sample disconnects and the network server automatically shuts down.

7. Close the ODBC Sample by tapping the × in the top right-hand corner of the window.



## Configuring Windows CE databases

This section describes how to configure a database for Windows CE using special features in Sybase Central. You will also learn how to copy an existing Adaptive Server Anywhere database to your Windows CE device, and how to erase Windows CE databases.

### Database properties

Most SQL features available in the full version of Adaptive Server Anywhere are available on the Windows CE version as well. These include transaction processing, referential integrity actions, procedures and triggers, and so on. However, the Java features and the remote data access features are not available on Windows CE.

You should be mindful of the unsupported features when setting database properties on a database intended for a Windows CE device.

☞ For more information, see [“SQL Anywhere Studio features not supported on Windows CE”](#) on page 80.

### The transaction log

The transaction log stores all changes made to a database, in the order in which they are made. In the event of a media failure on a database file, the transaction log is essential for database recovery. It also makes your work more efficient. By default, the transaction log is placed in the same directory as the database file. It is created when the database is started for the first time on your Windows CE device.

When you copy an existing database to your Windows CE device, you can copy both the database and transaction log files. If you do not copy the transaction log file to the device, a new log will be created when you start the database on your Windows CE device. The new log will not contain the information contained in the original transaction log. This can be problematic if the database was not shut down properly the last time it was used, or if the database is involved in synchronization. Thus, the best practice is to copy both the database and the transaction log files to the Windows CE device.

☞ For more information, see [“The transaction log”](#) [*ASA Database Administration Guide*, page 378].

### Using jConnect on Windows CE

jConnect is a pure Java JDBC driver for Adaptive Server Anywhere.

---

Sybase Central gives you the option to enable the jConnect JDBC driver so that Java applications can access Adaptive Server Anywhere databases.

By default, jConnect is not enabled if you specify that the database is being created for Windows CE. However, you can choose to enable jConnect.

Adding jConnect support to a database adds many entries into the system tables. This adds to the size of the database and, more significantly, adds about 200kb to the memory requirements for running the database, even if you do not use any jConnect functionality.

If you are not going to use jConnect, and if you are running in a limited-memory environment like Windows CE, you may not wish to add jConnect support to your database.

## Encryption

You can choose to secure your database either with simple or with strong encryption. The only way to change the encryption setting after the database has been initialized is by rebuilding the entire database.

☞ For more information, see [“Encrypting a database”](#) [*SQL Anywhere Studio Security Guide*, page 15].

☞ For more information, see [“Keeping your Windows CE database secure”](#) [*SQL Anywhere Studio Security Guide*, page 23].

## Additional properties

The following properties are configured during database creation. Once set, these properties can only be changed by rebuilding the database.

- ◆ Case sensitivity or insensitivity

☞ For more information, see [“Case sensitivity”](#) [*ASA SQL User's Guide*, page 486].

- ◆ Password case sensitivity or insensitivity

☞ For more information, see [“Case sensitivity”](#) [*ASA SQL User's Guide*, page 486].

- ◆ Treatment of trailing blanks in comparisons

By default, databases are created with trailing blanks classified as extra characters. For example, ‘Dirk’ is not the same as ‘Dirk ’.

You can create databases with blank padding, so that trailing blanks are ignored.

- ◆ Page size

☞ For more information, see “Table and page sizes” [*ASA SQL User’s Guide*, page 424].

◆ Collation sequence

When creating databases for Windows CE, you should use a collation based on the same single- or multibyte character set that Windows would use for the language of interest. For example, if you are using English, French, or German, use the 1252Latin1 collation. If you are using Japanese, use the 932JPN collation, and if you are using Korean, use the 949KOR collation.

☞ For more information, see “Understanding collations” [*ASA Database Administration Guide*, page 335].

## Creating a Windows CE database

There are three methods for creating an Adaptive Server Anywhere database for your Windows CE device. You can use the Sybase Central Create Database wizard to create a database that can be copied directly to your Windows CE device, and you can use the Initialization utility (dbinit) or the CREATE DATABASE statement in Interactive SQL to create a database that can be manually copied to your Windows CE device. This section describes all of these methods.

### Creating a Windows CE database using Sybase Central

Sybase Central has features to make database creation easy for Windows CE. If you have Windows CE services installed on your Windows desktop machine, you have the option to create a Windows CE database using Sybase Central. Sybase Central enforces the requirements for Windows CE databases, and gives you the option of copying the resulting database file to your Windows CE device.

❖ **To create a Windows CE database in Sybase Central and copy it directly to your Windows CE device**

1. Start Sybase Central on your desktop machine.

From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.

2. Start the Create Database wizard.

From the Tools menu, choose Adaptive Server Anywhere 9 ► Create Database.

3. Choose Create a Database on This Computer, and click Next.

---

The database is initialized on the desktop machine before being copied to the Windows CE device.

4. Specify a file name and directory to store the database on your desktop machine, and click Next.

5. Select Create This Database for Windows CE, and then click Next.

This option ensures that the new database conforms to the settings required to run the database on Windows CE.

When you click Next, Sybase Central attempts to connect to your Windows CE device. Your Windows CE device must be connected to the desktop machine in order to continue to the next step.

6. Select Copy the Database to Your Windows CE Device to have Sybase Central automatically copy the database to your device once it is initialized.

7. Specify the Windows CE directory to copy your database files to. The default is the main device directory.

**Tip**

Copy the database to the My Documents directory of your Windows CE device to make it simpler to start the database.

When starting a database on your Windows CE device using the Server Startup Options dialog, you can use Browse to search for the database file in the My Documents directory only.

If the database is not stored in the My Documents directory, you must type the path of the database in the Database field of the Server Startup Options dialog.

Optionally, you can select the Delete the Desktop Database After Copying option.

If you choose not to delete the desktop copy, a copy of the database file will be stored on your desktop machine in the directory that you specified in step 4.

Click Next.

8. If you are saving a copy of the database on your desktop machine, you can specify the directory where you want to save the transaction log file.

On your Windows CE device, the log file will be created in the same directory as the database file when the database is started on the network server for the first time.

Click Next.

9. Leave the Maintain the Following Mirror Log File checkbox blank.

A mirror log can help in data recovery should the transaction log become corrupt. However, it is not necessary.

10. Leave the Install jConnect Meta-information Support checkbox blank.

This option is not necessary, and adds extra memory requirements for running the database.

11. Set the level of encryption for your database by selecting the appropriate radio button.

If you select strong encryption, you must specify an encryption key. It is recommended that you choose a value for your key that is at least 16 characters long, contains a mix of upper and lowercase, and includes numbers, letters, and special characters.

**Warning**

*Be sure to store a copy of your key in a safe location. You require the key each time you want to start or modify the database. A lost key will result in a completely inaccessible database, from which there is no recovery.*

12. Follow the remaining instructions in the wizard.

13. Click Finish to create the database and copy it to your device.

A dialog appears, tracking the progress of the files being copied to your Windows CE device.

14. Once the wizard has copied the database to your Windows CE device, you can verify the location of the files.

From the Start menu, tap Programs ► File Explorer and navigate to the Windows CE directory that you copied the database to.

The database file will be listed. The log file is not listed until the first time you start the database on your Windows CE device.

## Creating a Windows CE database using dbinit

The Initialization utility (dbinit) can be used to create databases. However, you cannot copy them directly to a Windows CE device from this utility. You must manually copy databases to your Windows CE device.

☞ For more information, see [“Copying a database to your Windows CE device” on page 59](#).

---

❖ **To create a database using the dbinit utility**

1. At the command prompt, navigate to the directory where you want to store your database.

For example:

```
cd temp
```

2. Create your database by typing the following command:

```
dbinit database-name.db
```

**Tip**

You can also configure database properties such as encryption and page size using the dbinit utility.

☞ For more information, see [“Creating a database using the dbinit command-line utility” \[ASA Database Administration Guide, page 531\]](#).

3. Verify the successful creation of the database file by typing the following command:

```
dir
```

The database file that you created is listed in the directory.

## Creating a Windows CE database using the CREATE DATABASE statement

The CREATE DATABASE statement can be used to create databases in Interactive SQL on your desktop machine. However, you cannot copy them directly to a Windows CE device from this application. You must manually copy databases to your Windows CE device.

☞ For more information, see [“Copying a database to your Windows CE device” on page 59](#).

❖ **To create a database using the CREATE DATABASE statement**

1. Open Interactive SQL on your desktop machine.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL.
2. The Connect dialog appears.  
If the Connect dialog does not appear automatically, from the SQL menu, choose Connect.
3. On the Identification tab, enter the following values:
  - ◆ **User ID:** DBA

◆ **Password:** SQL

4. Select the ODBC Data Source Name option.
5. Click Browse, and choose the ASA 9.0 Sample data source.
6. Click OK to connect to the sample database.
7. Type the following statement in the SQL Statements pane of Interactive SQL:

```
CREATE DATABASE 'c:\\temp\\database-name.db'
TRANSACTION LOG ON
```

**Tip**

You can also configure database properties such as encryption and page size using the CREATE DATABASE statement.

☞ For more information, see [“CREATE DATABASE statement” \[ASA SQL Reference, page 338\]](#).

8. From the SQL menu, choose Execute.

A database and transaction log are created in the `c:\temp` directory of your desktop machine.

## Copying a database to your Windows CE device

Any existing Adaptive Server Anywhere database can be copied to your Windows CE device using the method described in this section. However, you must keep in mind that any database properties that are not supported on Windows CE will not work when you copy the database to your Windows CE device.

☞ For more information, see [“Database properties” on page 53](#).

☞ For more information on creating databases, see [“Creating a database” \[ASA SQL User’s Guide, page 31\]](#).

❖ **To copy a database to your Windows CE device**

1. Connect the Windows CE device to your desktop machine.
2. Open Windows Explorer on your desktop machine.
3. Browse to the directory on your desktop machine containing the database that you want to copy.
4. Right-click the database file and choose Copy.

- 
5. Open a second instance of Windows Explorer.
  6. Browse to the directory on your Windows CE device where you want to store the database file.

**Tip**

When starting a database on your Windows CE device using the Server Startup Options dialog, you can use Browse to search for the database file in the My Documents directory only.

If the database is not stored in the My Documents directory, you must type the path of the database in the Database field of the Server Startup Options dialog.

7. Right-click an open area of the Windows Explorer window for your Windows CE device and choose Paste.

The file is copied to the Windows CE device.

## Erasing a Windows CE database

Adaptive Server Anywhere for Windows CE does not support the Erase Database wizard, the DROP DATABASE statement, or the Erase utility (dberase). You must manually erase databases from your Windows CE device.

There are two methods for erasing a database from your Windows CE device. You can erase a database through the device interface, or you can connect your device to a desktop machine and erase the database using Windows Explorer. This section describes both methods.

### Erasing a database using the device interface

❖ **To erase a database using the device interface**

1. From the Start menu, tap Programs ► File Explorer and navigate to the directory containing the database file that you want to erase.
2. Tap and hold on the database file.  
A popup menu appears.
3. Tap Delete.
4. Tap Yes to confirm that you do want to delete the file.



**Tips**

The database must not be running when you attempt to delete it.

After you delete the database, delete the transaction log file. If the database was never started, no transaction log was created.

## Erasing a database using Windows Explorer

### ❖ To erase a database using Windows Explorer

1. Place your Windows CE device in its cradle and ensure that it connects to the desktop machine via ActiveSync.
2. Open Windows Explorer on the desktop machine.
3. Browse to the Windows CE directory where the database file is stored.
4. Right-click the database file and choose Delete.  
A dialog appears, asking you to confirm that you would like to delete the database file. Click Yes.

**Tips**

The database must not be running when you attempt to delete it.

After you delete the database, delete the transaction log file. If the database was never started, no transaction log was created.

---

## Using the database server on Windows CE

This section describes the database server, and how to start, stop, and configure the database server on Windows CE.

The database server supplied for Windows CE is the network database server. Its filename is *dbsrv9.exe*. The network server supports communications over a TCP/IP network link.

The usual client/server arrangement has the database server running on a machine with more power and resources than the client applications. Clearly, this is not the case with Windows CE; instead, the less powerful machine is running the database server.

The advantage of supplying a network server on Windows CE is that you can run administration utilities on a desktop machine to execute tasks on your Windows CE database. For example:

- ◆ You can use Sybase Central on your desktop machine to manage your database.
- ◆ You can use Interactive SQL on your desktop to load and unload data, and carry out queries.

☞ For more information, see [“Using the administration utilities” on page 69](#).

### Server and database options

Server and database options are one of the major ways of tuning Adaptive Server Anywhere behavior and performance. You can choose from many options to specify such features as how much memory the cache can use, the level of access needed to start a database on the network server, and the network protocols to use.

On Windows CE, options are set in the Server Startup Options dialog. This is different than other Windows operating systems where database server options are set on the command line. Most server options are available for Windows CE.

☞ For more information about database server options, see [“The Database Server” \[ASA Database Administration Guide, page 115\]](#).

☞ For information on unsupported options, see [“Server options not supported on Windows CE” on page 84](#).

## Tutorial: Using the database server on Windows CE

When you have completed this tutorial, you will be able to perform key tasks associated with the database server: starting and stopping the server, and running single and multiple databases.

### Requirements

- ◆ Complete all of the tasks in “Getting started” on page 38 before attempting this tutorial.

### Before you begin

You need to create two Windows CE databases for use in the tutorial.

#### ❖ To create databases for your Windows CE device

1. Ensure that your Windows CE device is connected to the desktop machine.
2. Open Sybase Central on your desktop machine.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.  
Sybase Central appears.
3. Start the Create Database wizard.  
From the Tools menu, choose Adaptive Server Anywhere 9 ► Create Database.  
The Create Database Wizard appears.
4. Create a database called *Alpha.db*.  
Disregard the directory that the database is stored in on the desktop machine, as the desktop copy will be deleted when the database is copied to your Windows CE device.
5. Click Next.
6. Select the Create This Database for Windows CE option.
7. Click Next.  
The wizard tests the connection to your Windows CE device.
8. Select the Copy the Database to Your Windows CE Device option.  
Copy the database to the My Documents directory on your Windows CE device by typing the following in the Windows CE File Name field: `\\My Documents\\Alpha.db`.
9. Select the Delete the Desktop Database After Copying option.

10. Click Finish to create the database with the default settings.  
Once the wizard finishes copying the *Alpha.db* database file to your Windows CE device, continue to the next step.
11. Repeat steps 3 through 9 to create a database called *Beta.db*.  
The path in step 7 for the second database is *|My Documents|Beta.db*.  
Once the wizard finishes copying the *Beta.db* database file to your Windows CE device, you are ready to begin the tutorial.

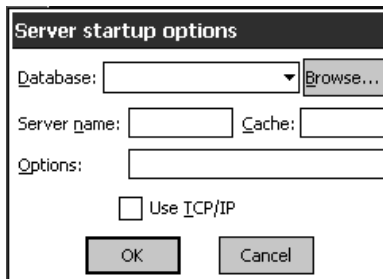
## Lesson 1: Starting the database server

This section describes the simple case of running a single database on Windows CE.

### ❖ To start a database on the server

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. Tap the *dbsrv9* icon.

The Server Startup Options dialog appears.



4. Specify the database file that you want to start.  
Tap Browse and locate the *Alpha.db* file in the My Documents directory.
5. Assign a name to the network server by typing **CEserver** in the Server field.
6. Use the default cache size of 600kb.

**Tip**

For the purposes of this tutorial, the default cache size is sufficient. However, a larger cache size can help improve performance for larger databases.

☞ For more information, see [“Using the cache to improve performance”](#) [*ASA SQL User’s Guide*, page 180].

7. Select the Use TCP/IP checkbox.

A TCP/IP connection is necessary to connect from a desktop machine to the database running on your Windows CE device. You will need to connect from your desktop machine in a later lesson.

8. In the Options field of the Server Startup Options dialog, type **-gd all**.

The `-gd` option sets the permissions to allow any user to start additional databases on the network server. This will be necessary in a later lesson.

☞ For more information, see [“-gd server option”](#) [*ASA Database Administration Guide*, page 142].

9. Tap OK to start the Alpha database running on the network server.

10. Navigate to the Today screen on your device.

11. Tap the Server icon located in the bottom right corner of the screen.

The Server Messages window appears.

When the message `Now accepting requests` appears in the Server Messages window, you are ready to proceed to the next lesson.

What’s next?

Next, you will learn how to start multiple databases on the network server on Windows CE.

## Lesson 2: Starting multiple databases on the server

On Windows CE devices, attempting to start a second Adaptive Server Anywhere server while a first server is already running brings the first server to the foreground. This is standard behavior for Windows CE applications. Because of this behavior, two servers cannot be run at the same time on a Windows CE device. As an alternative to multiple servers, one server can run multiple databases.

To start multiple databases on the server, you must first connect to one database running on your Windows CE device.

---

### ❖ To connect to the Alpha database using Sybase Central

1. Start Sybase Central on your desktop machine.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.
2. Choose Tools ► Connect.  
If a dialog prompts you to choose a plug-in, choose Adaptive Server Anywhere 9 and click OK.  
The Connect dialog appears.
3. On the Identification tab, enter the following values:
  - ◆ **User ID:** DBA
  - ◆ **Password:** SQL
4. Select the ODBC Data Source Name option.
5. Click Browse, choose the **CEdevice** data source that you created in [“Creating an ODBC data source to connect to your Windows CE device” on page 42](#) from the Data Source Names dialog, and click OK.
6. On the Database tab, type the server name in the Server Name field.  
In this example, the server name is CEserver, which you specified in the previous lesson.
7. Click OK to connect to the *Alpha.db* database running on your Windows CE device.

Now that you have started the server and connected to the Alpha database, you can start additional databases on your Windows CE device.

### ❖ To start multiple databases on the network server

1. In the left pane of Sybase Central, select the **CEserver** icon.  
This icon represents the network server running on your Windows CE device.
2. From the File menu, choose Start Database.  
The Start Database dialog appears.
3. In the Database File field, type the following path: **\\My Documents\Beta.db**
4. Click OK to start the database on the network server.

The database starts on the network server, but you must initiate a connection from your desktop machine.

❖ **To connect to the Beta database**

1. In the left pane of Sybase Central, select the **Beta** database icon.  
This icon represents the database that you just started on your Windows CE device.
2. From the File menu, choose Connect.  
The Connect dialog appears.
3. On the Identification tab, enter the following values:
  - ◆ **User ID:** DBA
  - ◆ **Password:** SQL
4. Click Browse and choose **CEdevice** from the Data Source Names dialog.
5. Click OK.
6. Click OK to connect to the Beta database running on your Windows CE device.

You can now view and manipulate the data in the Alpha and Beta databases using Sybase Central.

What's next?

Next, you will learn how to disconnect from the databases and shut down the database server on Windows CE.

### Lesson 3: Shutting down the server

Before you can shut down the network server on your Windows CE device, you must stop the connections from your desktop machine.

❖ **To disconnect from the Windows CE databases**

1. In Sybase Central, from the Tools menu, choose Disconnect.  
The Disconnect dialog appears.
2. Select the connection that corresponds to the Alpha database.
3. Click Disconnect.
4. Repeat steps 1 through 3 for the connection that corresponds to the Beta database.

Now that you have disconnected from the Windows CE databases in Sybase Central, you can shut down the network server.

---

❖ **To shut down the server**

1. Tap the database server icon located in the bottom right corner of the Today screen.

The Server Messages window appears.

2. Tap Shutdown.



## Using the administration utilities

This section describes specific considerations for using the SQL Anywhere Studio database administration utilities with Windows CE databases.

### Backup and recovery on Windows CE

Backup and recovery is vital to ensure you do not lose data in the event of data corruption or media failure. It is best to backup your Windows CE database to a physically separate location to safeguard against data loss due to theft or loss of the Windows CE device, or media failure on the Windows CE device. Most backup and recovery utilities are available on Windows CE. However, these utilities are not useful since you cannot use the utilities on Windows CE to store backups in a physically separate location.

Instead, data can be backed up by copying the entire database file to a desktop machine. You can also use synchronization to maintain an up-to-date copy of your Windows CE database on a desktop machine.

☞ For more information, see [“Synchronization Basics”](#) [*MobiLink Administration Guide*, page 7].

### Tutorial: Managing Windows CE databases with Sybase Central

Sybase Central is a database management tool that provides Adaptive Server Anywhere database settings, properties, and utilities in a graphical user interface. Sybase Central can also be used for managing other products, including MobiLink synchronization.

This tutorial provides a brief introduction to using Sybase Central from a desktop machine to manage databases on your Windows CE device. You will learn how to connect to the sample database on your Windows CE device. Once connected, you can use Sybase Central to view and edit data, manage users, and work with stored procedures.

#### Requirements

- ◆ Complete all of the tasks in [“Getting started”](#) on page 38 before you begin the tutorial.
- ◆ Connect your Windows CE device to a desktop machine.

#### Before you begin

Before you begin, be sure to make a copy of the sample database so you can restore it to its original form when you are finished.

#### ❖ To create a backup copy of the sample database

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.

- 
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
  3. In an open area of the screen, tap and hold your stylus.  
A popup menu appears.
  4. Tap New Folder.  
Name the new folder **Sample Backup**.
  5. Tap and hold the *asademo* database file.  
Two files named *asademo* may appear in the list. The larger of the two is the database file, while the other is the transaction log file. Be sure to select the correct file.
  6. Once the popup menu appears, tap Copy.
  7. Open the *Sample Backup* folder that you created by tapping it.  
The folder opens.
  8. Tap and hold a blank area of the screen.  
A popup menu appears.
  9. Tap Paste.  
A copy of the *asademo.db* file is pasted into the *Sample Backup* folder.

**Tip**

Work with the original version of the sample database. Do not change the backup copy.

## Lesson 1: Start the sample database

A database must be running on your Windows CE device if you want to connect to it from Sybase Central.


### ❖ To start the sample database

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. Tap the *dbsrv9* icon.  
The Server Startup Options dialog appears.

4. In the Database field type the path of the sample database: **\Program Files\Sybase\ASA9\asademo.db**.
5. Assign a name to the network server by typing **CEserver** in the Server field.
6. Use the default cache size of 600kb.

**Tip**

For the purposes of this tutorial, the default cache size is sufficient. However, a larger cache size can help improve performance for larger databases.

 For more information, see [“Using the cache to improve performance” \[ASA SQL User’s Guide, page 180\]](#).

7. Select the Use TCP/IP option.  
A TCP/IP connection is necessary to connect from a desktop machine to the database running on your Windows CE device.
8. Tap OK to start the sample database running on the network server.  
The Server Messages window appears briefly, then disappears.
9. Navigate to the Today screen on your device.
10. Tap the Server icon located in the bottom right-hand corner of the screen.  
The Server Messages window appears.  
When the message `Now accepting requests` appears in the Server Messages window, you are ready to move on to the next lesson.

What’s next?

Next you will learn how to connect from Sybase Central to the database running on your Windows CE device.

## Lesson 2: Start Sybase Central and connect

Now that the sample database is running on your Windows CE device, you can connect to it from Sybase Central. This allows you to view and manage the database from your desktop machine.

### ❖ To connect from Sybase Central to a database on your Windows CE device

1. Start Sybase Central on your desktop machine.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.

- 
2. Choose Tools ► Connect.

If a dialog prompts you to choose a plug-in, select Adaptive Server Anywhere 9, and then click OK.

The Connect dialog appears.

3. On the Identification tab, enter the following values:

- ◆ **User ID:** DBA

- ◆ **Password:** SQL

4. Select the ODBC Data Source Name option.

5. Click Browse, choose the *CEdevice* data source that you created in [“Creating an ODBC data source to connect to your Windows CE device” on page 42](#) from the Data Source Names dialog, and click OK.

6. On the Database tab, type the server name in the Server Name field.

In this example, the server name is CEserver, which you specified in the previous lesson.

7. Click OK to connect to the sample database running on your Windows CE device.

What's next?

You can now view and manage the data in the sample database from Sybase Central.

## Where do I go from here?

Learn more about Sybase Central

You can learn more about Sybase Central through the tutorial [“Managing Databases with Sybase Central” on page 241](#). The tutorial includes lessons on viewing and editing data, managing users, and working with stored procedures.

The tutorial pertains to using Sybase Central with a sample database running on a desktop machine. Thus, the following lessons do not apply when managing Windows CE databases:

- ◆ [“Lesson 1: Start Sybase Central and connect” on page 243](#)

Instead, use the connection procedure that you learned in this tutorial.

- ◆ [“Lesson 7: Back up your database” on page 258](#)

Disregard this lesson. The backup procedure described is not supported on Windows CE.

Administer several databases at once

You can administer several databases at one time by starting multiple databases on the network server on your Windows CE device.

☞ For more information, see “[Lesson 2: Starting multiple databases on the server](#)” on page 65.

## Tutorial cleanup

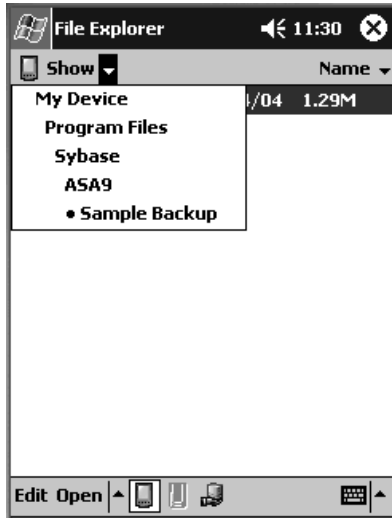
When you are finished the tutorial, you need to shut down the server, and restore the sample database from the backup copy that you made.

### ❖ To shut down the server

1. Tap the database server icon located in the bottom right corner of the Today screen.  
The Server Messages window appears.
2. Tap Shutdown.

### ❖ To restore the sample database

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. Tap the *Sample Backup* folder to open it.
4. Tap and hold the *asademo* file.  
A popup menu appears.
5. Tap Copy.
6. Navigate back to the ASA9 folder.  
Tap the navigation dropdown list at the top of the window. Then tap ASA9.



7. Tap and hold a blank area of the screen.  
A popup menu appears.
8. Tap Paste.
9. A warning dialog appears.  
Tap OK to replace the existing *asademo.db* file with the copy you made from the backup.

## Tutorial: Managing Windows CE databases with Interactive SQL

Interactive SQL is an application that allows you to query and alter data in your database, and modify the structure of your database. Interactive SQL provides a pane for you to enter SQL statements, as well as panes that display information about how the query was processed and the result set.

This tutorial provides a brief introduction to using Interactive SQL from a desktop machine to manage databases on your Windows CE device. You will learn how to connect to the sample database on your Windows CE device from Interactive SQL. Once connected, you can use Interactive SQL to execute SQL statements.

Before you begin

Before you begin, be sure to make a copy of the sample database so you can restore it to its original form when you are finished.

❖ **To create a backup copy of the sample database**

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. In an open area of the screen, tap and hold your stylus.  
A popup menu appears.
4. Tap New Folder.  
Name the new folder **Sample Backup**.
5. Tap and hold the *asademo* database file.  
Two files named *asademo* may appear in the list. The larger of the two is the database file, while the other is the transaction log file. Be sure to select the correct file.
6. Once the popup menu appears, tap Copy.
7. Open the *Sample Backup* folder that you created by tapping it.  
The folder opens.
8. Tap and hold a blank area of the screen.  
A popup menu appears.
9. Tap Paste.  
A copy of the *asademo.db* file is pasted into the *Sample Backup* folder.

**Tip**

Work with the original version of the sample database. Do not change the backup copy.

**Lesson 1: Start the sample database**

A database must be running on your Windows CE device if you would like to connect to it from Interactive SQL.

---

### ❖ To start the sample database

1. Open the File Explorer on your Windows CE device.  
From the Start menu, tap Programs ► File Explorer.
2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.
3. Tap the dbsrv9 icon.  
The Server Startup Options dialog appears.
4. In the Database field, type the path of the sample database: **\Program Files\Sybase\ASA9\asademo.db**.
5. Assign a name to the network server by typing **CEserver** in the Server field.
6. Set the cache size by typing **5MB** in the Cache field.  
The default cache size on Windows CE is 600kb. However, a larger cache size is recommended to help improve performance.
7. Select Use TCP/IP.
8. Click OK to start the sample database running on the network server.  
The Server window appears briefly, then disappears.
9. Navigate to the Today screen on your device.
10. Tap the Server icon located in the bottom right-hand corner of the screen.  
The Server window appears.  
When the message *Now accepting requests* appears in the Server window, you are ready to move on to the next lesson.

What's next?

Next you will learn how to connect from Interactive SQL to the database running on your Windows CE device.

## Lesson 2: Start Interactive SQL and connect

Now that the sample database is running on your Windows CE device, connect to it from Interactive SQL to view and manage the database from your desktop machine.



❖ **To connect from Interactive SQL to a database on your Windows CE device**

1. Start Interactive SQL on your desktop machine.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL.
2. The Connect dialog appears automatically when Interactive SQL launches.  
If the Connect dialog does not appear automatically, from the SQL menu, choose Connect.
3. On the Identification tab, enter the following values:
  - ◆ **User ID:** DBA
  - ◆ **Password:** SQL
4. Select the ODBC Data Source Name option.
5. Click Browse, choose the **CEdevice** data source that you created in [“Getting started” on page 38](#) from the Data Source Names dialog, and click OK.
6. On the Database tab, specify the server name in the Server Name field.  
In this example, the server name is CEserver, which you specified in the previous lesson.
7. Click OK to connect to the sample database running on your Windows CE device.

What's next?

You can now view and manage the data in the sample database from Interactive SQL.

### Where do I go from here?

Learn more about Interactive SQL

☞ You can learn more about Interactive SQL through the tutorial [“Using Interactive SQL” on page 217](#). The tutorial includes lessons on viewing and editing data, and working with SQL statements.

The tutorial pertains to using Interactive SQL with a sample database running on a desktop machine. Thus, the following lesson does not apply when managing Windows CE databases:

- ◆ [“Lesson 1: Start Interactive SQL” on page 219](#)

Instead, use the procedure that you learned in this tutorial.

---

## Tutorial cleanup

When you are finished the tutorial, you need to shut down the server, and restore the sample database from the backup copy that you made.

### ❖ To shut down the server

1. Tap the database server icon located in the bottom right corner of the Today screen.

The Server Messages window appears.

2. Tap Shutdown.

### ❖ To restore the sample database

1. Open the File Explorer on your Windows CE device.

From the Start menu, tap Programs ► File Explorer.

2. Navigate to the Adaptive Server Anywhere directory by tapping Program Files ► Sybase ► ASA9.

3. Tap the Sample Backup folder to open it.

4. Tap and hold the *asademo* file.

A popup menu appears.

5. Tap Copy.

6. Navigate back to the ASA9 folder.

Tap the navigation dropdown list at the top of the window. Then tap ASA9.



7. Tap and hold a blank area of the screen.  
A popup menu appears.
8. Tap Paste.
9. A warning dialog appears.  
Tap OK to replace the existing *asademo.db* file with the copy you made from the backup.

---

## SQL Anywhere Studio features not supported on Windows CE

This section lists the components and features of SQL Anywhere Studio that are unsupported or have altered functionality on Windows CE. Where available, alternatives to unsupported features are listed.

☞ For more information about supported and unsupported components on Windows CE, see [“Windows and NetWare operating systems” on page 98](#).

## Adaptive Server Anywhere features not supported on Windows CE

The following features are not supported on Windows CE:


Component or feature	Considerations
<b>Personal database server</b> The personal server is called <i>dbeng9</i> . This executable does not support client/server communications across a network.	In lieu of the personal database server, the network database server is provided. The network server is called <i>dbsrv9</i> . This executable supports local connections and client/server communications across a network.  ☞ For more information, see <a href="#">“Running the Database Server” [ASA Database Administration Guide, page 3]</a> .
<b>Open Client</b> Sybase Open Client provides customer applications, third-party products, and other Sybase products with the interfaces needed to communicate with Adaptive Server Anywhere and other Open Servers.	To run Open Client applications, you must install and configure Open Client components on the machine where the application is running. These components are not supported on Windows CE, and thus Open Client cannot function on the platform.
<b>Java in the database</b> Adaptive Server Anywhere is a runtime environment for Java. This means that Java classes can be executed in the database server.	Adaptive Server Anywhere for Windows CE does not act as a runtime environment for Java. Thus, Java classes cannot be executed in the database server on Windows CE.

Component or feature	Considerations
<p><b>Remote data access</b> Adaptive Server Anywhere remote data access gives you access to data in other data sources. You can use this feature to migrate data into an Adaptive Server Anywhere database and query data across databases.</p>	<p>This feature is not supported on Windows CE.</p> <p>If you copy a database to your Windows CE device that uses remote data access features, a communication error occurs when you try to access data in a proxy table.</p>
<p><b>External stored procedures</b> Adaptive Server Anywhere can call a function in an external library from a stored procedure or user-defined function.</p>	<p>Adaptive Server Anywhere for Windows CE does not support the ability to call a procedure from an external library, including system extended stored procedures.</p>
<p><b>Dynamic cache sizing</b> This feature causes the cache size to increase when necessary, and reduces the cache size when memory is needed for other applications.</p>	<p>Windows CE does not support dynamic cache sizing. On Windows CE, the cache is set on the Server Startup Options dialog and remains static.</p>
<p><b>SPX protocol</b> SPX is a communication protocol from Novell.</p>	<p>In lieu of the SPX protocol, Windows CE uses the TCP/IP protocol.</p>

---

## Adaptive Server Anywhere features limited on Windows CE

The following features have altered or limited functionality on Windows CE:

Component or feature	Considerations
<b>ODBC clients</b> Microsoft Corporation defines the Open Database Connectivity (ODBC) interface, which is a standard interface for connecting client applications to database-management systems in the Windows 95/98/Me and Windows NT/2000/XP environments. Many client applications, including application development systems, use the ODBC interface to access a wide range of database systems.	ODBC clients function differently than Windows desktop clients because Windows CE does not provide an ODBC driver manager or an ODBC Administrator. On this platform, Adaptive Server Anywhere uses ODBC data sources stored in files.  For more information, see “Using ODBC data sources on Windows CE” [ <i>ASA Database Administration Guide</i> , page 57].
<b>jConnect</b> The jConnect JDBC driver is a Java implementation of the JavaSoft JDBC standard. It provides Java developers with native database access in multi-tier and heterogeneous environments.	The jConnect driver can be enabled when you create a database for Windows CE. This can be useful if you want to move the database to a desktop machine that supports Java. However, enabling the jConnect driver adds to the size of the database and, more significantly, adds about 200 kb to the memory requirements for running the database. This additional memory requirement should be considered when running the database in a limited-memory environment like Windows CE.
<b>iAnywhere JDBC driver</b> The iAnywhere JDBC driver provides a JDBC driver that has some performance benefits and feature benefits compared to the pure Java jConnect JDBC driver, but which is not a pure-Java solution.	The iAnywhere JDBC driver can only be enabled if you have a Java Virtual Machine installed on your Windows CE device.

## SQL Remote features not supported on Windows CE

SQL Remote is supported on Windows CE with the following exceptions:

Component or feature	Considerations
<p><b>Adaptive Server Anywhere extraction utility (dbxtract)</b> This utility is used to extract a remote Adaptive Server Anywhere database from a consolidated Adaptive Server Enterprise or Adaptive Server Anywhere database.</p>	<p>Windows CE does not support this utility. If necessary, a Windows CE database can be copied to a desktop machine so that the Adaptive Server Anywhere extraction utility can be used.</p>
<p><b>MAPI message type</b> Microsoft's Messaging Application Programming Interface (MAPI), used in several popular e-mail systems such as Microsoft Mail, can be used to transport messages for SQL Remote messaging.</p>	<p>This message system is not supported on Windows CE.</p> <p>Windows CE supports several other message types used in SQL Remote replication, which are identified in "Supported operating systems" [<i>SQL Remote User's Guide</i>, page 447].</p>
<p><b>VIM message type</b> Lotus's Vendor Independent Messaging (VIM), used in Lotus Notes and cc:Mail can be used to transport messages for SQL Remote messaging.</p>	<p>This message system is not supported on Windows CE.</p> <p>Windows CE supports several other message types used in SQL Remote replication, which are identified in "Supported operating systems" [<i>SQL Remote User's Guide</i>, page 447].</p>

## Administration tools not supported on Windows CE

SQL Anywhere Studio includes several tools for administering databases. These include Sybase Central, Interactive SQL, and command line utilities. None of these administration tools can be deployed to Windows CE. Instead, database administration is performed from a Windows-based desktop machine that is connected to the Windows CE device.

☞ For more information, see "Using the administration utilities" on page 69.

---

## Server options not supported on Windows CE

This section describes those database server options that are not supported or have altered functionality on Windows CE.

Option	Considerations
<b>@data option</b> This option allows you to specify environment variables and configuration files on the command line.	Windows CE does not support environment variables.
<b>-? server option</b> This option is used to display usage information.	Windows CE does not support this option.
<b>-ca option</b> This option is used to enforce a static cache size.	Windows CE does not support dynamic cache sizing.
<b>-ch option</b> This option is used to set a maximum cache size, as a limit to automatic cache growth.	Windows CE does not support dynamic cache sizing.
<b>-cl option</b> This option is used to set a minimum cache size as a lower limit to automatic cache resizing.	Windows CE does not support dynamic cache sizing.
<b>-cs option</b> This option is used to display cache size changes in the Server Messages window.	Windows CE does not support dynamic cache sizing.
<b>-cw option</b> This option is used to enable use of Address Windowing Extensions (AWE) for setting the size of the database server cache.	Windows CE does not support AWE.
<b>-d option</b> This option is used to force the use of POSIX I/O.	Windows CE does not support POSIX I/O.



Option	Considerations
<p><b>-ec option</b> This option is used to encrypt all native Adaptive Server Anywhere packets (DBLib, ODBC, and OLE DB) transmitted to and from all clients. TDS packets are not encrypted.</p>	<p>Strong communication encryption is not supported on Windows CE. Only the <b>none</b> and <b>simple</b> settings are supported for this server option.</p>
<p><b>-gb option</b> This option is used to set the database process priority class.</p>	<p>Windows CE does not support Priority classes.</p>
<p><b>-ge option</b> This option is used to set stack size for external functions.</p>	<p>Windows CE does not support external functions.</p>
<p><b>-gss option</b> This option is used to set the stack size per internal execution thread in the server.</p>	<p>This option has no effect on Windows operating systems.</p>
<p><b>-gx option</b> This option is used to set the maximum number of requests that can concurrently execute blocking system calls.</p>	<p>This option is only of use on Windows NT. On other platforms, including Windows CE, the -gx value is equivalent to the -gt value.</p>
<p><b>-qi option</b> This option is used to control whether the database server tray icon and window appear.</p>	<p>When running, the network server appears as an icon in the bottom right-hand corner of the Today screen on your Windows CE device. This feature cannot be disabled.</p>
<p><b>-s option</b> This option is used to set the user ID for syslog messages.</p>	<p>This option only applies to UNIX operating systems.</p>
<p><b>-sc option</b> This option is used to disable the shared memory communications protocol and use Named Pipes.</p>	<p>Windows CE does not support the Named Pipes protocol.</p>

Option	Considerations
<b>-tmf option</b> This option is used for recovery from distributed transactions in unusual circumstances.	Windows CE does not support the Distributed Transaction Coordinator.
<b>-tmt option</b> This option is used to set a reenlistment timeout for participation in distributed transactions.	Windows CE does not support the Distributed Transaction Coordinator.
<b>-u option</b> This option is used to open files using the operating system disk cache.	This server option does not apply to Windows CE because the platform does not run on a hard disk.
<b>-ua option</b> This option is used to turn off use of asynchronous I/O.	This option only applies to Linux operating systems.
<b>-ud option</b> This option is used to run the server as a daemon.	This option only applies to UNIX operating systems.
<b>-ut option</b> This option is used to touch temporary files.	This option only applies to UNIX operating systems.
<b>-ux option</b> This option is used to open the Server Startup Options dialog or display the Server Messages window on Linux and Solaris.	This option only applies to Linux and Solaris operating systems.
<b>-y option</b> This option is used to make the server run as a Windows service.	Windows CE does not support Windows services.

## SQL statements not supported on Windows CE

When a client application wants to carry out a database task, such as retrieving information using a query or inserting a new row into a table, it does so using Structured Query Language (SQL) statements. This section

describes those SQL statements that are not supported on Windows CE, and those that have altered or limited functionality.

☞ For a complete list of SQL statements, see “SQL Statements” [*ASA SQL Reference*, page 253].

The following SQL statements are not supported on Windows CE:

SQL statement	Considerations
<p><b>ALTER WRITEFILE statement (deprecated)</b> This statement is used to change the name of the read-only database file to which a write file refers.</p>	<p>Windows CE does not support write files, and does not support this statement.</p>
<p><b>CREATE COMPRESSED DATABASE statement (deprecated)</b> This statement is used to create a compressed database from an existing database file, or to expand a compressed database.</p>	<p>Windows CE does not support compressed databases, and does not support this statement.</p>
<p><b>CREATE EXISTING TABLE statement</b> This statement is used to create a new proxy table, which represents an existing object on a remote server.</p>	<p>Windows CE does not support proxy tables, and does not support this statement.</p>
<p><b>CREATE EXTERNLOGIN statement</b> This statement is used to assign an alternate login name and password to be used when communicating with a remote server.</p>	<p>Connecting to a remote server is necessary to map remote objects to a proxy table in the local database. Windows CE does not support proxy tables, and does not support this statement.</p>
<p><b>CREATE SERVER statement</b> Use this statement to add a server to the SYSSERVERS system table.</p>	<p>Windows CE does not support proxy tables, and does not support this statement.</p>

SQL statement	Considerations
<p><b>CREATE WRITEFILE statement (deprecated)</b> This statement is used to create a write file for a database.</p>	<p>Windows CE does not support write files, and does not support this statement.</p>
<p><b>DROP DATABASE statement</b> This statement is used to delete all database files associated with a database.</p>	<p>The DROP DATABASE statement is not supported on Windows CE.</p>
<p><b>DROP SERVER statement</b> This statement is used to drop a remote server from the Adaptive Server Anywhere catalog.</p>	<p>Windows CE does not support remote data access, and does not support remote servers. Thus, the DROP SERVER statement is not supported on Windows CE.</p>
<p><b>INSTALL JAVA statement</b> This statement is used to make Java classes available for use within a database.</p>	<p>Java in the database is not supported on Windows CE. Thus, the INSTALL JAVA statement is not supported on Windows CE.</p>
<p><b>REMOVE JAVA statement</b> This statement is used to remove a class or a JAR file from a database. When a class is removed it is no longer available for use as a column or variable type. The class or JAR must already be installed.</p>	<p>Java in the database is not supported on Windows CE. Thus, the REMOVE JAVA statement is not supported on Windows CE.</p>
<p><b>REORGANIZE TABLE statement</b> This statement is used to defragment tables when a full rebuild of the database is not possible due to the requirements for continuous access to the database.</p>	<p>This statement is not supported on Windows CE.</p>

SQL statement	Considerations
<p><b>RESTORE DATABASE statement</b> This statement is used to restore a backed up database from an archive.</p>	<p>The archive file is created using the archive method of database backup, which is not supported on Windows CE. Thus, the RESTORE DATABASE statement is not supported on Windows CE.</p>
<p><b>START JAVA statement</b> This statement is used to start the Java Virtual Machine.</p>	<p>Java in the database is not supported on Windows CE. Thus, the START JAVA statement is not supported.</p>
<p><b>STOP JAVA statement</b> This statement is used to stop the Java Virtual Machine.</p>	<p>Java in the database is not supported on Windows CE. Thus, the STOP JAVA statement is not supported.</p>

The following SQL statements have altered or limited functionality on Windows CE:

SQL statement	Considerations
<p><b>ALTER DATABASE statement</b> This statement is used to upgrade a database created with previous versions of the software, or to add Java or jConnect support to a database.</p>	<p>The ALTER DATABASE UPGRADE syntax is used to add Java or jConnect support to the database. Windows CE does not support Java in the database, so it does not support this particular syntax of the ALTER DATABASE statement.</p>
<p><b>BACKUP statement</b> This statement is used to back up a database and transaction log.</p>	<p>Only the BACKUP DATABASE DIRECTORY syntax is supported on Windows CE.</p>
<p><b>CREATE DATABASE statement</b> This statement is used to create a database. The database is stored as an operating system file.</p>	<p>The CREATE DATABASE statement is not supported on Windows CE. However, the CREATE DATABASE statement can be used to initialize a database on a desktop machine, which can later be copied to a Windows CE device.</p>

SQL statement	Considerations
<p><b>CREATE EVENT statement</b> This statement is used to define an event and its associated handler for automating predefined actions. Also, to define scheduled actions.</p>	<p>DiskSpace event types are not supported on Windows CE. However, you can use this statement to define the GlobalAutoIncrement event type or the ServerIdle event type.</p>
<p><b>CREATE FUNCTION statement</b> This statement is used to create a new function in the database.</p>	<p>A function using the EXTERNAL NAME clause is a wrapper around a call to a function in an external library. External functions are not supported on Windows CE. Thus, this clause is not supported. However, the CREATE FUNCTION statement can be used on Windows CE to create user-defined SQL functions for use in the database.</p>
<p><b>CREATE TABLE statement</b> This statement is used to create a new table in the database and, optionally, to create a table on a remote server.</p>	<p>The AT clause creates a remote table on a different server and also a proxy table on the current database that maps to the remote table. Windows CE does not support proxy tables, and so does not support the AT clause of this statement. Otherwise, this statement is supported on Windows CE.</p>


## Sybase Central wizards not supported on Windows CE

Sybase Central is a database management tool that provides Adaptive Server Anywhere database settings, properties, and utilities in a graphical user interface. This section presents information on those wizards that are not supported or have altered functionality on Windows CE and provides alternatives where possible.

Wizard	Considerations
<p><b>Backup Database wizard</b> This wizard creates a single file containing a backup of the database and transaction log files. This type of backup is known as an archive backup.</p>	<p>The archive backup is not supported on Windows CE. The Backup Database wizard is not supported.</p> <p>☞ For more information, see “Types of backup” [<i>ASA Database Administration Guide</i>, page 381].</p> <p>Sybase Central also has the Create Backup Images wizard, which makes a separate backup of the database and transaction log files. This wizard is supported on Windows CE.</p> <p>☞ For more information, see “Backing up a database using the Create Backup Images wizard” [<i>ASA Database Administration Guide</i>, page 498].</p>
<p><b>Change Log File Settings wizard</b> With the Change Log File Settings wizard, you can display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.</p>	<p>This wizard cannot map to the Windows CE directory where the log files are stored. The Change Log File Settings wizard is not supported.</p>
<p><b>Compress Database wizard (deprecated)</b> The Compress Database wizard reads the given database file and creates a compressed database file. Compressed databases are usually 40 to 60 per cent of their original size.</p>	<p>Compressed databases are not supported on Windows CE. The Compress Database wizard is not supported on Windows CE.</p>

Wizard	Considerations
<p><b>Create Database wizard</b> With the Create Database wizard, you can initialize (create) a database. A number of database attributes are specified at initialization and cannot be changed later except by unloading, reinitializing, and rebuilding the entire database.</p>	<p>This wizard has features special to Windows CE database creation provided Windows CE services are installed on the desktop machine running Sybase Central.</p> <p>☞ For more information, see <a href="#">“Creating a Windows CE database” on page 55.</a></p>
<p><b>Erase Database wizard</b> This wizard is used to delete a database and related transaction log files.</p>	<p>This wizard cannot be executed on a database that is running. In order to administer a Windows CE database from a desktop machine, the database must be running on the network server on the Windows CE device.</p>
<p><b>Service Creation wizard</b> The Service Creation wizard is a tool used to create, delete, and modify Adaptive Server Anywhere services. Installing an application as a Windows service enables it to run even when you log off.</p>	<p>This wizard also involves mapping to a directory. Windows CE device directories cannot be mapped from this wizard. The Erase Database wizard is not supported on Windows CE.</p> <p>Windows CE cannot be configured to run an application as a service. The Service Creation wizard is not supported on Windows CE.</p>
<p><b>Translate Log File wizard</b> With the Translate Log File wizard, you can translate a transaction log into a SQL command file.</p>	<p>This wizard involves mapping to a directory. Windows CE device directories cannot be mapped from this wizard. The Translate Log File wizard is not supported on Windows CE.</p>



Wizard	Considerations
<p><b>Uncompress Database wizard</b> With the Uncompress Database wizard, you can expand a compressed database file created by the Compression utility or Compress Database wizard. The Uncompress Database wizard reads the compressed file and restores the database file to its uncompressed state.</p>	<p>Compressed databases are not supported on Windows CE. Thus, the Uncompress Database wizard is not supported on Windows CE.</p>
<p><b>Unload Database wizard</b> With the Unload Database wizard, you can unload a database and put a set of data files in a named directory. The Unload Database wizard creates an Interactive SQL command file to rebuild your database. It also unloads all of the data in each of your tables into files in the specified directory in comma-delimited format.</p>	<p>This wizard cannot map to the Windows CE directory where the database files are stored. Thus, the Unload Database wizard is not supported.</p> <p>However, you can unload a Windows CE database by copying it to your desktop machine and using the Unload Database wizard.</p>
<p><b>Upgrade Database wizard</b> The Upgrade wizard upgrades a database from an older version of Adaptive Server Anywhere to a newer version of Adaptive Server Anywhere, and allows you to take advantage of the newer release's list of features.</p>	<p>This wizard is not supported on Windows CE. However, you can upgrade a Windows CE database by copying it to your desktop machine and using this wizard before copying the database back to your Windows CE device.</p> <p> For more information, see “Upgrading a database using the Upgrade Database wizard” [<i>ASA Database Administration Guide</i>, page 599].</p>



---

## CHAPTER 5

# SQL Anywhere Studio Supported Platforms

### About this chapter

This chapter lists the supported platforms for the individual components of SQL Anywhere Studio.

Supported platform information is also available at [http://www.iAnywhere.com/products/supported\\_platforms\\_9.html](http://www.iAnywhere.com/products/supported_platforms_9.html) and at [http://www.iAnywhere.com/products/supported\\_platforms.html](http://www.iAnywhere.com/products/supported_platforms.html).

#### **Availability**

Listing a platform as supported does not necessarily indicate immediate availability of the software. For example, for a given software version not all platforms are released simultaneously. Features and platform support for unreleased software are subject to change.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	96
<a href="#">Windows and NetWare operating systems</a>	98
<a href="#">UNIX, Linux, and Macintosh operating systems</a>	103
<a href="#">MobiLink synchronization consolidated databases</a>	107
<a href="#">iAnywhere Solutions ODBC drivers supported platforms</a>	108
<a href="#">UltraLite target platforms</a>	109
<a href="#">Operating system versions</a>	111

---

# Introduction

SQL Anywhere Studio provides database management and synchronization technologies for a wide variety of operating systems. It includes the following major components:

- ◆ **Adaptive Server Anywhere** A relational database system for use all the way from a Small and Medium Business (SMB) database server through desktop database applications down to Windows CE devices.
- ◆ **UltraLite** A relational database designed specifically for small devices. It runs on Palm OS and Windows CE devices.
- ◆ **MobiLink** A synchronization system that provides two-way synchronization between many Adaptive Server Anywhere or UltraLite databases and a central database server.
- ◆ **SQL Remote** A synchronization system for Adaptive Server Anywhere and Sybase Adaptive Server Enterprise databases.
- ◆ **Administration tools** A suite of tools provides management for all components in SQL Anywhere Studio.

This paper lists which components of SQL Anywhere Studio run on which operating systems. The information is organized by component within SQL Anywhere Studio.

☞ This chapter provides information current at the time of writing. For the latest information, see [http://www.iAnywhere.com/products/supported\\_platforms.html](http://www.iAnywhere.com/products/supported_platforms.html).

## Overview

The following table summarizes platform support for the major components. The table does not list detailed information such as differences in support for individual features on the listed platforms, or which flavors and versions of UNIX are supported. For detailed information, see the tables in the remainder of this document.

The UNIX platform support is of two kinds.

- ◆ **Full edition** Complete releases of SQL Anywhere Studio are available on Sun Solaris and Linux.
- ◆ **Deployment edition** The deployment edition contains core Adaptive Server Anywhere. Some deployment releases contain the MobiLink synchronization server. Deployment edition platforms include HP-UX, IBM AIX, Compaq Tru64, and 64-bit Linux operating systems.

☞ The supported versions of the listed operating systems are as described in more detail in “[Operating system versions](#)” on page 111.

Component	Windows (32-bit)	Windows (64-bit)	Mac OS X	UNIX (full)	UNIX (deployment)	NetWare	Windows CE	Palm OS
Adaptive Server Anywhere database server	✓	✓	✓	✓	✓	✓	✓	
Adaptive Server Anywhere clients	✓	✓	✓	✓	✓	✓ <sup>1</sup>	✓	
UltraLite databases	✓ <sup>2</sup>						✓	✓
UltraLite development	✓							
MobiLink synchronization server	✓	✓	✓	✓	✓ <sup>3</sup>			
SQL Remote	✓	✓	✓	✓		✓	✓	
Replication Agent	✓	32-bit software		✓				
Administration tools	✓	32-bit JRE	✓	✓		✓ <sup>4</sup>		

<sup>1</sup>Embedded SQL only<sup>2</sup>For development and testing only<sup>3</sup>32-bit HP-UX support on PA RISC (9.0.1 and later) and AIX support only<sup>4</sup>Limited set

# Windows and NetWare operating systems

The following tables list the supported operating systems for various components and features of SQL Anywhere Studio.

Unless otherwise noted, the supported versions of the listed operating systems are as described in [“Operating system versions” on page 111](#).

## Adaptive Server Anywhere

System requirements depend on the database size, workload, and required performance. The following are minimal requirements only.

Adaptive Server Anywhere (both personal database server and network database server) can run with as little as 4 MB of memory with an additional 8 KB per client connection. More memory will improve performance significantly. If you use Java in the database, Adaptive Server Anywhere requires an additional 5 MB of memory. Your computer must have this much memory in addition to the requirements for the operating system.

Component or Feature	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
Personal database server	✓	✓		
Network database server	✓	✓	✓	✓
ODBC clients	✓	✓	✓ <sup>5</sup>	
OLE DB clients	✓	✓	✓	
Embedded SQL clients	✓	✓	✓	✓
Open Client clients	✓	✓		
JDBC clients (jConnect)	✓	32-bit software	✓	
JDBC clients (iAnywhere JDBC driver)	✓	✓		
SQL preprocessor	✓	32-bit software		
Java in the database	✓			✓
Remote data access <sup>6</sup>	✓	✓		✓ <sup>7</sup>
External stored procedures	✓	✓ <sup>8</sup>		✓

<sup>5</sup>ODBC driver manager may not be available. May have to link directly to ODBC driver.

<sup>6</sup>Requires iAnywhere Solutions ODBC driver on Windows. For a list of supported platforms, see [“iAnywhere Solutions ODBC drivers supported platforms” on page 108](#).

<sup>7</sup>Requires Java in the database and JDBC driver

<sup>8</sup>The older interface to external stored procedures is available only in 32-bit software.

Component or Feature	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
Dynamic cache resizing	✓	✓		
Strong encryption	✓	✓	✓	✓
CREATE DATABASE, RESTORE DATABASE, DROP DATABASE statements	✓	✓		✓
SPX protocol	✓			✓

#### UltraLite development platforms

Component	Windows NT / 2000 / XP	Windows (64-bit)	Windows 98 SE	Windows CE	NetWare
SQL preprocessor (Embedded SQL only)	✓		✓		
UltraLite generator (static C++ API, static Java API)	✓		✓		
UltraLite for AppForge MobileVB	✓				
UltraLite ActiveX	✓				
Native UltraLite for Java	✓				
UltraLite.NET	✓ <sup>9</sup>				
UltraLite for C++	✓				

Although most 32-bit software works under emulation in 64-bit Windows, the static Java API is not available on 64-bit Windows.

☞ For information about deployment platforms, see [“UltraLite target platforms” on page 109](#).

#### Development environments

You can use one of the following development environments to build UltraLite applications for Palm OS:

- ◆ Metrowerks CodeWarrior version 8 or 9.

<sup>9</sup>Requires Visual Studio.NET or Visual Studio.NET 2003, with .NET Compact Framework version 1.0.5000 or later.

CodeWarrior includes a version of the Palm SDK. Depending on the particular devices you are targeting, you may want to upgrade your Palm SDK to a more recent version than that included in the development tool.

- ◆ AppForge MobileVB or Crossfire, using the UltraLite MobileVB component.

MobiLink synchronization server

Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
MobiLink synchronization server	✓	✓		
TCP/IP synchronization	✓	✓		
HTTPS synchronization	✓	✓		
HTTP synchronization	✓	✓		
Transport-Layer Security	✓			
Java synchronization logic	✓	32-bit software		
.NET synchronization logic	✓	✓		
MobiLink with Messaging (QAnywhere)	✓	✓		
Server-Initiated Synchronization	✓	✓		

MobiLink Adaptive Server Anywhere clients

Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
ASA MobiLink clients ( <i>dbmlsync</i> )	✓	✓	✓	
TCP/IP synchronization	✓	✓	✓	
HTTP synchronization	✓	✓	✓	
HTTPS synchronization	✓	✓	✓	
Transport-layer security	✓	✓	✓	
QAnywhere clients	✓	✓	✓	

MobiLink utilities



Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
Redirector	✓ <sup>10</sup>			
MobiLink extraction utility ( <i>mlx-tract</i> )	✓	✓		

## SQL Remote

SQL Remote components that operate against Adaptive Server Enterprise databases require Sybase Open Client and/or Open Server libraries.

Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
ASA Message Agent ( <i>dbremote</i> )	✓	✓	✓	✓
ASE Message Agent ( <i>ssremote</i> )	✓	32-bit software		
ASE stable queue ( <i>ssqueue</i> )	✓	32-bit software		
ASA extraction utility ( <i>dbxtract</i> )	✓	✓		
ASE extraction utility ( <i>ssxtract</i> )	✓	32-bit software		
File message type	✓	✓	✓	✓
FTP message type	✓	✓	✓	✓
MAPI message type	✓	✓		
SMTP message type	✓	✓	✓	✓
VIM message type	v			

## Replication agent

The Replication Agent requires Sybase Open Client and Open Server libraries.

Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
Replication Agent	✓	32-bit software		

## Administration tools

<sup>10</sup>Windows NT/2000/XP only

---

Component	Windows (32-bit)	Windows (64-bit)	Windows CE	NetWare
Command line administration utilities	✓	✓		✓
Sybase Central	✓	32-bit JRE		
Interactive SQL	✓	32-bit JRE		✓ <sup>11</sup>
DBConsole	✓	32-bit JRE		✓ <sup>12</sup>

With the exception of the command line administration utilities and the tools provided on NetWare, the administration tools employ a Java 2 runtime environment, version 1.4.1.

The following paragraphs are taken from the Java 2 SDK, Standard Edition documentation, and apply to the administration tools:

*The Java™ 2 SDK is intended for use on Microsoft Windows 95, 98 (1st or 2nd edition), NT 4.0 with Service Pack 5, Me, 2000 Professional, 2000 Server, 2000 Advanced Server, or XP operating systems running on Intel hardware.*

*A Pentium 166MHz or faster processor with at least 32 megabytes of physical RAM is required to run graphically based applications. Forty-eight megabytes of RAM is recommended for applets running within a browser using the Java Plug-in product. Running with less memory may cause disk swapping which has a severe effect on performance. Very large programs may require more RAM for adequate performance.*

*You should have 70 megabytes of free disk space before attempting to install the Java 2 SDK software. If you also want to install the documentation download bundle, you will need an additional 120 megabytes of free disk space.*

---

<sup>11</sup>A more limited version than is provided on other operating systems

<sup>12</sup>A more limited version than is provided on other operating systems

## UNIX, Linux, and Macintosh operating systems

Unless otherwise noted, the supported versions of the listed operating systems are as described in “[Operating system versions](#)” on page 111.

### Adaptive Server

#### Anywhere

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
Personal database server	✓	✓	✓	✓	✓	✓
Network database server	✓	✓	✓	✓	✓	✓
ODBC clients <sup>13</sup>	✓	✓	✓	✓	✓	✓
Embedded SQL clients	✓	✓	✓	✓	✓	✓
SQL preprocessor	✓	✓	✓	✓	✓	✓
Open Client clients	✓	✓	✓	✓	✓	✓
JDBC clients (jConnect driver)	✓	✓	✓	✓	✓	✓
Java in the database	✓	✓	✓	✓	✓	✓
Remote data access <sup>14</sup>	✓	✓	✓			
External stored procedures	✓	✓	✓	✓	✓	✓
iAnywhere JDBC driver <sup>15</sup>	✓	✓	✓	✓	✓	✓

**Linux Note for Adaptive Server Anywhere** Crashes have been observed under the 2.4.2 SMP kernel when using multiple processors. Under some conditions, the operating system may cause the process memory to be corrupted resulting in subsequent application crashes. Later 2.4.x kernels do not exhibit this problem.

### MobiLink synchronization server

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
MobiLink synchronization server	✓	✓	✓	✓	✓	

<sup>13</sup>ODBC driver manager may not be available. May have to link directly to ODBC driver.

<sup>14</sup>JDBC remote data sources require Java in the database and a JDBC driver. ODBC remote data sources requires iAnywhere Solutions ODBC driver. For a list of supported platforms, see “[iAnywhere Solutions ODBC drivers supported platforms](#)” on page 108.

<sup>15</sup>Requires JRE 1.4 or higher on Itanium chips. Requires JRE 1.3 or higher on other chips.

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
TCP/IP synchronization	✓	✓	✓	✓	✓	
HTTPS synchronization	✓	✓	✓	✓	✓	
HTTP synchronization	✓	✓	✓	✓	✓	
Transport-Layer Security	✓	✓	✓			
Java synchronization logic	✓	✓	✓	✓	✓	
.NET synchronization logic						
MobiLink with Messaging (QAnywhere)	✓	✓	✓	✓	✓	
Server-Initiated Synchronization	✓	✓	✓	✓	✓	

MobiLink support for HP-UX is available only on 32-bit PA RISC, and starts with version 9.0.1. Mac OS X support for TLS also starts with 9.0.1.

If you use Java synchronization logic, MobiLink with Messaging, or Server-Initiated Synchronization on HP-UX or AIX, you must set SHLIB\_PATH (on HP-UX) or LIBPATH (on AIX) to include the directory containing an installed JRE.

#### Adaptive Server Anywhere MobiLink client

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
ASA MobiLink clients ( <i>dbmlsync</i> )	✓	✓	✓			
TCP/IP synchronization (client)	✓	✓	✓			
HTTPS synchronization (client)	✓	✓	✓			
HTTP synchronization (client)	✓	✓	✓			
MobiLink extraction utility ( <i>mlxtract</i> )	✓	✓	✓			

#### SQL Remote

SQL Remote components that operate against Adaptive Server Enterprise databases require Sybase Open Client and/or Open Server libraries. These

libraries are not available on Compaq Tru64.

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
ASA Message Agent ( <i>dbremote</i> )	✓	✓	✓			
ASE Message Agent ( <i>ssremote</i> )	✓	✓				
ASE stable queue ( <i>ssqueue</i> )	✓	✓				
ASA extraction utility ( <i>dbextract</i> )	✓	✓	✓			
ASE extraction utility ( <i>ssextract</i> )	✓	✓				
File message type	✓	✓	✓			
FTP message type	✓	✓	✓			
MAPI message type						
SMTP message type	✓	✓	✓			
VIM message type						

Replication agent      The Replication Agent requires Sybase Open Client and Open Server libraries.

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
Replication Agent	✓	✓				

Administration tools

Component	Solaris	Linux	Mac OS X	HP-UX	AIX	Tru64
Command line administration utilities	✓	✓	✓	✓	✓	✓
Sybase Central	✓	✓	✓			
Interactive SQL	✓	✓	✓			

---

With the exception of the command-line administration utilities, the administration tools employ a Java 2 runtime environment, version 1.3.1.

The following paragraphs are taken from the Java 2 SDK, Standard Edition Version 1.3.1 system requirement for Solaris, and apply to the administration tools:

*The Java 2 SDK, Standard Edition, v. 1.3.1 (J2SDK 1.3.1) is intended for use on Solaris 2.6, Solaris 7, Solaris 8, and Solaris 9 Operating Environments.*

*Prior to installing the Java 2 SDK, you should insure that you have installed the full set of required patches needed for support of this release. To obtain patches, see the SunSolve support web site. You will find a patch cluster for each Solaris operating environment platform. Each patch cluster applies to all supported versions of the Java 2 Standard Edition (J2SE) on the given platform.*

*See also Solaris Font Package Requirements for information about which font packages should be on your system.*

## MobiLink synchronization consolidated databases

MobiLink supports the following consolidated databases:

Database server	Version
Adaptive Server Anywhere	Current
Sybase Adaptive Server Enterprise	11.5 and later
Oracle	8i and 9i
Microsoft SQL Server	7 and 2000
IBM DB2	7 and 8 UDB

### Dependencies

For a list of supported operating systems, see “[MobiLink synchronization server](#)” on page 100 and “[MobiLink synchronization server](#)” on page 103.

For the MobiLink synchronization server to connect to a consolidated database, an ODBC driver is required. iAnywhere Solutions provides a selection of ODBC drivers described in “[iAnywhere Solutions ODBC drivers supported platforms](#)” on page 108. Some database vendors also provide their own ODBC drivers.

For a list of ODBC drivers and their status see *Recommended ODBC Drivers for MobiLink* at [http://www.iAnywhere.com/developer/technotes/odbc\\_mobilink.html](http://www.iAnywhere.com/developer/technotes/odbc_mobilink.html).

---

## iAnywhere Solutions ODBC drivers supported platforms

iAnywhere Solutions ODBC drivers provide ODBC access to third-party databases for the MobiLink synchronization server and for Adaptive Server Anywhere remote data access.

The Adaptive Server Anywhere ODBC driver is not listed in this section. For information on the Adaptive Server Anywhere ODBC driver, see [“Windows and NetWare operating systems” on page 98](#).

iAnywhere Solutions ODBC drivers are available for the following operating system and database combinations:

<b>Operating system</b>	<b>Database</b>
Windows (32-bit)	Sybase Adaptive Server Enterprise 11.5 and later Oracle 8i and 9i IBM DB2 7 and 8 UDB
Solaris 7.0, 8.0, and 9.0	Sybase Adaptive Server Enterprise 11.5 and later Oracle 8i and 9i
IBM AIX	Sybase Adaptive Server Enterprise 11.5 and later Oracle 8i and 9i IBM DB2 7 and 8 UDB
Linux <sup>16</sup>	Sybase Adaptive Server Enterprise 11.5 and later Oracle 8i and 9i
HP-UX (32-bit PA RISC only)	Adaptive Server Anywhere IBM DB2 7 and 8 UDB

---

<sup>16</sup>Red Hat, SuSE, and Caldera distributions only.



## UltraLite target platforms

The following table lists the supported operating systems for UltraLite deployment, including synchronization streams.

☞ For a list of supported development platforms, see [“UltraLite development platforms” on page 99](#).

☞ For an explanation of the column headings, see [“Operating system versions” on page 111](#).

Component	Win- dows CE	Palm	Windows XP
UltraLite static C/C++ API	✓	✓	✓
UltraLite embedded SQL	✓	✓	✓
UltraLite static Java API <sup>17</sup>	✓		✓
UltraLite for AppForge Mo- bileVB	✓	✓	✓
UltraLite ActiveX	✓		✓
UltraLite .NET	✓ <sup>18</sup>		✓ <sup>19</sup>
Native UltraLite for Java	✓ <sup>20</sup>		✓
UltraLite C++ component	✓	✓	✓
UltraLite for M-Business Any- where	✓ <sup>21</sup>	✓ <sup>22</sup>	
TCP/IP synchronization	✓	✓	✓
HTTP synchronization	✓	✓	✓
HTTPS synchronization	✓	✓	✓
HotSync synchronization		✓	

<sup>17</sup>Requires PersonalJava or JDK 1.2.

<sup>18</sup>Requires .NET Compact Framework version 1.0.3705 or later.

<sup>19</sup>Requires .NET Compact Framework version 1.0.5000 or later.

<sup>20</sup>Deployment onto a CE/ARM device requires a Java VM on the device. Native UltraLite for Java should work with any PersonalJava 1.2 (or higher)-based VM, however, only the PersonalJava-based Jeode VM and the CrEme VM (version 3.24 or higher) have been tested.

<sup>21</sup>Requires Pocket PC on an ARM processor

<sup>22</sup>Requires Palm OS 5

---

Component	Windows CE	Palm	Windows XP
ActiveSync synchronization (3.5 and 3.6)	✓ <sup>23, 24</sup>		
Transport-layer security over HTTP or TCP/IP <sup>25</sup>	✓ <sup>26</sup>	✓	✓

**Notes**

UltraLite dynamic memory requirements mean that devices with Palm OS 3.5, or devices with less than 4 MB of memory, may not run for all but very small database schemas.

---

<sup>23</sup>Not supported by UltraLite components except Native UltraLite for Java and UltraLite.NET

<sup>24</sup>Not supported on Smartphone 2002

<sup>25</sup>Not supported by UltraLite components. Static Java API requires JDK 1.2.2 or greater. JDK 1.4 is not supported.

<sup>26</sup>Pocket PC required

## Operating system versions

### Availability

Features and platform support for unreleased software are subject to change. Listing a platform as supported does not necessarily indicate immediate availability of the software.

For example, for a given software version not all platforms are released simultaneously, and information about later platforms is not guaranteed to be correct in earlier releases.

Unless otherwise specified, operating system names correspond to the following versions:

Operating system	9.0.0	9.0.1	9.0.2
Microsoft Windows (32-bit)	<p>The following are supported:</p> <ul style="list-style-type: none"> <li>◆ Windows NT 4.0 or later</li> <li>◆ Windows 2000</li> <li>◆ Windows XP, including Windows XP embedded and Windows XP Tablet PC on x86-architecture processors.</li> <li>◆ A list of components required on the Windows XP Embedded image can be found at <a href="http://www.sybase.com/detail?id=1019835">http://www.sybase.com/detail?id=1019835</a>.</li> <li>◆ Windows Server 2003, 32-bit versions.</li> <li>◆ Windows 95/98/Me.</li> </ul>	Unchanged from 9.0.0	Unchanged from 9.0.1
Microsoft Windows (64-bit)	Windows Server 2003, 64-bit versions.	Unchanged from 9.0.0	Unchanged from 9.0.1

Operating system	9.0.0	9.0.1	9.0.2
Microsoft Windows CE	<p>Windows CE 3.0 operating system on any of the following processors:</p> <ul style="list-style-type: none"> <li>◆ MIPS processor</li> <li>◆ ARM processors, including the XScale series.</li> <li>◆ x86 processor. UltraLite supports the x86 processor for the Pocket PC 2002 emulator; Adaptive Server Anywhere supports x86 devices and emulation.</li> </ul> <p>The Windows CE emulator is also supported for development purposes.</p> <p>Windows CE 3.0 includes support for Pocket PC, including Pocket PC 2002, as well as Handheld PC.</p> <p>Windows CE 4.1 on the XScale (Intel PXA255) processor in ARMV4 mode.</p> <p>Pocket PC 2003 and other versions of Windows CE 4.2 on the XScale (Intel PXA255) processor in ARMV4 mode.</p>	<p>Windows CE 4.1 running in “Thumb” mode on the ARM processor, known as ARMV4T, is supported starting with version 9.0.1.</p> <p>UltraLite supports the Smartphone 2002 platform in addition to the other Windows CE platforms.</p> <p>The Pocket PC 2000 emulator is no longer supported.</p>	Unchanged from 9.0.1
Novell NetWare	5.1 with service pack 6, 6.0 with service pack 3, and 6.5.	Unchanged from 9.0.0	Unchanged from 9.0.1
Palm OS	Palm Computing Platform devices running the Palm OS version 3.5, 4.x, or 5.0.	Unchanged from 9.0.0	Unchanged from 9.0.1

Operating system	9.0.0	9.0.1	9.0.2
Linux	<p>Linux distributions are all based on defined versions of the kernel, the C library (glibc) and the ncurses library. The supported versions are as follows:</p> <ul style="list-style-type: none"> <li>◆ <b>kernel</b> 2.4.18 or 2.4.2</li> <li>◆ <b>glibc</b> 2.1.2, 2.1.3, or 2.2</li> <li>◆ <b>ncurses</b> 4.2</li> </ul> <p>The following Linux distributions are among those that meet the above requirements.</p> <ul style="list-style-type: none"> <li>◆ <b>Red Hat</b> 6.1, 6.2, 7.0, 7.1, 7.2, 7.3, or Advanced Server 2.1.</li> <li>◆ <b>SuSE</b> 6.3, 6.4, 7.0, 7.2, 7.3, or 8.0 (but not 7.1)</li> <li>◆ <b>Caldera</b> 2.4 or 3.e</li> <li>◆ <b>TurboLinux</b> 6.1, 6.5 or 7.0</li> <li>◆ <b>Mandrake</b> 7.2, 8.0, or 8.1</li> </ul> <p>Linux distributions are supported on x86-architecture processors.</p> <p>The 64-bit deployment edition is available for Red Hat Advanced Server on Itanium 2 processors.</p>	<p>32-bit Linux distributions with the following versions of the kernel and glibc:</p> <ul style="list-style-type: none"> <li>◆ Kernel 2.4.2 to 2.4.21 and glibc 2.2.1 to 2.3.2</li> <li>◆ Kernel 2.6.0 and glibc 2.3.2</li> </ul> <p>64-bit Linux distributions on the Itanium chip with the following versions of the kernel and glibc:</p> <ul style="list-style-type: none"> <li>◆ Kernel 2.4.9 with glibc 2.2.4, (for example, RedHat Advanced Server 2.1)</li> <li>◆ Kernel 2.6.0 with glibc 2.3.2 (for example, RedHat Enterprise Server 3)</li> </ul>	<p>Changes are possible before the release of this version.</p>
Sun Solaris	<p>7.0<sup>27</sup>, 8.0, or 9.0 32-bit or 64-bit operating systems on SPARC processors.</p>	<p>Unchanged from 9.0.0</p>	<p>Changes are possible before the release of this version.</p>

<sup>27</sup>Patches 106541-14 and 106980-13 recommended to resolve observed SIGIO problems.

<b>Operating system</b>	<b>9.0.0</b>	<b>9.0.1</b>	<b>9.0.2</b>
Mac OS X	Mac OS X version 10.2.n, where n is 2 or higher	Version 10.2 and 10.3 on the G4 processor. Version 10.3 on the G5 processor.	Changes are possible before the release of this version.
Compaq Tru64	4.0D, 5.1, or 5.2 on Alpha processors.	4.2, 5.1, or 5.2 on Alpha processors.	Changes are possible before the release of this version.
Hewlett Packard HP-UX	11.0 or 11i, 32-bit version on PA-RISC processors. HP-UX 11i is also supported in a 64-bit version on Itanium 2 processors.	Unchanged from 9.0.0	Changes are possible before the release of this version.
IBM AIX	4.3.3, 5.1, or 5.2 on Power PC processors	Unchanged from 9.0.0	Changes are possible before the release of this version.

PART III

# WORKING WITH DATABASES

This part introduces the relational database model and the basics of data manipulation





---

## CHAPTER 6

# The Architecture of Database Applications

About this chapter

This section introduces some of the terms and concepts that are important when talking about relational databases.

Contents

<b>Topic:</b>	<b>page</b>
Relational database concepts	118
The pieces of a database system	123
How the pieces fit together	125
Multi-tier computing architecture	128
Using multiple databases	129
Application programming interfaces	131
Inside Adaptive Server Anywhere	136

---

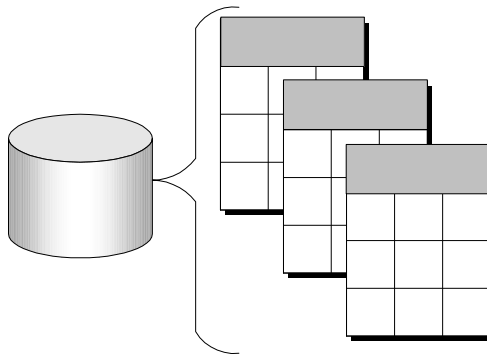
# Relational database concepts

A **relational database-management system** (RDBMS) is a system for storing and retrieving data, in which the data is organized into interrelated tables.

SQL Anywhere Studio provides two relational database systems. **Adaptive Server Anywhere** is the primary, full featured RDBMS, with a multitude of uses, from a network database server hosting many clients to a compact embedded database. UltraLite is a small-footprint relational database. The **UltraLite** deployment technology allows you to use Adaptive Server Anywhere features on even the smallest of devices.

## Database tables

In a relational database, all data is held in **tables**, which are made up of **rows** and **columns**.



Each table has one or more columns, and each column is assigned a specific data type, such as an integer, a sequence of characters (for text), or a date. Each row in the table has a single value for each column.

For example, a table containing employee information may look as follows:

emp_ID	emp_lname	emp_fname	emp_phone
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

### Characteristics of relational tables

The tables of a relational database have some important characteristics:

- ◆ There is no significance to the order of the columns or rows.
- ◆ Each row contains one and only one value for each column, or contains NULL, which indicates that there is no value for that column.

- ◆ All values for a given column have the same data type.

The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their equivalent in non-relational databases. This manual uses the informal terms.

Informal relational term	Formal relational term	Non-relational term
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record

What do you keep in each table?

Each table in the database should hold information about a specific thing, such as employees, products, or customers.

By designing a database this way, you can set up a structure that eliminates redundancy and the possible inconsistencies caused by redundancy. For example, both the sales and accounts payable departments might enter and look up information about customers. In a relational database, the information about customers is stored only once, in a table that both departments can access.

☞ For more information about database design, see “[Designing Your Database](#)” [*ASA SQL User’s Guide*, page 3].

## Relations between tables

You use primary keys and foreign keys to describe relationships between the information in different tables. **Primary keys** identify each row in a table uniquely, and **foreign keys** define the relationships between rows in different tables.

Primary keys and foreign keys let you use relational databases to hold information in an efficient manner, without redundancy.

### Tables have a primary key

Each table in a relational database should have a **primary key**. The primary key is a column, or set of columns, that uniquely identifies each row. No two rows in a table may have the same primary key value.

Examples

In the sample database, the employee table stores information about employees. It has a primary key column named emp\_id, which holds a unique ID number assigned to each employee. A single column holding an ID number is a common way to assign primary keys, and has advantages over names and other identifiers that may not always be unique.

---

A more complex primary key can be seen in the `sales_order_items` table of the sample database. The table holds information about individual items on orders from the company, and has the following columns:

- ◆ **id** An order number, identifying the order the item is part of.
- ◆ **line\_id** A line number, identifying each item on any order.
- ◆ **prod\_id** A product ID, identifying the product being ordered.
- ◆ **quantity** A quantity, displaying how many items were ordered.
- ◆ **ship\_date** A ship date, displaying when the order was shipped.

A particular sales order item is identified by the order it is part of, and by a line number on the order. These two numbers are stored in the `id` and `line_id` columns. Items may share a single `id` value (corresponding to an order for more than one item) or they may share a `line_id` number (all first items on different orders have a `line_id` of 1). No two items share both values, and so the primary key is made up of these two columns.

## Tables are related by foreign keys

The information in one table is related to that in other tables by **foreign keys**.

### Example

The sample database has one table holding employee information and one table holding department information. The department table has the following columns:

- ◆ **dept\_id** An ID number for the department. This is the primary key for the table.
- ◆ **dept\_name** The name of the department.
- ◆ **dept\_head\_id** The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the employee table. Instead, the employee table contains a column holding a number that matches one of the `dept_id` values in the department column.

The `dept_id` column in the employee table is called a foreign key to the department table. A foreign key references a particular row in the table containing the corresponding primary key.

In this example, the employee table (which contains the foreign key in the relationship) is called the **foreign table** or **referencing table**. The department table (which contains the referenced primary key) is called the **primary table** or the **referenced table**.

## Other database objects

There is more to a relational database than simply a set of related tables. You will also find the following objects in a relational database:

- ◆ **Indexes** Indexes allow quick lookup of information. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

Indexes are an important design element for high performance. You must usually create indexes explicitly, but indexes for primary and foreign keys and for unique columns are created automatically. Once created, the use of indexes is transparent to the user.

- ◆ **Views** Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

The tables that actually hold the information are sometimes called **base tables** to distinguish them from views. A view is defined with a SQL query on base tables or other views.

- ◆ **Stored procedures and triggers** These are routines held in the database itself that act on the information in the database.

You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can take parameters. For example, you might create a stored procedure that returns the names of all customers who have spent more than the amount that you specify as a parameter in the call to the procedure.

A trigger is a special stored procedure that automatically fires whenever a user updates, deletes, or inserts data, depending on how you define the trigger. You associate a trigger with a table or columns within a table. Triggers are useful for automatically maintaining business rules in a database.

- ◆ **Users and groups** Each user of a database has a user ID and password. You can set permissions for each user so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups in order to make the administration of permissions easier.

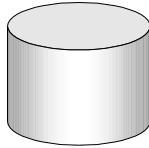
- 
- ◆ **Java objects** You can install Java classes into the database. Java classes provide a powerful way of building logic into your database, and a special class of user-defined data types for holding information.

All of these items together make up a **relational database-management system** (RDBMS); a system for storing and retrieving data, in which the data is organized into interrelated tables.

## The pieces of a database system

Relational database management systems contain the following pieces:

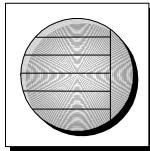
**A database** Data is stored in a database. In diagrams in the documentation, a database is indicated by a cylinder:



An Adaptive Server Anywhere database is a file, usually with an extension of *.db*. Adaptive Server Anywhere includes a sample database for you to work with: this is the file *asademo.db* in your Adaptive Server Anywhere installation directory.

**A database server** The database server manages the database. No other application addresses the database file directly; they all communicate with the database server.

In the documentation, a database server is indicated as follows:



Adaptive Server Anywhere provides two versions of its database server: the **personal database server** and the **network database server**. In addition to the features of the personal server, the network database server also supports client/server communications across a network, while the personal database server can accept connections only from applications running on the same machine. The request-processing engine is identical in both servers.

**A programming interface** Applications communicate with the database server using a programming interface. You can use ODBC, JDBC, OLE DB, Sybase Open Client, or Embedded SQL.

Many application development tools provide their own programming environment that hides the details of the underlying interface. For example, if you develop an application using Sybase PowerBuilder, you never have to make an ODBC function call. Nevertheless, behind the scenes each of these tools is using one of the programming interfaces.

The programming interface provides a library of function calls for communicating with the database. For ODBC and JDBC, the library is

---

commonly called a **driver**. The library is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on PC operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

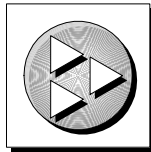
In diagrams in the documentation, a programming interface is indicated as follows:



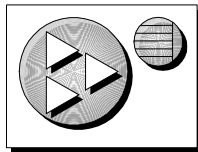
**A client application** Client applications use one of the programming interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported interfaces.

In diagrams in the documentation, a client application is indicated by the following icon:



UltraLite database servers are custom-generated for each UltraLite application, and are part of the application itself. An UltraLite application together with its database server is indicated as follows:





## How the pieces fit together

Database applications can connect to a database server located on the same machine as the application itself, or in the case of the network database server, on a different machine. In addition, with Adaptive Server Anywhere you can build distributed databases, with physically distinct databases on different machines sharing data.

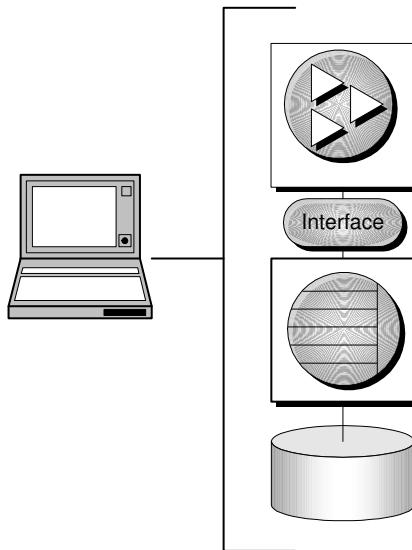
### Personal applications and embedded databases

You can use Adaptive Server Anywhere to build a complete application and database on a single computer. In the simplest arrangement, this is a **standalone application** or **personal application**: it is self-contained with no connection to other databases. In this case, the database server and the database can be started by the client application, and it is common to refer to the database as an **embedded database**. As far as the end user is concerned, the database is a part of the application. When a database server is used as an embedded database, it is sometimes called a **database engine**.

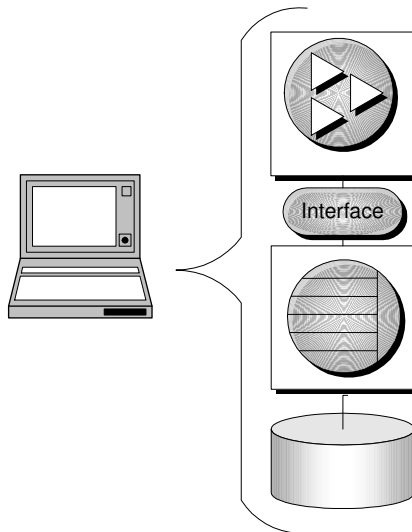
Many relational database management systems require experienced staff for administration. A characteristic of embedded databases is the ability to run entirely without administration.

The Adaptive Server Anywhere personal database server is generally used for standalone applications. A client application connects through a programming interface to a database server running on the same machine:

Standalone applications have the following architecture, with a client application connecting through a programming interface to a database server running on the same machine:



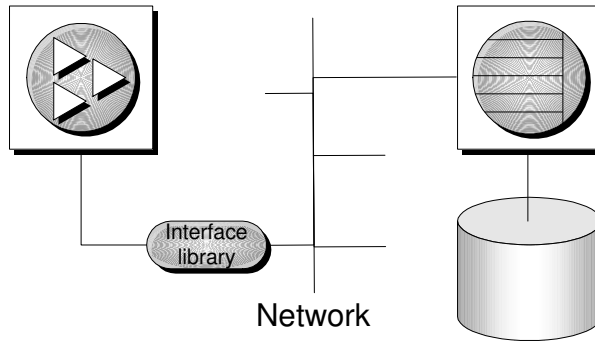
The Adaptive Server Anywhere personal database server is generally used for standalone applications, although you can also use applications on the same machine as the network server.



## Client/server applications and multi-user databases

You can use Adaptive Server Anywhere to build an installation with many applications running on different machines, connected over a network to a single database server running on a separate machine. This is a **client/server**

or **multi-user database** environment, and has the following architecture. The interface library is located on each client machine.



In this case, the database server is the Adaptive Server Anywhere network database server, which supports network communications.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

For a client application to work in a client/server environment, you need only identify the server to which it should connect.

---

## Multi-tier computing architecture

In multi-tier computing, application logic is held in an application server, such as Sybase EAServer, which sits between the database server and the client applications. In many situations, a single application server may access multiple databases in addition to non-relational data stores. In the Internet case, client applications are browser-based, and the application server is generally a Web server extension.

Sybase EAServer stores application logic in the form of components, and makes these components available to client applications. The components may be PowerBuilder components, Java beans, or COM components.

Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. Adaptive Server Anywhere is well-suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator. Both Sybase Enterprise Application Server and Microsoft Transaction Server use DTC to provide transaction services to their client applications.

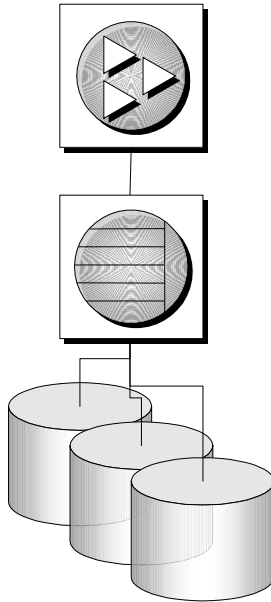
☞ For more information, see [“Three-Tier Computing and Distributed Transactions”](#) [*ASA Programming Guide*, page 507].

## Using multiple databases

This section describes extensions to the Adaptive Server Anywhere architecture described above for the case where you want to use more than one database.

### Running multiple databases on a single database server

Both the Adaptive Server Anywhere personal database server and the network database server can manage several databases simultaneously. Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, or by connecting to a database using the DatabaseFile connection parameter.

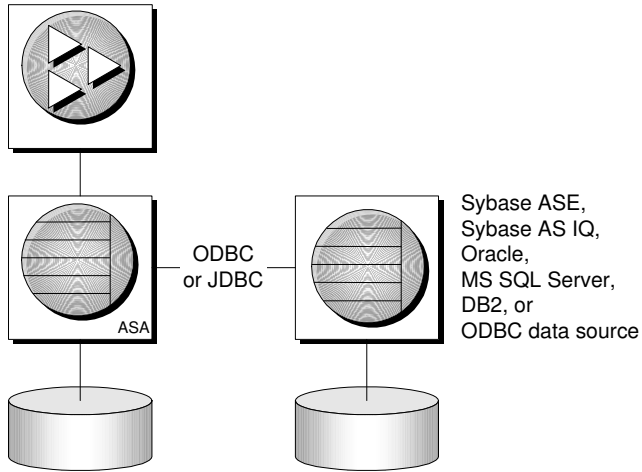
☞ For more information, see [“The database server”](#) [*ASA Database Administration Guide*, page 116], and [“DatabaseFile connection parameter \[DBF\]”](#) [*ASA Database Administration Guide*, page 185].

### Accessing data in other databases

You can access databases on multiple database servers, or even on the same

---

server, using the Adaptive Server Anywhere Remote Data Access features. The application is still connected to a single database as in the architecture diagrams above, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected.



☞ For more information, see [“Accessing Remote Data” \[ASA SQL User’s Guide, page 601\]](#).

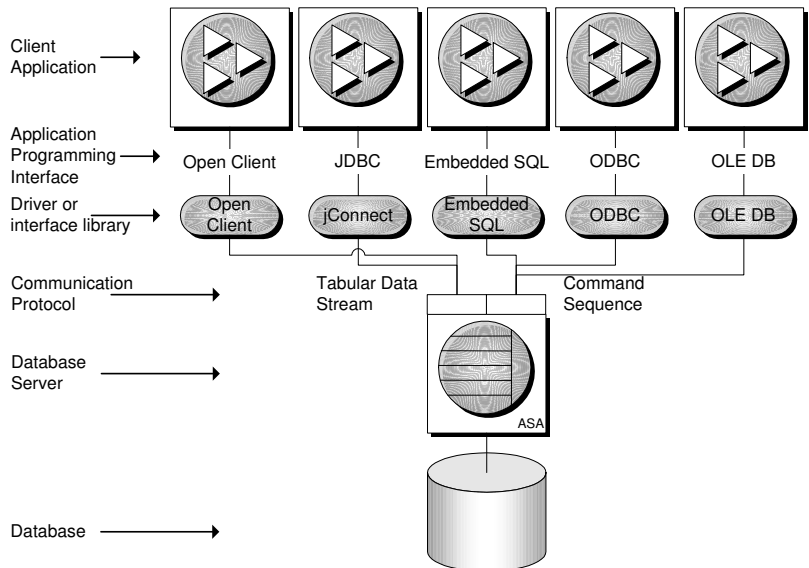
## Application programming interfaces

The current section describes application architecture in more detail. An overview of database application architecture was given in “[How the pieces fit together](#)” on page 125.

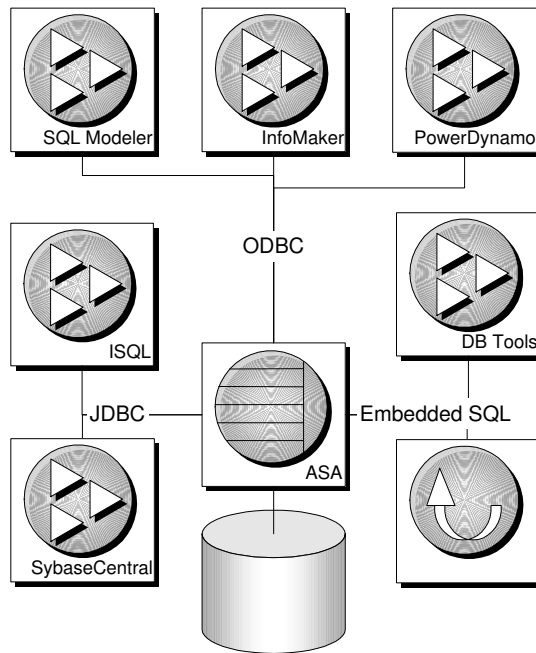
Adaptive Server Anywhere supports a wide set of programming interfaces to provide flexibility in the kinds of applications and application development environments you can use.

### Supported interfaces and protocols

The following diagram displays the supported interfaces, and the interface libraries used. In most cases, the interface library is the same name as the interface itself.



The applications supplied with SQL Anywhere Studio use several of these interfaces:



## Communications protocols

Each interface library communicates with the database server using a **communication protocol**. Adaptive Server Anywhere supports two communication protocols, **Tabular Data Stream (TDS)** and **Command Sequence**. These protocols are internal, and for most purposes it does not matter which one you are using. Your choice of development environment will be governed by your available tools.

The major differences you will see are upon connecting to the database. The Command Sequence applications and TDS applications use different methods to identify a database and database server, and so connection dialogs are different.

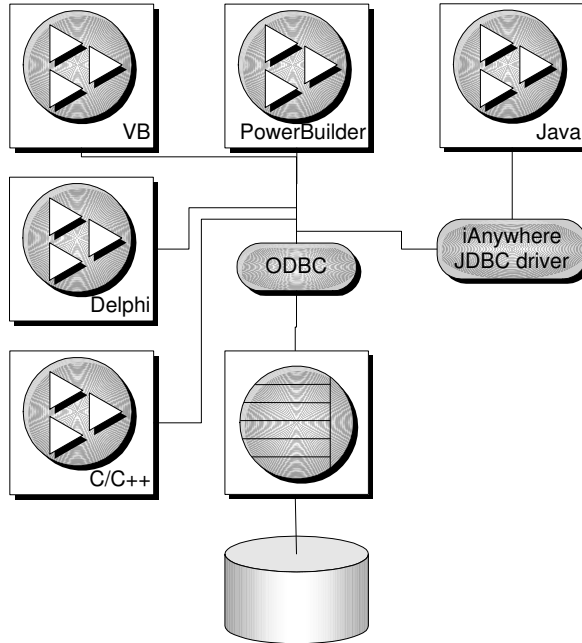
**TDS** This protocol is used by Sybase Adaptive Server Enterprise, Open Client applications, and Java applications that use the jConnect JDBC driver connect using TDS.

**Command Sequence** This protocol is used by Adaptive Server Anywhere, Adaptive Server IQ, and is used by embedded SQL, ODBC, and OLE DB applications.



## ODBC applications

You can develop ODBC applications using a variety of development tools and programming languages, as shown in the figure below.

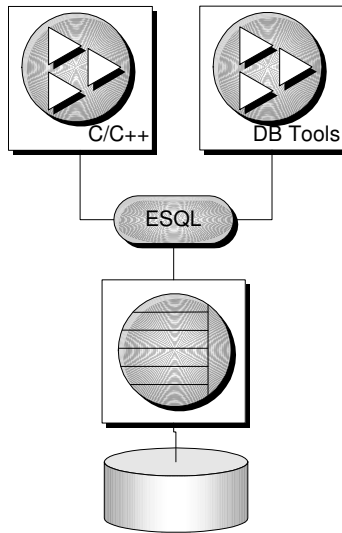


For example, of the applications supplied with SQL Anywhere Studio, InfoMaker and SQL Modeler use ODBC to connect to the database.

## Embedded SQL applications

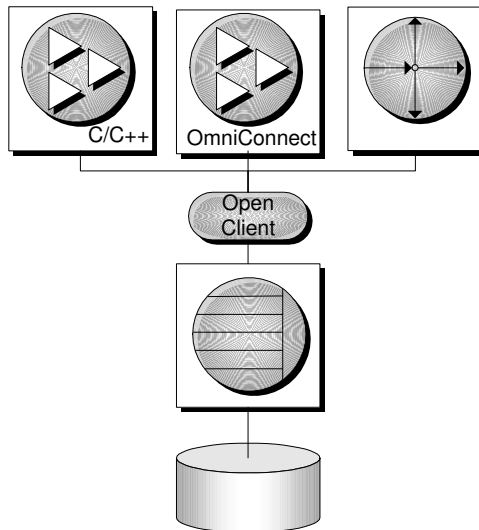
You can develop C or C++ applications using the embedded SQL programming interface. The command-line database tools are examples of applications developed in this manner.

Embedded SQL is also a programming interface for UltraLite applications.



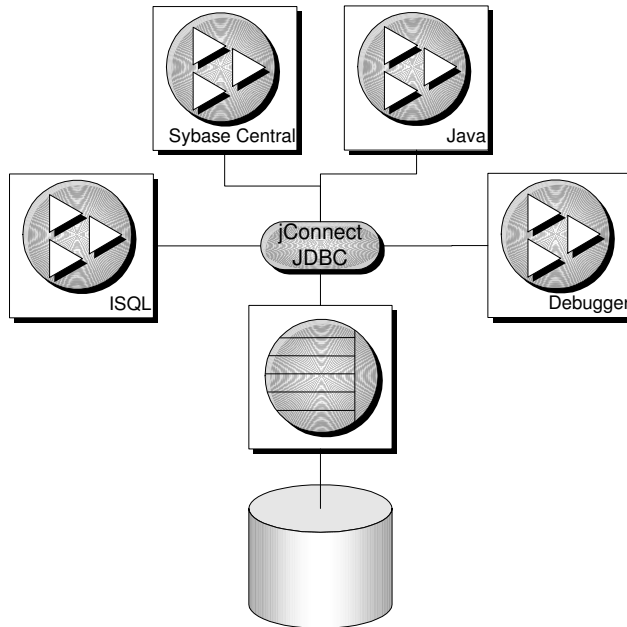
## Open Client applications

Open Client is an interface that is also supported by Sybase Adaptive Server Enterprise. You can develop Open Client applications in C or C++. Other Sybase applications, such as OmniConnect or Replication Server, use Open Client.



## JDBC applications

You can develop applications that use JDBC to connect to Adaptive Server Anywhere using Java. Several of the applications supplied with SQL Anywhere Studio use JDBC: the stored procedure debugger, Sybase Central, and Interactive SQL.



Java and JDBC are also important programming languages for developing UltraLite applications.

## OLE DB applications

Adaptive Server Anywhere includes an OLE DB driver. You can develop applications using Microsoft's OLE DB interface directly, or using the ActiveX Data Objects (ADO) interface. The ADO interface is included in Visual Basic and other Microsoft programming tools. SQL Anywhere Studio also includes an ADO.NET data provider for Adaptive Server Anywhere.

---

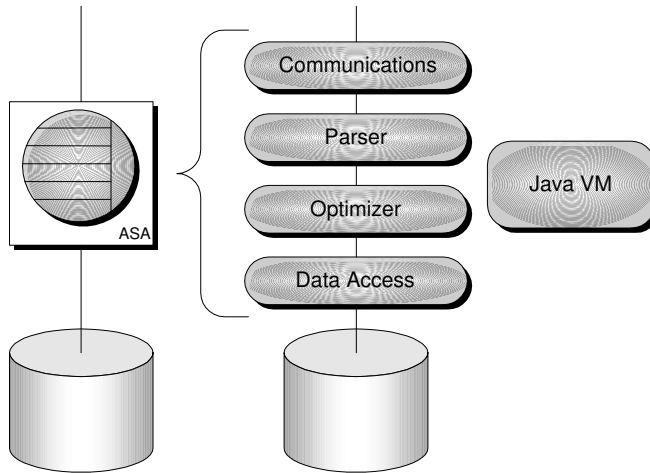
## Inside Adaptive Server Anywhere

While you never need to deal with the internals of the database server, a glimpse behind the scenes may help you understand the process better.

### Inside the database server

The Adaptive Server Anywhere database server has an internal structure that allows many requests to be handled efficiently.

- ◆ A communications layer handles the actual exchange of data with client applications. This layer receives requests from client applications, and returns results. The timing of these actions is governed by a negotiation between client and server to make sure that the network traffic is kept to a minimum, but that the data is made available as soon as possible on the client side.
- ◆ The parser checks each SQL statement sent to the database server, and transforms it into an internal form for processing.
- ◆ If the request is a query, an update, or delete statement, there may be many different ways of accessing the data, which may take massively different times. The job of the optimizer is to select from among all these possibilities the best route to getting the required data quickly.
- ◆ A Java Virtual Machine is built into the database server, and any Java operations sent by client applications, or invoked internally by the database server, are handled by the Java VM.
- ◆ The lowest level of the database server is concerned with reading and writing data from the disk, caching data in memory to avoid unnecessary disk access, and balancing the demands of different users.



## Inside the database

All the information in an Adaptive Server Anywhere database is usually stored in a single database file, which can be copied from one machine to another. It is possible to make databases of several files, but this is generally only required for very large databases.

In addition to the database file, Adaptive Server Anywhere uses two other files when it is running a database. These are the transaction log and the temporary file.

- ◆ **The database file** Internally, the database file is composed of pages: fixed size areas of disk. The data access layer reads and writes data one page at a time. Many pages hold the data that is in the database tables, but other pages hold index information, information about the distribution of data within the database, and so on.
- ◆ **The transaction log** The transaction log is a separate file that contains a record of all the operations performed on the database. Normally, it has the same name as the database file, except that it ends with the suffix *.log* instead of *.db*. It has three important functions.
  - **Record operations on your data to enable recovery** You can recreate your database from a backup together with the transaction log if the database file is damaged.
  - **Improve performance** By writing information to the transaction log, the database server can safely process your statements without writing to the database file as frequently.
  - **Enable database replication** SQL Remote and the MobiLink client

---

utility use this file to replicate the changes to your database on portable computers which are occasionally connected to the network.

- ◆ **The temporary file** The temporary file is opened when the database server starts, and is closed down when the server stops. As its name suggests, the temporary file is used while the server is running to hold temporary information. The temporary file does not hold information that needs to be kept between sessions.

The temporary file is stored in your temporary directory. The location of this directory is generally identified by your TEMP environment variable.

---

## CHAPTER 7

# Selecting Data from Database Tables

### About this chapter

This chapter introduces Structured Query Language (SQL) and queries, which retrieve data from a database. It describes how to retrieve data from a single table.

Other parts of the documentation set assume that you are familiar with the concepts introduced in this chapter.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">SQL and database computing</a>	140
<a href="#">SELECT statement</a>	142
<a href="#">Selecting a complete table</a>	143
<a href="#">Selecting columns from a table</a>	145
<a href="#">Ordering query results</a>	148
<a href="#">Selecting rows from a table</a>	151

---

## SQL and database computing

When a client application wants to carry out a database task, such as retrieving information using a query or inserting a new row into a table, it does so using **Structured Query Language (SQL)** statements. SQL is a relational database language that has been standardized by the ANSI and ISO standards bodies.

Depending on how you develop a client application, SQL statements could be supplied in function calls from the programming language. Some application development tools provide user interfaces for building and generating SQL statements.

The programming interface delivers the SQL statement to the database server. The database server receives the statement and executes it, returning the required information (such as query results) back to the application.

Client/server communication protocols carry information between the client application and the database server, and programming interfaces define how an application sends the information. No matter what interface you use, and what network protocol you use, it is SQL statements that are sent to a server, and the results of SQL statements that are returned to the client application.

Example

This SQL statement extracts the last names of all employees from the employee table in the sample database:

```
SELECT emp_lname
FROM employee
```

You can send queries like this to the database server using Interactive SQL or you can build the query into your own application.

Example

This SQL statement creates a table called food that lists types of food and the amount in stock in the employee cafeteria:

```
CREATE TABLE food (
    food_id integer primary key,
    food_type char(20),
    quantity integer
)
```

## Queries

The “Q” in “SQL” stands for **query**. You query (or **retrieve**) data from a database with the SELECT statement. The basic query operations in a relational system are projection, restriction, and join. The SELECT



statement implements all of them.

#### Projections and restrictions

A **projection** is a subset of the columns in a table. A **restriction** (also called **selection**) is a subset of the rows in a table, based on some conditions.

For example, the following SELECT statement retrieves the names and prices of all products that cost more than \$15:

```
SELECT name, unit_price
FROM product
WHERE unit_price > 15
```

This query uses both a projection (SELECT name, unit\_price) and a restriction (WHERE unit\_price > 15).

#### Joins

A join links the rows in two or more tables by comparing the values in columns of each table. For example, you might want to select the order item identification numbers and product names for all order items that shipped more than a dozen pieces of merchandise:

```
SELECT sales_order_items.id, product.name
FROM product JOIN sales_order_items
WHERE sales_order_items.quantity > 12
```

The product table and the sales\_order\_items table are joined together based on the foreign key relationship between them.

## Other SQL statements

You can do more with SQL than just query. SQL includes statements that create tables, views, and other database objects. It also includes statements that modify tables (the INSERT, UPDATE, and DELETE statements), and statements that perform many other database tasks discussed in this manual.

## The system tables

Every Adaptive Server Anywhere database contains a set of system tables. These are special tables that the system uses to manage data and the system. These tables are also sometimes called the **data dictionary** or the **system catalog**.

System tables contain information about the database. You never alter the system tables directly in the way that you can alter other tables. The system tables hold information about the tables in a database, the users of a database, the columns in each table, and so on. This information is data about data, or **metadata**.

☞ For more information about the Adaptive Server Anywhere system tables, see “System Tables” [[ASA SQL Reference, page 671](#)].

---

## SELECT statement

All interaction between applications (clients) and database servers is carried out by sending SQL statements to the database server, which returns information to the client.

The SELECT statement retrieves information from a database for use by the client application. SELECT statements are also called **queries**. The information is delivered to the client in the form of a result set. The client application can then process the result set. For example, Interactive SQL displays the result set in the Results pane. Result sets consist of a set of rows, just like tables in the database.

SELECT statements can become highly complex in applications retrieving very specific information from many tables. This chapter uses only simple SELECT statements: more advanced queries are described in later tutorials.

☞ For the full syntax of the SELECT statement, see “[SELECT statement](#)” [*ASA SQL Reference*, page 597].

### Notes

You should run the Adaptive Server Anywhere software on your computer while you read and work through the examples in this chapter.

Each example instructs you to type commands into Interactive SQL and describes what you will see on your computer screen. If you cannot run the software as you read the tutorials, you will still be able to learn about SQL, but you will not have the opportunity to experiment on your own.

The examples assume that you have started Interactive SQL and are connected to the sample database.

## Selecting a complete table

The simplest SELECT statement retrieves all the data in a single table. This SELECT statement has the following syntax:

```
SELECT * FROM table-name
```

where *table-name* should be replaced with the name of the table you are querying. The asterisk (\*) is a short form for a list of all columns.

### ❖ List all products sold by the company

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT * FROM Product
```

The SELECT statement retrieves all the rows and columns of the product table, and displays them on the Results tab in the Results pane:

id	name	description	size	color	quantity	unit_price
300	Tee Shirt	Tank Top	Small	White	28	9
301	Tee Shirt	V-neck	Medium	Orange	54	14
302	Tee Shirt	Crew Neck	One size fits all	Black	75	14
400	Baseball Cap	Cotton Cap	One size fits all	Black	112	9
...	...	...	...	...	...	...

The Product table contains seven columns. Each column has a name, such as color or id. There is a row for each product that the company sells, and each row has a single value in each column. For example, the product with ID 301 is a Tee Shirt. It is a V-neck style in medium size, and is orange in color.

### Notes

- ◆ **Table names are case insensitive** The table name Product starts with an upper case P, even though the real table name is all lower case. Adaptive Server Anywhere databases can be created as case-sensitive or case-insensitive in their string comparisons, but are always case insensitive in their use of identifiers such as table names and column names.

☞ For information on creating databases, see “Creating a database” [ASA SQL User’s Guide, page 31], or “The Initialization utility” [ASA Database Administration Guide, page 530].

- 
- ◆ **SQL keywords are case insensitive** You can enter **select** or **Select** instead of **SELECT**. In the documentation, upper case letters are generally used for SQL keywords.
  - ◆ **Line breaks are not important** You can type the statements all on one line, or break them up by pressing Enter at the end of each line. Some SQL statements, such as the **SELECT** statement, consist of several parts, called **clauses**. In many examples, each clause is placed on a separate line for easier reading, so the statement in the example is commonly written as follows in the documentation:

```
SELECT *  
FROM product
```

- ◆ **Row order in the result set is insignificant** There is no guarantee of the order in which rows are returned from the database, and no meaning to the order. If you wish to retrieve rows in a particular order, you must specify the order in the query.

#### Exercise

Try querying other tables in the sample database, such as the `employee`, `customer`, `contact`, or `sales_order` tables.

## Selecting columns from a table

You can limit the columns that a `SELECT` statement retrieves by listing the desired columns immediately after the `SELECT` keyword. This `SELECT` statement has the following syntax:

```
SELECT column-name-1, column-name-2,...
FROM table-name
```

where *column-name-1*, *column-name-2*, and *table-name* should be replaced with the names of the desired columns and table you are querying.

The list of result-set columns is called the **select list**. It is separated by commas. There is no comma after the last column in the list, or if there is only one column in the list. Limiting the columns in this way is sometimes called a **projection**.

### ❖ List the name, description, and color of each product

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT name, description, color
FROM product
```

<b>name</b>	<b>description</b>	<b>color</b>
Tee Shirt	Tank Top	White
Tee Shirt	V-neck	Orange
Tee Shirt	Crew Neck	Black
Baseball Cap	Cotton Cap	Black
...	...	...

#### Rearranging columns

The columns appear in the order in which you type them in the `SELECT` statement. If you want to rearrange the columns, simply change the order of the column names in the statement. For example, to put the **description** column on the left, use the following statement:

```
SELECT description, name, color
FROM product
```

## Using calculated columns

The columns of the result set do not need to be just columns in tables. They can also be expressions calculated from the underlying data. You can

---

combine table columns into a single result-set column, and you can use a wide variety of functions and operators to control the results you get.

❖ **List the value in stock of each product**

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT id, ( unit_price * quantity ) AS value
FROM product
```

id	value
300	252
301	756
302	1050
400	1008
...	...

Notes

- ◆ **Columns can be given an alias** By default the column name is the expression listed in the select list, but for calculated columns the expression is cumbersome and not very informative. Here the calculated column is renamed in the select list as value. value is the **alias** for the column.
- ◆ **Other operators are available** In the above example, the multiplication operator is used to combine the columns. You can use other operators, including the standard arithmetic operators as well as logical operators and string operators.

For example, the following query lists the full names of all customers:

```
SELECT id, (fname || ' ' || lname ) AS "Full name"
FROM customer
```

The || operator concatenates strings. In this query, the alias for the column has spaces, and so must be surrounded by double quotes. This rule applies not only to column aliases, but to table names and other identifiers in the database.

☞ For a complete list of operators, see “Operators” [ASA SQL Reference, page 11].

- ◆ **Functions can be used** In addition to combining columns, you can use a wide range of built-in functions to produce the results you want.

For example, the following query lists the product names in upper case:

```
SELECT id, UCASE( name )  
FROM product
```

<b>id</b>	<b>UCASE(product.name)</b>
300	TEE SHIRT
301	TEE SHIRT
302	TEE SHIRT
400	BASEBALL CAP
...	...

☞ For a complete list of functions, see [“Alphabetical list of functions”](#) [ASA SQL Reference, page 106].

---

## Ordering query results

Unless otherwise requested, the database server returns the rows of a table in an order that has no meaning. Often it is useful to look at the rows in a table in a more meaningful sequence. For example, you might like to see products in alphabetical order.

You order the rows in a result set by an `ORDER BY` clause to the end of the `SELECT` statement. This `SELECT` statement has the following syntax:

```
SELECT column-name-1, column-name-2,...
FROM table-name
ORDER BY order-by-column-name
```

where *column-name-1*, *column-name-2*, and *table-name* should be replaced with the names of the desired columns and table you are querying, and where *order-by-column-name* is a column in the table. As before, you can use the asterisk as a short form for all the columns in the table.

### ❖ List the products in alphabetical order

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT id, name, description
FROM product
ORDER BY name
```

id	name	description
400	Baseball Cap	Cotton Cap
401	Baseball Cap	Wool cap
700	Shorts	Cotton Shorts
600	Sweatshirt	Hooded Sweatshirt
...	...	...

Notes

- ◆ **The order of clauses is important** The `ORDER BY` clause must follow the `FROM` clause and the `SELECT` clause.
- ◆ **You can specify either ascending or descending order** The default order is ascending. You can specify a descending order by adding the keyword `DESC` to the end of the clause, as in the following query:

```
SELECT id, quantity
FROM product
ORDER BY quantity DESC
```



id	quantity
400	112
700	80
302	75
301	54
600	39
...	...

- ◆ **You can order by several columns** The following query sorts first by size (alphabetically), and then by name:

```
SELECT id, name, size
FROM product
ORDER BY size, name
```

id	name	size
600	Sweatshirt	Large
601	Sweatshirt	Large
700	Shorts	Medium
301	Tee Shirt	Medium
...	...	...

- ◆ **The ORDER BY column does not need to be in the select list** The following query sorts products by unit price, even though the price is not included in the result set

```
SELECT id, name, size
FROM product
ORDER BY unit_price
```

id	name	size
500	Visor	One size fits all
501	Visor	One size fits all
300	Tee Shirt	Small
400	Baseball Cap	One size fits all
...	...	...

- ◆ **If you do not use an ORDER BY clause, and you execute a query**

---

**more than once, you may appear to get different results** This is because Adaptive Server Anywhere may return the same result set in a different order. In the absence of an ORDER BY clause, Adaptive Server Anywhere returns rows in whatever order is most efficient. This means the appearance of result sets may vary depending on when you last accessed the row and other factors. The only way to ensure that rows are returned in a particular order is to use ORDER BY.

## Using indexes to improve ORDER BY performance

Sometimes there is more than one possible way for the Adaptive Server Anywhere database server to execute a query with an ORDER BY clause. You can use indexes to enable the database server to search the tables more efficiently.

Queries with WHERE and ORDER BY clauses

An example of a query that can be executed in more than one possible way is one that has both a WHERE clause and an ORDER BY clause.

```
SELECT *  
FROM customer  
WHERE id > 300  
ORDER BY company_name
```

In this example, Adaptive Server Anywhere must decide between two strategies:

1. Go through the entire customer table in order by company name, checking each row to see if the customer id is greater than 300.
2. Use the key on the id column to read only the companies with id greater than 300. The results would then need to be sorted by company name.

If there are very few id values greater than 300, the second strategy is better because only a few rows are scanned and quickly sorted. If most of the id values are greater than 300, the first strategy is much better because no sorting is necessary.

Solving the problem

Creating a two-column index on id and company\_name could solve the example above. Adaptive Server Anywhere can use this index to select rows from the table in the correct order. However, keep in mind that indexes take up space in the database file and involve some overhead to keep up to date. Do not create indexes indiscriminately.

👉 For more information, see [“Using indexes” \[ASA SQL User’s Guide, page 167\]](#).

## Selecting rows from a table

You can limit the rows that a `SELECT` statement retrieves by adding a `WHERE` clause to your query. This is sometimes called applying a **restriction** to the result set. The `WHERE` clause includes a **search condition** that specifies the rows to be returned. This `SELECT` statement has the following syntax:

```
SELECT column-name-1, column-name-2,...
FROM table-name
WHERE search-condition
```

where, as before, *column-name-1*, *column-name-2*, and *table-name* should be replaced with the names of the desired columns and table you are querying. The *search-condition* is described more below. If you use an `ORDER BY` clause, it must come after the `WHERE` clause.

### ❖ List all products colored black

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT *
FROM product
WHERE color = 'black'
```

id	name	description	size	color	quantity	unit_price
302	Tee Shirt	Crew Neck	One size fits all	Black	75	14
400	Baseball Cap	Cotton Cap	One size fits all	Black	112	9
501	Visor	Plastic Visor	One size fits all	Black	28	7
...	...	...	...	...	...	...

### Notes

- ◆ The `WHERE` clause includes a search condition to select rows. In this case the search condition is `color = 'black'`. Other search conditions are described in the following sections.

☞ For information on search conditions, see [“Search conditions” \[ASA SQL Reference, page 23\]](#).

- ◆ The single quotes around **black** are required. They indicate that **black** is a character string. Double quotes have a different meaning. Double quotes can be used to make otherwise invalid strings valid for column names and other identifiers.

☞ For information about strings, see [“Strings” \[ASA SQL Reference, page 9\]](#).

- ◆ The sample database is not case sensitive, so you would get the same results whether you searched for **BLACK**, **black**, or **Black**.
- ◆ How you order clauses is important. The **SELECT** list is followed by the **FROM** clause, followed by the **WHERE** clause, and then the **ORDER BY** clause. Typing the clauses in a different order gives a syntax error.

### Exercise

Try some queries that combine what you have learned in this chapter. Here is one query that lists the names and birth dates of all employees named John.

```
SELECT ( emp_fname || ' ' || emp_lname) AS Name,
       birth_date
FROM employee
WHERE emp_fname = 'John'
ORDER BY birth_date
```

Name	birth_date
John Letiecq	4/27/1939
John Sheffield	9/25/1955

## Comparing dates in search conditions

You can use operators other than equals to select a set of rows that satisfy the search condition. The inequality operators (< and >) can be used to compare numbers, dates, and even character strings.

### ❖ List all employees born before March 13, 1964

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date < 'March 13, 1964'
ORDER BY birth_date DESC
```

emp_lname	birth_date
Ahmed	12/12/1963
Dill	7/19/1963
Rebeiro	4/12/1963
Garcia	3/3/1963
Pastor	7/14/1962
...	...

- ◆ **Automatic conversion to dates** The Adaptive Server Anywhere database server knows that the `birth_date` column contains dates, and automatically converts the string `'March 13, 1964'` to a date.
- ◆ **Ways of specifying dates** There are many ways of specifying dates. The following are all accepted by Adaptive Server Anywhere:

```
'March 13, 1964'
'1964/03/13'
'1964-03-13'
```

You can tune the interpretation of dates in queries by setting a database option. Dates in the format `yyyy/mm/dd` or `yyyy-mm-dd` are always recognized unambiguously as dates, regardless of the `DATE_ORDER` setting.

☞ For information on controlling allowable date formats in queries, see [“DATE\\_ORDER option \[compatibility\]”](#) [*ASA Database Administration Guide*, page 648], and [“Setting options”](#) [*ASA Database Administration Guide*, page 614].

- ◆ **Other comparison operators** For a complete list of available comparison operators, see [“Comparison operators”](#) [*ASA SQL Reference*, page 11].

## Pattern matching in search conditions

Pattern matching is a versatile way of identifying character data. In SQL, the `LIKE` keyword is used to search for patterns. Pattern matching employs wildcard characters to match different combinations of characters.

### ❖ List all employees whose last name begins with BR

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT emp_lname, emp_fname
FROM employee
WHERE emp_lname LIKE 'br%'
```

emp_lname	emp_fname
Breault	Robert
Braun	Jane

The `%` in the search condition indicates that any number of other characters may follow the letters `BR`.

---

❖ **List all employees whose last name begins with BR, followed by zero or more letters and a T, followed by zero or more letters**

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT emp_lname, emp_fname  
FROM employee  
WHERE emp_lname LIKE 'BR%T%'
```

emp_lname	emp_fname
Breault	Robert

The first % sign matches the string **ea**ul, while the second % sign matches the empty string (no characters).

Another special character that can be used with LIKE is the \_ (underscore) character, which matches exactly one character. For example, the pattern 'BR\_U%' matches all names starting with BR and having U as the fourth letter. In Braun the \_ character matches the letter A and the % matches N.

☞ For more information, see “LIKE conditions” [*ASA SQL Reference*, page 25].

## Matching rows by sound

With the SOUNDEX function, you can match rows by sound. For example, suppose a phone message was left for a name that sounded like “Ms. Brown”. Which employees in the company have names that sound like Brown?

❖ **List employees with a last name that sound like Brown**

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT emp_lname, emp_fname  
FROM employee  
WHERE SOUNDEX( emp_lname ) = SOUNDEX( 'Brown' )
```

emp_lname	emp_fname
Braun	Jane

The algorithm used by SOUNDEX makes it useful mainly for English-language databases.

☞ For more information, see “SOUNDEX function [String]” [*ASA SQL Reference*, page 225].

## Using compound search conditions

Search conditions can be combined using the AND and OR logical operators to make compound search conditions. Each individual piece of the search condition is sometimes called a **predicate**.

### ❖ List all employees born before March 13, 1964, except the employee named Whitney

1. In Interactive SQL, type the following in the SQL Statements pane:

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date < '1964-3-13'
AND emp_lname <> 'whitney'
```

emp_lname	birth_date
Cobb	12/4/1960
Jordan	12/13/1951
Breault	5/13/1947
Espinoza	12/14/1939
Dill	7/19/1963
Francis	9/12/1954

## Shortcuts for compound search conditions

Using the short form  
BETWEEN

SQL has two short forms for compound search conditions. The first, BETWEEN, is used when you are looking for a range of values. For example the following two queries are equivalent:

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date BETWEEN '1963-1-1' AND '1965-3-31'
```

and

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date >= '1963-1-1'
AND birth_date <= '1965-3-31'
```

Using the short form IN

The second short form, IN, may be used when looking for one of a number of values. The following two statements are equivalent:

---

```
SELECT emp_lname, emp_id
FROM employee
WHERE emp_lname IN ('yeung','bucceri','charlton')
```

and

```
SELECT emp_lname, emp_id
FROM employee
WHERE emp_lname = 'yeung'
OR emp_lname = 'bucceri'
OR emp_lname = 'charlton'
```



---

## CHAPTER 8

# Selecting Data from Multiple Tables

### About this chapter

This chapter describes database queries that look at information in more than one table. To do this, SQL provides the **JOIN** operator. There are several different ways to join tables together in queries, and this chapter describes some of the more important ones.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	158
<a href="#">Joining tables using the cross product</a>	159
<a href="#">Using the ON phrase to restrict a join</a>	160
<a href="#">Joining tables using key joins</a>	162
<a href="#">Joining tables using natural joins</a>	164
<a href="#">Joining tables using outer joins</a>	166

---

# Introduction

Sometimes it is necessary to view data from multiple related tables. This chapter explains how to use a **join** to view the data in a useful and meaningful way.

## Displaying a list of tables

In Interactive SQL, you can display a list of tables by pressing the F7 key.



Select a table and click Show Columns to see the columns for that table. The ESCAPE key takes you back to the table list, and pressing it again will take you back to the SQL Statements pane. Pressing ENTER instead of ESCAPE copies the selected table or column name into the SQL Statements pane at the current cursor position.

Press ESCAPE to leave the list.

☞ For a diagram of the tables in the sample database, see [“About the sample database” on page 198](#).

## Joining tables using the cross product

One of the tables in the sample database is `sales_order`, which lists the orders placed to the company. Each order has a `sales_rep` column, containing the employee ID of the sales representative responsible for the order. There are 648 rows in the `sales_order` table and 75 rows in the `employee` table.

The cross product join is a simple starting point for understanding joins, but not very useful in itself.

### ❖ List all data in the `employee` and `sales_order` tables

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT *  
FROM sales_order CROSS JOIN employee
```

The results of this query, which appear on the Results tab in the Interactive SQL Results pane, match every row in the `employee` table with every row in the `sales_order` table. Since there are 75 rows in the `employee` table and 648 rows in the `sales_order` table, there are  $75 \times 648 = 48,600$  rows in the result of the join. Each row consists of all columns from the `sales_order` table followed by all columns from the `employee` table. This join is called a **full cross product**.

Subsequent sections describe how to construct more selective joins. The more selective joins can be thought of as applying a restriction to the cross product table.

☞ For more information, see “Cross joins” [*ASA SQL User’s Guide*, page 274].

---

## Using the ON phrase to restrict a join

The ON phrase applies a restriction to the rows in a join, in much the same way that the WHERE clause applies restrictions to the rows of a query.

The ON phrase allows more useful joins than the CROSS JOIN to be constructed. For example, you can apply the ON phrase to a join of the sales\_order and employee table to retrieve only those rows for which the sales\_rep in the sales\_order table is the same as the one in the employee table in every row of the result. Then each row contains information about an order and the sales representative responsible for it.

### ❖ List all sales orders with their dates, and the employee responsible for each

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT employee.emp_lname,  
       sales_order.id,  
       sales_order.order_date  
FROM sales_order JOIN employee  
   ON sales_order.sales_rep = employee.emp_id
```

emp_lname	id	order_date
Chin	2008	4/2/2001
Chin	2020	3/4/2001
Chin	2032	7/5/2001
Chin	2044	7/15/2000
Chin	2056	4/15/2001
...	...	...

### Notes

The table name is given as a prefix to identify the columns. Using the table name prefix clarifies the statement, and is required when two tables have a column with the same name.

The results of this query contain only 648 rows (one for each row in the sales\_order table). Of the 48,600 rows in the cross product, only 648 of them have the employee number equal in the two tables.

The ordering of the results has no meaning. You could add an ORDER BY clause to impose a particular order on the query.

The ON clause includes columns that are not included in the final result set.

☞ For more information, see “Explicit join conditions (the ON phrase)” [ASA SQL User’s Guide, page 271].

---

## Joining tables using key joins

Many common joins are between two tables related by a foreign key. The most common join restricts foreign key values to be equal to primary key values.

The **KEY JOIN** operator joins two tables based on foreign key relationship. In other words, Adaptive Server Anywhere generates an **ON** clause that equates the primary key column from one table with the foreign key column of the other.

The example in the previous section restricts foreign key values in the `sales_order` table to be equal to the primary key values in the `employee` table.

```
SELECT employee.emp_lname,
       sales_order.id,
       sales_order.order_date
FROM sales_order JOIN employee
   ON sales_order.sales_rep = employee.emp_id
```

The query can be more simply expressed using a **KEY JOIN**:

```
SELECT employee.emp_lname,
       sales_order.id,
       sales_order.order_date
FROM sales_order KEY JOIN employee
```

**KEY JOIN** is just a shortcut for typing the **ON** clause; the two queries are identical. Key join is the default when you specify **JOIN**, but do not specify **CROSS**, **NATURAL**, **KEY**, or use an **ON** clause.

If you look at the diagram of the `employee` database, lines between tables represent foreign keys. You can use the **KEY JOIN** operator anywhere two tables are joined by a line in the diagram.

☞ For a diagram of the sample database, see [“About the sample database” on page 198](#).

### Joining more than two tables

Two or more tables can be joined using join operators. The following query uses four tables to list the total value of the orders placed by each customer. It connects the four tables `customer`, `sales_order`, `sales_order_items`, and `product` using the single foreign-key relationships between each pair of these tables.

## ❖ List companies and the total value of their orders

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT company_name,
       SUM( sales_order_items.quantity *
           product.unit_price) AS value
FROM ( ( customer KEY JOIN sales_order )
      KEY JOIN sales_order_items )
      KEY JOIN product
GROUP BY company_name
```

company_name	value
Bensoul's Boutique	1332
Bush Pro Shop	2940
Sterling & Co.	6804
Ocean Sports	3744
...	...

## Notes

Your result set may appear in a different order. There is no significance to the order of the rows in the result set.

The example uses the SUM operator, which is an aggregate function. Aggregate functions work with GROUP BY clauses to return values for each row group. In this example, the sum of `sales_order_items.quantity * product.unit_price`—that is, the total amount of money paid per product type—is calculated for each `company_name`, thereby returning each company's sales.

The parentheses in the FROM clause help to clarify the order in which the joins are made.

☞ For more information on complex key joins, see “Key joins” [*ASA SQL User's Guide*, page 294].

---

## Joining tables using natural joins

The `NATURAL JOIN` operator joins two tables based on common column names. In other words, Adaptive Server Anywhere generates an `ON` clause that equates the common columns from each table.

### ❖ List all employees and their departments

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT emp_lname, dept_name
FROM employee NATURAL JOIN department
```

emp_lname	dept_name
Whitney	R & D
Cobb	R & D
Breault	R & D
Shishov	R & D
Driscoll	R & D
...	...

Adaptive Server Anywhere looks at the two tables and determines that the only column name they have in common is `dept_id`. The following `ON CLAUSE` is internally generated and used to perform the join:

```
FROM employee JOIN department
ON employee.dept_id = department.dept_id
```

`NATURAL JOIN` is just a shortcut for typing the `ON` clause; the two queries are identical.

### Errors using `NATURAL JOIN`

This join operator can cause problems by equating columns you may not intend to be equated. For example, the following query generates unwanted results:

```
SELECT *
FROM sales_order NATURAL JOIN customer
```

The result of this query has no rows. Adaptive Server Anywhere internally generates the following `ON` clause:

```
FROM sales_order JOIN customer
ON sales_order.id = customer.id
```

The `id` column in the `sales_order` table is an ID number for the *order*. The `id`



column in the customer table is an ID number for the *customer*. None of them matched. Of course, even if a match were found, it would be a meaningless one.

☞ For more information, see “Natural joins” [[ASA SQL User’s Guide](#), page 290].

---

## Joining tables using outer joins

In the previous examples, you created joins that returned rows only if they satisfied the join conditions. These joins are called **inner joins**, and are the default. Sometimes, you may wish to preserve all the rows in one table. To do this, you use an **outer join**.

You can specify a **right outer join**, which preserves all the rows in the right table; a **left outer join**, which preserves the left table; or a **full outer join**, which preserves all the rows in both tables.

### ❖ List all customers, and the dates of any orders they have placed

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT lname, order_date, city
FROM customer KEY LEFT OUTER JOIN sales_order
WHERE customer.state = 'NY'
ORDER BY order_date
```

lname	order_date	city
Thompson	(NULL)	Bancroft
Reiser	1993-01-22	Rockwood
Clarke	1993-01-27	Rockwood
Mentary	1993-01-30	Rockland
...	...	...

The statement includes all customers, whether or not they have placed an order. If a particular customer has placed no orders, each column in the result that corresponds to order information contains NULL.

☞ For more information, see “Outer joins” [*ASA SQL User’s Guide*, page 276].

---

## CHAPTER 9

# Selecting Data Using Subqueries

About this chapter

This chapter shows how to use the results of one query as part of another `SELECT` statement. This is a useful tool in building more complex and informative queries.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introducing subqueries</a>	168
<a href="#">Single-row and multiple-row subqueries</a>	170
<a href="#">Using subqueries instead of joins</a>	172

---

## Introducing subqueries

A relational database allows you to store related data in more than one table. You can extract data from related tables using **subqueries**—queries that appear in another query’s WHERE clause or HAVING clause. Subqueries make some queries easier to write than joins, and there are queries that cannot be written without using subqueries.

Subqueries use the results of one query as part of another query. This section illustrates a situation where subqueries can be used by building a query that lists order items for products that are low in stock.

There are two queries involved in producing this list. This section first describes them separately, and then shows the single query that produces the same result.

### ❖ List all products for which there are less than 20 items in stock

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT id, description, quantity
FROM product
WHERE quantity < 20
```

id	description	quantity
401	Wool cap	12

The query shows that only wool caps are low in stock.

### ❖ List all order items for wool caps

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT *
FROM sales_order_items
WHERE prod_id = 401
ORDER BY ship_date DESC
```

id	line_id	prod_id	quantity	ship_date
2082	1	401	48	7/9/2001
2053	1	401	60	6/30/2001
2125	2	401	36	6/28/2001
2027	1	401	12	6/17/2001

id	line_id	prod_id	quantity	ship_date
...	...	...	...	...

This two-step process of identifying items low in stock and identifying orders for those items can be combined into a single query using subqueries.

### ❖ List all order items for items that are low in stock

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT *
FROM sales_order_items
WHERE prod_id IN
  ( SELECT id
    FROM product
    WHERE quantity < 20 )
ORDER BY ship_date DESC
```

id	line_id	prod_id	quantity	ship_date
2082	1	401	48	7/9/2001
2053	1	401	60	6/30/2001
2125	2	401	36	6/28/2001
2027	1	401	12	6/17/2001
...	...	...	...	...

The subquery in the statement is the phrase enclosed in parentheses:

```
( SELECT id
  FROM product
  WHERE quantity < 20 )
```

The subquery makes a list of all values in the id column in the product table, satisfying the WHERE clause search condition.

The subquery returns a set of rows, but only a single column. The IN keyword treats each value as a member of a set and tests whether each row in the main query is a member of the set.

---

## Single-row and multiple-row subqueries

There are constraints on the number of rows and columns that a subquery may return. If you use IN, ANY, or ALL, the subquery may return several rows, but only one column. If you use other operators, the subquery must return a single value.

A multiple-row subquery Two tables in the sample database are concerned with financial results. The fin\_code table is a small table holding the different codes for financial data and their meanings:

To list the revenue items from the fin\_data table, type the following:

```
SELECT *
FROM fin_data
WHERE fin_data.code IN
  ( SELECT fin_code.code
    FROM fin_code
    WHERE type = 'revenue' )
```

year	quarter	code	amount
1999	Q1	r1	1023
1999	Q2	r1	2033
1999	Q3	r1	2998
1999	Q4	r1	3014
2000	Q1	r1	3114

This example has used qualifiers to clearly identify the table to which the code column in each reference belongs. In this particular example, the qualifiers could have been omitted.

Two other keywords can be used as qualifiers for operators to allow them to work with multiple rows: ANY and ALL.

The following query is identical to the successful query above:

```
SELECT *
FROM fin_data
WHERE fin_data.code = ANY
  ( SELECT fin_code.code
    FROM fin_code
    WHERE type = 'revenue' )
```

While the =ANY condition is identical to the IN condition, ANY can also be used with inequalities such as < or > to give more flexible use of subqueries.

The ALL keyword is similar to the word ANY. For example, the following

query lists financial data that is not revenues:

```
SELECT *
FROM fin_data
WHERE fin_data.code <> ALL
  ( SELECT fin_code.code
    FROM fin_code
    WHERE type = 'revenue' )
```

This is equivalent to the following command using NOT IN:

```
SELECT *
FROM fin_data
WHERE fin_data.code NOT IN
  ( SELECT fin_code.code
    FROM fin_code
    WHERE type = 'revenue' )
```

A common error using subqueries

In general, subquery result sets are restricted to a single column. The following example does not make sense because Adaptive Server Anywhere would not know which column from `fin_code` to compare to the `fin_data.code` column.

```
-- this query is incorrect
SELECT *
FROM fin_data
WHERE fin_data.code IN
  ( SELECT fin_code.code, fin_code.type
    FROM fin_code
    WHERE type = 'revenue' )
```

Single-row subqueries

While subqueries used with an IN condition may return a set of rows, a subquery used with a comparison operator must return only one row. For example the following command results in an error since the subquery returns two rows:

```
-- this query is incorrect
SELECT *
FROM fin_data
WHERE fin_data.code =
  ( SELECT fin_code.code
    FROM fin_code
    WHERE type = 'revenue' )
```

---

## Using subqueries instead of joins

Suppose you need a chronological list of orders and the company that placed them, but would like the company name instead of their customer ID. You can get this result using a join as follows:

Using a join

To list the order id, date, and company name for each order since the beginning of 2001, type the following:

```
SELECT    sales_order.id,
          sales_order.order_date,
          customer.company_name
FROM sales_order
  KEY JOIN customer
WHERE order_date > '2001/01/01'
ORDER BY order_date
```

id	order_date	company_name
2473	1/4/2001	Peachtree Active Wear
2474	1/4/2001	Sampson & Sons
2106	1/5/2001	Salt & Pepper's
2475	1/5/2001	Cinnamon Rainbow's
2036	1/5/2001	Hermanns

Using a subquery

The following statement obtains the same results using a subquery instead of a join:

```
SELECT sales_order.id,
       sales_order.order_date,
       ( SELECT company_name FROM customer
         WHERE customer.id = sales_order.cust_id )
FROM sales_order
WHERE order_date > '2001/01/01'
ORDER BY order_date
```

The subquery refers to the `cust_id` column in the `sales_order` table even though the `sales_order` table is not part of the subquery. Instead, the `sales_order.cust_id` column refers to the `sales_order` table in the main body of the statement. This is called an **outer reference**. Any subquery that contains an outer reference is called a **correlated subquery**.

A subquery can be used instead of a join whenever only one column is required from the other table. (Recall that subqueries can only return one column.) In this example, you only needed the `company_name` column so the join could be changed into a subquery.



If the subquery might have no result, this method is called an outer join. The join in previous sections of the tutorial is more fully called an inner join.

### Using an outer join

To list all customers in Washington state, together with their most recent order ID, type the following:

```
SELECT    company_name, state,
          ( SELECT MAX( id )
            FROM sales_order
            WHERE sales_order.cust_id = customer.id )
FROM customer
WHERE state = 'WA'
```

company_name	state	MAX(sales_order.id)
Custom Designs	WA	2547
It's a Hit!	WA	(NULL)

The It's a Hit! company placed no orders, and the subquery returns NULL for this customer. Companies who have not placed an order are not listed when inner joins are used.

You could also specify an outer join explicitly. In this case, a GROUP BY clause is also required.

```
SELECT company_name, state,
       MAX( sales_order.id )
FROM customer
      KEY LEFT OUTER JOIN sales_order
WHERE state = 'WA'
GROUP BY company_name, state
```



---

## CHAPTER 10

# Selecting Aggregate Data

### About this chapter

This chapter describes how to construct queries that tell you aggregate information. Examples of aggregate information are as follows:

- ◆ The total of all values in a column.
- ◆ The number of distinct entries in a column.
- ◆ The average value of entries in a column.

### Contents

<b>Topic:</b>	<b>page</b>
Summarizing data	176
A first look at aggregate functions	177
Applying aggregate functions to grouped data	178
Restricting groups	180

---

## Summarizing data

Some queries examine aspects of the data in your table that reflect properties of groups of rows rather than of individual rows. For example, you may wish to find the average amount of money that a customer pays for an order, or to see how many employees work for each department. For these types of tasks, you use **aggregate** functions and the **GROUP BY** clause.

## A first look at aggregate functions

Aggregate functions return a single value for a set of rows. If there is no GROUP BY clause, an aggregate function returns a single value for all the rows that satisfy other aspects of the query.

### ❖ List the number of employees in the company

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT COUNT( * )
FROM employee
```

---

**COUNT(\*)**

75

The result set consists of only one column, with title **COUNT( \* )**, and one row, which contains the total number of employees.

### ❖ List the number of employees in the company and the birth dates of the oldest and youngest employee

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT COUNT(*), MIN(birth_date), MAX(birth_date)
FROM employee
```

COUNT(*)	MIN(employee.birth_date)	MAX(employee.birth_date)
75	i/1936	1/18/1973

The functions COUNT, MIN, and MAX are called **aggregate functions**, which are functions that summarize information. Other aggregate functions include statistical functions such as AVG, STDDEV, and VARIANCE. All but COUNT have the name of a column as a parameter.

☞ For more information, see “Aggregate functions” [ASA SQL Reference, page 92].

---

## Applying aggregate functions to grouped data

In addition to providing information about an entire table, aggregate functions can be used on groups of rows. The GROUP BY clause arranges rows into groups, and aggregate functions return a single value for each group of rows.

Example

### ❖ List the sales representatives and the number of orders each has taken

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT sales_rep, count( * )
FROM sales_order
GROUP BY sales_rep
ORDER BY sales_rep
```

sales_rep	count(*)
129	57
195	50
299	114
467	56
...	...

A GROUP BY clause tells Adaptive Server Anywhere to partition the set of all the rows that would otherwise be returned. All rows in each partition, or group, have the same values in the named column or columns. There is only one group for each unique value or set of values. In this case, all the rows in each group have the same sales\_rep value.

Aggregate functions such as COUNT are applied to the rows in each group. Thus, this result set displays the total number of rows in each group. The results of the query consist of one row for each sales rep ID number. Each row contains the sales rep ID, and the total number of sales orders for that sales representative.

Whenever GROUP BY is used, the resulting table has one row for each column or set of columns named in the GROUP BY clause.

☞ For more information, see [“The GROUP BY clause: organizing query results into groups” \[ASA SQL User’s Guide, page 243\]](#).

A common error with GROUP BY

A common error with GROUP BY is to try to get information that cannot

properly be put in a group. For example,

```
-- This query is incorrect
SELECT sales_rep, emp_lname, COUNT( * )
FROM sales_order KEY JOIN employee
GROUP BY sales_rep
```

gives the following error:

Function or column reference to 'emp\_lname' in the select list must also appear in a GROUP BY

An error is reported because Adaptive Server Anywhere cannot be sure that each of the result rows for an employee with a given ID all have the same last name.

To fix this error, add the column to the GROUP BY clause.

```
SELECT sales_rep, emp_lname, COUNT( * )
FROM sales_order KEY JOIN employee
GROUP BY sales_rep, emp_lname
ORDER BY sales_rep
```

If this is not appropriate, you can instead use an aggregate function to select only one value, as shown:

```
SELECT sales_rep, MAX( emp_lname ), COUNT( * )
FROM sales_order KEY JOIN employee
GROUP BY sales_rep
ORDER BY sales_rep
```

The MAX function chooses the maximum (last alphabetically) last name from the detail rows for each group. This statement is valid because there can be only one distinct maximum value. In this case, the same last name appears on every detail row within a group.

---

## Restricting groups

You have already seen how to restrict rows in a result set using the WHERE clause. You restrict the rows in groups using the HAVING clause.

### ❖ List all sales representatives with more than 55 orders

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
SELECT sales_rep, count( * ) AS orders
FROM sales_order KEY JOIN employee
GROUP BY sales_rep
HAVING count( * ) > 55
ORDER BY orders DESC
```

sales_rep	orders
299	114
129	57
1142	57
467	56

#### Order of clauses

A GROUP BY must always appear before a HAVING clause. If both are present, a WHERE clause must appear before a GROUP BY clause.

HAVING clauses and WHERE clauses can both be used in a single query. Conditions in the HAVING clause logically restrict the rows of the result only after the groups have been constructed. Criteria in the WHERE clause are logically evaluated before the groups are constructed, and so save time.

👉 For more information, see “[The HAVING clause: selecting groups of data](#)” [*ASA SQL User’s Guide*, page 248].

## Combining WHERE and HAVING clauses

Sometimes you can specify the same set of rows using either a WHERE clause or a HAVING clause. In such cases, one method is not more or less efficient than the other. The optimizer always automatically analyzes each statement you type and selects an efficient means of executing it. It is best to use the syntax that most clearly describes the desired result. In general, that means eliminating undesired rows in earlier clauses.

Example

To list all sales reps with more than 55 orders and an ID of more than 1000,



type the following statement.

```
SELECT sales_rep, count( * )
FROM sales_order KEY JOIN employee
WHERE sales_rep > 1000
GROUP BY sales_rep
HAVING count( * ) > 55
ORDER BY sales_rep
```

The following statement produces the same results.

```
SELECT sales_rep, count( * )
FROM sales_order KEY JOIN employee
GROUP BY sales_rep
HAVING count( * ) > 55 AND sales_rep > 1000
ORDER BY sales_rep
```

Adaptive Server Anywhere detects that both statements describe the same result set, and so executes each efficiently.



---

## CHAPTER 11

# Updating the Database

About this chapter

This chapter describes how to make changes to the contents of your database. It includes descriptions of how to add rows, remove rows, and change the contents of rows, as well as how to make changes permanent or correct changes<sup>4</sup> you have made.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Introduction</a>	184
<a href="#">Adding rows to a table</a>	185
<a href="#">Modifying rows in a table</a>	186
<a href="#">Deleting rows</a>	187
<a href="#">Grouping changes into transactions</a>	188
<a href="#">Integrity checking</a>	191

---

## Introduction

This chapter describes how to make changes to the data in your database. There are three basic operations:

- ◆ **Insert** You can add rows to tables to include new data.
- ◆ **Delete** You can delete rows in tables to remove data.
- ◆ **Update** You can modify existing rows in tables.

Each operation is carried out by executing a SQL statement.

## Adding rows to a table

Suppose that the company decides to sell a new product, a brown acrylic vest. This action requires you to add data to the product table of the sample database.

### ❖ Add a brown acrylic vest to the product table

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
INSERT
INTO product ( id, name, description, size, color,
               quantity, unit_price )
VALUES ( 800, 'Vest', 'Acrylic Vest', 'Extra Large',
        'Brown', 42, 20.00 )
```

If you make a mistake and forget to specify one of the columns, Adaptive Server Anywhere reports an error.

You can also add new rows to database tables from the result set in Interactive SQL.

☞ For more information, see [“Editing table values in Interactive SQL” on page 227](#).

**NULL values in columns** The NULL value is a special value used to indicate that something is either not known or not applicable. Some columns are allowed to contain the NULL value, and others are not.

---

## Modifying rows in a table

In most databases, you need to update records that are already stored in the database. For example, the manager ID should change when employees are transferred between departments, as well as the department ID.

### ❖ Transfer employee #195 to department 400 in Interactive SQL

1. In Interactive SQL, type the following in the SQL Statements pane and press F5 to execute the statement.

```
UPDATE employee
SET dept_id = 400, manager_id = 1576
WHERE emp_id = 195
```

The statement carries out both updates at the same time for employee Marc Dill (employee ID 195).

Since Adaptive Server Anywhere updates all rows that satisfy the conditions of the WHERE clause, sometimes more than one row is updated at one time. For example, if a group of sales employees are transferred into marketing and have their dept\_id column updated, the following statement sets the manager\_id for all employees in the marketing department to 1576.

```
UPDATE employee
SET manager_id = 1576
WHERE dept_id = 400
```

For employees already in the marketing department, no change is made.

You can also modify rows from the result set in Interactive SQL.

☞ For more information, see [“Editing table values in Interactive SQL” on page 227](#).

## Deleting rows

Sometimes you will want to remove rows from a table. Suppose Rodrigo Guevara (employee ID 249) leaves the company. The following statement deletes Rodrigo Guevara from the employee table.

```
DELETE
FROM employee
WHERE emp_id = 249
```

With UPDATE and DELETE, the search condition can be as complicated as you need. For example, if the employee table is being reorganized, the following statement removes from the employee table all male employees hired between March 3, 1988 and March 3, 1989.

```
DELETE
FROM employee
WHERE sex = 'm'
      AND start_date BETWEEN '1988-03-03'
      AND '1989-03-03'
```

You can also delete rows from database tables from the Interactive SQL result set.

☞ For more information, see [“Editing table values in Interactive SQL” on page 227](#).

Since you have made changes to the database that you do not want to keep, you should undo the changes as follows:

```
ROLLBACK
```

---

## Grouping changes into transactions

Adaptive Server Anywhere expects you to group your commands into transactions. You commit a transaction to make changes to your database permanent. When you alter your data, your alterations are not made permanent right away. Instead, they are stored in your **transaction log** and are made permanent when you enter the COMMIT command.

Knowing which commands or actions signify the start or end of a transaction lets you take full advantage of transactions.

### Starting transactions

Transactions start with one of the following events:

- ◆ The first statement following a connection to a database.
- ◆ The first statement following the end of a transaction.

### Completing transactions

Transactions complete with one of the following events:

- ◆ A COMMIT statement makes the changes to the database permanent.
- ◆ A ROLLBACK statement undoes all the changes made by the transaction.
- ◆ A statement with a side effect of an automatic commit is executed: Database definition commands, such as ALTER, CREATE, COMMENT, and DROP all have the side effect of an automatic commit.
- ◆ A disconnection from a database performs an implicit rollback.

### Options in Interactive SQL

Interactive SQL provides you with two options which let you control when and how transactions end:

- ◆ If you set the option AUTO\_COMMIT to ON, Interactive SQL automatically commits your results following every successful statement and automatically performs a ROLLBACK after each failed statement.
- ◆ The setting of the option COMMIT\_ON\_EXIT controls what happens to uncommitted changes when you exit Interactive SQL. If this option is set to ON (the default), Interactive SQL does a COMMIT; otherwise it undoes your uncommitted changes with a ROLLBACK statement.

#### **If you are using a data source**

By default, ODBC operates in autocommit mode. Even if you have set the AUTO\_COMMIT option to OFF in Interactive SQL, ODBC's setting will override Interactive SQL's. You can change ODBC's setting using the SQL\_ATTR\_AUTOCOMMIT connection attribute. ODBC autocommit is independent of the CHAINED option.



Adaptive Server Anywhere also supports Transact-SQL commands, such as `BEGIN TRANSACTION`, for compatibility with Sybase Adaptive Server Enterprise.

☞ For further information, see “An overview of Transact-SQL support” [*ASA SQL User’s Guide*, page 474].

## Making changes permanent

The `COMMIT` statement makes all changes permanent.

You should use the `COMMIT` statement after groups of statements that make sense together. For example, if you want to transfer money from one customer’s account to another, you should add money to one customer’s account, then delete it from the other’s, and then commit, since in this case it does not make sense to leave your database with less or more money than it started with.

You can instruct Adaptive Server Anywhere to commit your changes automatically by setting the `AUTO_COMMIT` option to `ON`. This is an Interactive SQL option. When `AUTO_COMMIT` is set to `ON`, Adaptive Server Anywhere must update the database after every insert, update, and delete statement you make to it. This can slow down performance considerably. Therefore, it is a good idea to leave the `AUTO_COMMIT` option set to `OFF`.

### Use COMMIT with care

When trying the examples in this tutorial, be careful not to `COMMIT` any changes until you are sure that you want to change the database permanently.

☞ For more information about Interactive SQL options, see “Interactive SQL options” [*ASA Database Administration Guide*, page 630].

## Canceling changes

Any uncommitted change you make can be cancelled. SQL allows you to undo all of the changes you made since your last commit with the `ROLLBACK` statement.

The `ROLLBACK` statement undoes all changes you have made to the database since the last time you made changes permanent.

☞ For more information on Interactive SQL options, see “Interactive SQL options” [*ASA Database Administration Guide*, page 630].

---

## Transactions and data recovery

Suppose that a system failure or power outage suddenly takes your database server down. Adaptive Server Anywhere is carefully designed to protect the integrity of your database in such circumstances. It provides you with a number of independent means of restoring your database. For example, it provides you with a **log file** which you may store on a separate drive so that should the system failure damage one drive, you still have a means of restoring your data. In addition, when you use a log file, Adaptive Server Anywhere does not have to update your database as frequently, which improves your database's performance.

Transaction processing allows the database server to identify states in which your data is in a consistent state. Transaction processing ensures that if, for any reason, a transaction is not successfully completed, then the entire transaction is undone, or rolled back. The database is left entirely unaffected by failed transactions.

Adaptive Server Anywhere's transaction processing ensures that the contents of a transaction are processed securely, even in the event of a system failure in the middle of a transaction.

☞ For a detailed description of data recovery mechanisms, see “[Backup and Data Recovery](#)” [*ASA Database Administration Guide*, page 373].

## Integrity checking

Adaptive Server Anywhere automatically checks for some common errors in your data.

### Inserting duplicate data

For example, suppose you attempt to create a department, but supply a dept\_id value that is already in use:

To do this, enter the command:

```
INSERT
INTO department ( dept_id, dept_name, dept_head_id )
VALUES ( 200, 'Eastern Sales', 902 )
```

The INSERT is rejected as it would make the primary key for the table not unique. Since dept\_id field is a primary key, duplicate values are not permitted.

### Inserting values that violate relationships

The following statement inserts a new row in the sales\_order table, but incorrectly supplies a sales\_rep ID that does not exist in the employee table.

```
INSERT
INTO sales_order ( id, cust_id, order_date,
sales_rep)
VALUES ( 2700, 186, '1995-10-19', 284 )
```

There is a one-to-many relationship between the employee table and the sales\_order table, with a join between the sales\_rep field of the sales\_order table and the emp\_id field of the employee table. Only after a record in the table containing the primary key for the join (the employee table) has been entered can a corresponding record on table containing the foreign key (the sales\_order table) be inserted.

#### Foreign keys

The primary key for the employee table is the employee ID number. The sales rep ID number in the sales\_rep table is a **foreign key** for the employee table, meaning that each sales rep number in the sales\_order table must match the employee ID number for some employee in the employee table.

When you try to add an order for sales rep 284 you get an error message:

```
No primary key value for foreign key 'ky_so_employee_id'
in table 'sales_order'
```

There isn't an employee in the employee table with that ID number. This prevents you from inserting orders without a valid sales rep ID. This kind of

---

validity checking is called **referential integrity** checking as it maintains the integrity of references among the tables in the database.

☞ For more information on primary and foreign keys, see [“Relations between tables” on page 119](#).

## Errors on DELETE or UPDATE

Foreign key errors can also arise when doing update or delete operations. For example, suppose you try to remove the R&D department from the department table. The dept\_id field, being the primary key of the department table, constitutes the ONE side of a one-to-many relationship (the dept\_id field of the employee table is the corresponding foreign key, and hence forms the MANY side). A record on the ONE side of a relationship may not be deleted until all corresponding records on the MANY side are deleted.

```
DELETE
FROM department
WHERE dept_id = 100
```

Example: DELETE errors

An error is reported indicating that there are other records in the database that reference the R&D department, and the delete operation is not carried out.

```
primary key for row in table 'department' is referenced
in another table
```

In order to remove the R&D department, you need to first get rid of all employees in that department:

```
DELETE
FROM employee
WHERE dept_id = 100
```

You can now perform the deletion of the R&D department.

You should cancel these changes to the database (for future use) by entering a ROLLBACK statement:

```
ROLLBACK WORK
```

All changes made since the last successful COMMIT WORK are undone. If you have not done a COMMIT, then all changes since you started Interactive SQL are undone.

Example: UPDATE errors

Now, suppose you try to change the dept\_id field from the employee table. The dept\_id field, being the foreign key of the employee table, constitutes the MANY side of a one-to-many relationship (the dept\_id field of the department table is the corresponding primary key, and hence forms the ONE side). A record on the MANY side of a relationship may not be

changed unless it corresponds to a record on the ONE side. That is, unless it has a primary key to reference.

For example, the following UPDATE statement causes an integrity error:

```
UPDATE employee
SET dept_id = 600
WHERE dept_id = 100
```

The error no primary key value for foreign key 'ky\_dept\_id' in table 'employee' is raised because there is no department with the a primary key of 600 in the department table.

In order to change the value of the dept\_id field in the employee table, it must correspond to an existing value in the department table. For example:

```
UPDATE employee
SET dept_id = 300
WHERE dept_id = 100
```

This statement can be executed because the dept\_id of 300 corresponds to the existing Finance department.

Cancel these changes to the database by entering a ROLLBACK statement:

```
ROLLBACK WORK
```

Checking the integrity after the COMMIT WORK is complete

In all of the above examples, the integrity of the database was checked as each command was executed. Any operation that would result in an inconsistent database is not performed.

It is possible to configure the database so that the integrity is not checked until the COMMIT WORK is done. This is important if you want to change the value of a referenced primary key; for example, deleting the R&D department in the employee and department tables. In order to make these deletions, the database has to be inconsistent in between the changes. In this case, you must configure the database to check only on commits.

☞ For more information, see “[WAIT\\_FOR\\_COMMIT option \[database\]](#)” [*ASA Database Administration Guide*, page 702].

You can also define foreign keys in such a way that they are automatically modified to be consistent with changes made to the primary key. In the above example, if the foreign key from employee to department were defined with ON DELETE CASCADE, then deleting the department ID would automatically delete the corresponding entries in the employee table.

In the above cases, there is no way to have an inconsistent database committed as permanent. Adaptive Server Anywhere also supports alternative actions if changes would render the database inconsistent.

---

☞ For more information, see the chapter [“Ensuring Data Integrity”](#) [*ASA SQL User’s Guide*, page 79].

## PART IV

# SQL ANYWHERE STUDIO TUTORIALS

This part contains tutorials to help you get started using all the components of SQL Anywhere Studio. Each chapter is a self contained unit on a specific topic.





---

## CHAPTER 12

# The Sample Database

### About this chapter

This chapter provides basic information about the database servers and the sample database. It shows you how to make a copy of the sample database, how to start the database server running the sample database, how to view the database server window, and how to shut down the server.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About the sample database</a>	198
<a href="#">Lesson 1: Make a copy of the sample database</a>	200
<a href="#">Lesson 2: Start the Adaptive Server Anywhere database server</a>	201
<a href="#">Lesson 3: Displaying the database server window</a>	203
<a href="#">Lesson 4: Shut down the database server</a>	205
<a href="#">Summary</a>	206

---

## About the sample database

The Adaptive Server Anywhere database server is the piece of software that manages the database. The database server allows access to databases from client applications, and processes commands in a secure and efficient manner. Only one database server at a time can manage any one database.

All access to your database must be made through a database server.

Two kinds of database server

There are two versions of the Adaptive Server Anywhere database server. The **personal server** can only accept connections from applications or users running on the same machine. It is intended for single-user, same-machine use.

By contrast, the **network server** supports client/server communication over a network and is intended for multi-user operation.

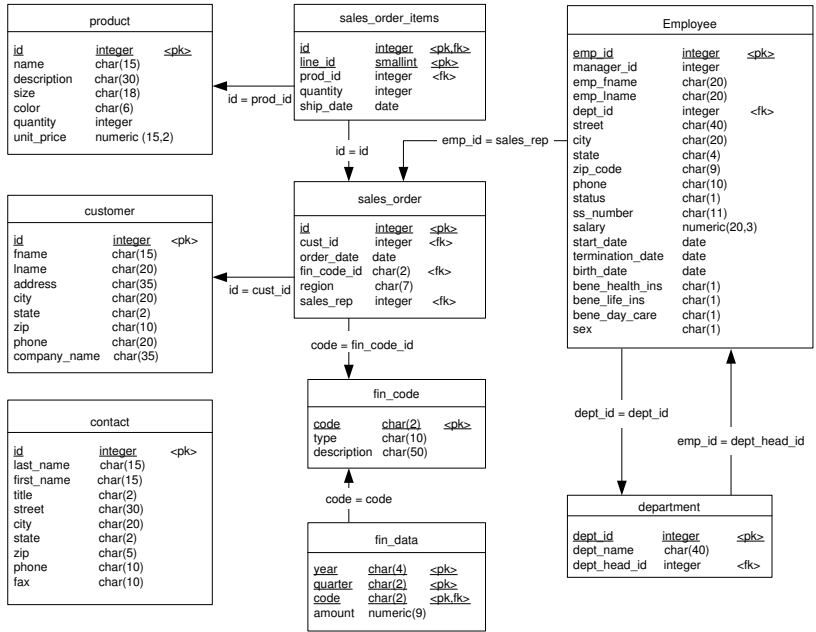
The two database servers are exactly the same in their query processing and other internal operations: both the personal database server and the network database server are equivalent.

The sample database

This tutorial focuses on the sample database. The sample database represents a small company that makes a limited range of sports clothing. It contains internal information about the company (employees, departments, and financial data), as well as product information (products) and sales information (sales orders, customers, and contacts). All information in the sample database is fictional.

The following figure displays the tables in the sample database and how they are related to each other. The boxes represent tables, and the arrows represent foreign key relationships. Primary key columns are underlined.

asademo.db



---

## Lesson 1: Make a copy of the sample database

Before you begin, make a copy of the sample database so that you can restore it after you have made changes.

### ❖ To copy the sample database

1. Navigate to the directory that contains the sample database file, *asademo.db*.  
The default installation directory for it is *Program Files\Sybase\Adaptive Server Anywhere 9*.
2. Create a sub-directory to hold a copy of *asademo.db*.  
For example, call it *demoback*.
3. Create a copy of *asademo.db* in the backup directory.  
Work with the original version. Do not change the backup copy.

## Lesson 2: Start the Adaptive Server Anywhere database server

In this section you start the Adaptive Server Anywhere database server running the sample database. The sample database is held in a file named *asademo.db* in your SQL Anywhere installation directory.

Depending on the operating system you are using, you may have a choice of how to start the database server running the sample database.

### ❖ To start the personal database server running the sample database (Windows)

1. From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Personal Server Sample.

This starts a personal server running the sample database. The database server window appears briefly, then quickly disappears. The server then appears as an icon in the system tray, at the opposite end of the Taskbar from the Start button.

### ❖ To start the database server running the sample database (Command prompt)

1. Open a command prompt.
2. Change to the SQL Anywhere installation directory.

On Windows operating systems, the default installation directory is *C:\Program Files\Sybase\SQL Anywhere 9*.

3. Start the database server running the sample database.

The way you start the database server depends on your operating system, and on whether you wish to connect to the database from other machines on the network.

- ◆ If you wish to connect only from the same machine on Windows or UNIX operating systems, enter the following command to start the personal database server:

```
dbeng9 -n asademo9 asademo.db
```

- ◆ If you wish to connect to the database server from other machines on the network on Windows or UNIX operating systems, enter the following command to start the network database server:

```
dbsrv9 -n asademo9 asademo.db
```

- 
- ◆ On NetWare, only the network database server is provided. Enter the following command:

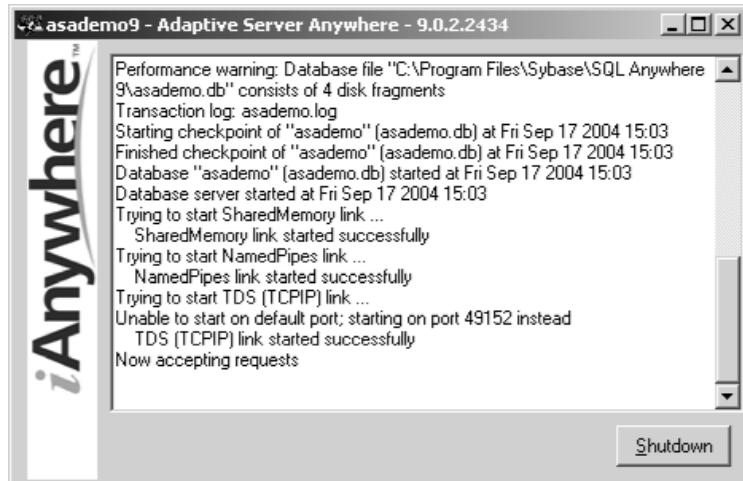
```
load dbsrv9.nlm -n asademo9 asademo.db
```

The server appears as an icon in the system tray, at the opposite end of the Taskbar from the Start button.

## Lesson 3: Displaying the database server window

At this point, you have successfully started the database server running the sample database. However, you can't see or manipulate the data in the database yet.

In fact, the Adaptive Server Anywhere icon is the only visible indication that anything has happened. You can display the database server window now (or at any time for that matter) in Windows by double-clicking the Adaptive Server Anywhere icon in the system tray.



The server window displays useful information, including the following:

- ◆ **The server name** The name in the title bar (in this case **asademo9**) is the **server name**. A server name is assigned each time a database server is started. This name can be used by applications when they connect to a database.
- ◆ **The version and build numbers** The numbers following the server name (in this case **9.0.2.2434**) are the **version and build numbers**. The version number represents the specific release of the SQL Anywhere Studio, and the build number relates to the specific instance that the software was compiled.
- ◆ **Startup information** When a database server starts, it sets aside some memory that it uses when processing database requests. This is called the **cache**. The amount of cache memory appears in the window. The cache is organized in fixed-size **pages**, and the page size also appears in the window.

- 
- ◆ **Database information** The names of the database file and its transaction log file appear in the window.

In this case, the startup cache size and page size are the default values. For many purposes, including those of this tutorial, the default startup options are fine.



## Lesson 4: Shut down the database server

You can now shut down the database server you just started.

❖ **Shut down the database server running the sample database (Windows)**

1. Double-click the Adaptive Server Anywhere icon in the Windows task bar.

The database server window appears.

2. Click Shutdown.

---

## Summary

In this tutorial, you learned how to make a copy of the sample database, how to start the database server running the sample database, and how to view the contents of the database server window. You also learned how to shut the database server down.

---

## CHAPTER 13

# Making the Connection

About this chapter

This chapter shows you how to create an ODBC data source, and how to use it to establish a connection between a database and an application. In this case, the application is Sybase Central.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About connections</a>	208
<a href="#">About ODBC data sources</a>	209
<a href="#">Lesson 1: Create an ODBC data source</a>	210
<a href="#">Lesson 2: Connect using the Sample ODBC data source</a>	213
<a href="#">Lesson 3: Other means of connecting</a>	214
<a href="#">Lesson 4: Disconnecting from the sample database</a>	215
<a href="#">Summary</a>	216

---

## About connections

Any client application that uses a database must establish a connection to that database before any work can be done. The **connection** forms a channel through which all activity from the client application takes place. For example, your user ID determines permissions to carry out actions on the database—and the database server has your user ID because it is part of the request to establish a connection.

## About ODBC data sources

The **Open Database Connectivity (ODBC)** interface is defined by Microsoft Corporation, and is a standard interface for connecting client applications to database-management systems in Windows environments. Connections are made by specifying connection parameters. It is often convenient to collect a set of connection parameters together and store them in an **ODBC data source**. ODBC data sources are a convenient way of saving connection parameters for repeated use.

Once you have a data source, your connection string can simply name the data source to use:

```
DSN=my data source
```

You can use ODBC data sources to connect to Adaptive Server Anywhere databases from any of the following applications:

- ◆ Sybase Central and Interactive SQL.
- ◆ All the Adaptive Server Anywhere utilities.
- ◆ PowerDesigner and InfoMaker.
- ◆ Any application development environment that supports ODBC, such as Microsoft Visual Basic, Sybase PowerBuilder, and Borland Delphi.
- ◆ Adaptive Server Anywhere client applications on UNIX. On UNIX, the data source is stored as a file.

# Lesson 1: Create an ODBC data source

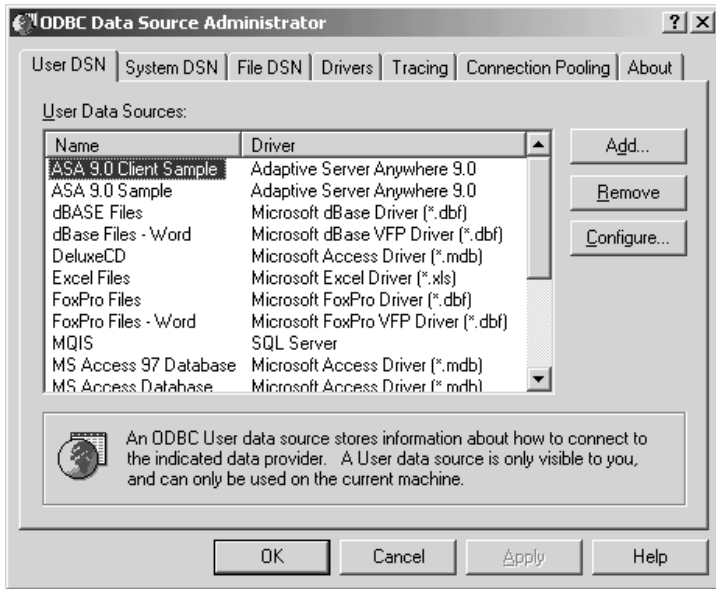
This section describes how to create a simple ODBC data source in Windows.

## ❖ To create a simple ODBC data source

1. Start the ODBC Administrator:

- ◆ From the Start menu, choose Programs > SQL Anywhere 9 > Adaptive Server Anywhere > ODBC Administrator.

The ODBC Data Source Administrator appears, displaying a list of the data sources you currently have installed on your computer.

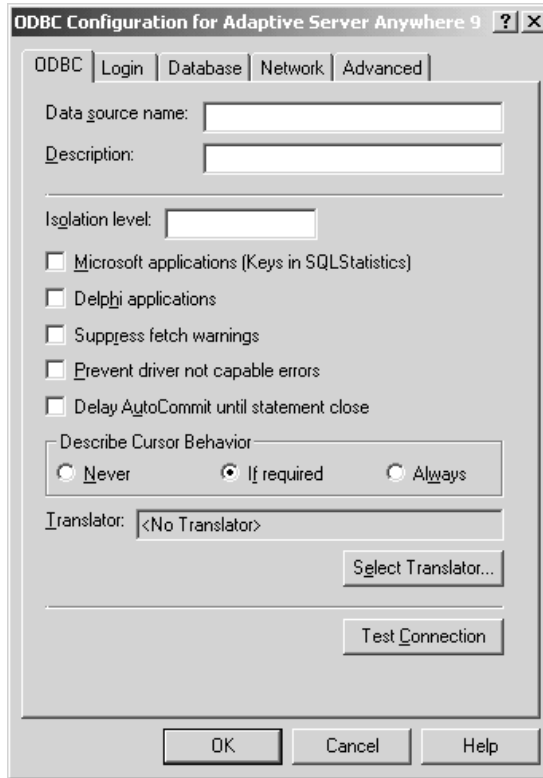


2. On the User DSN tab, click Add.

The Create New Data Source wizard appears.

3. Select **Adaptive Server Anywhere 9.0** from the list of drivers, and click Finish.

The Configuration dialog appears.



Many of the fields in this dialog are optional. Click the Help button at the bottom of each tab for a description of all the fields on that tab. For many purposes you only need to use a few parameters. The following data source parameters are sufficient for the Adaptive Server Anywhere sample database:

- ◆ **Data Source Name (ODBC tab)** Enter a data source name of **Sample**. This is the name that appears in the Connect dialog. It can contain spaces, but should be short.
- ◆ **User ID (Login tab)** Enter the default user ID **DBA**. This is the database user ID you will use to connect.
- ◆ **Password (Login tab)** Enter **SQL**. This is the default password for the DBA user ID.
- ◆ **Database File (Database tab)** Browse to your SQL Anywhere installation directory, and select the *asademo.db* file. This is the Adaptive Server Anywhere sample database.

4. Test the data source.

On the ODBC tab, click Test Connection.

---

If the datasource was created properly, the message Connection Successful appears.

If the message Connection Failed appears, review the data source settings and make corrections where necessary.

5. Click OK to create the data source.



## Lesson 2: Connect using the Sample ODBC data source

For this example, you will connect to the sample database from Sybase Central, Adaptive Server Anywhere's graphical tool for managing databases.

### ❖ To connect to the sample database from Sybase Central

1. From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central to start Sybase Central.
2. From the Tools menu, choose Connect to open the Connect dialog.  
If a dialog prompts you to choose a plug-in, choose Adaptive Server Anywhere 9 and click OK.  
The Connect dialog opens.
3. Select the ODBC Data Source Name option.
4. In the ODBC Data Source Name box, type **Sample**, and click OK.  
Alternatively, you could also click Browse and choose the ASA 9.0 Sample from the Data Source Names dialog.
5. Click OK to connect to the sample database.  
The dialogs disappear, and the database opens in the Sybase Central window.

---

## Lesson 3: Other means of connecting

You can use ODBC data sources to connect to Adaptive Server Anywhere—not only from ODBC applications, but also from embedded SQL applications, such as the administration utilities, and from Interactive SQL and Sybase Central—even though these applications do not use ODBC. The functionality to process ODBC data sources is built into the embedded SQL client library, and into Sybase Central and Interactive SQL.

☞ Adaptive Server Anywhere supports several programming interfaces in addition to ODBC. For more information, see “[Introduction to connections](#)” [*ASA Database Administration Guide*, page 38].

The Connect dialog in Sybase Central and Interactive SQL has fields for entering an ODBC Data Source Name or ODBC Data Source File.

## Lesson 4: Disconnecting from the sample database

To complete this tutorial, shut down the database server and close Sybase Central.

### ❖ To shut down the database server

1. From the Tools menu, choose Disconnect.

The database server disconnects from the sample database and shuts down.

### ❖ To close Sybase Central

1. From the File menu, choose Exit.

Sybase Central shuts down.

---

## Summary

In this tutorial, you learned how to create an ODBC data source, and how to use it to establish a connection between a database and Sybase Central.

---

## CHAPTER 14

# Using Interactive SQL

About this chapter

This chapter discusses how to start and use Interactive SQL. Interactive SQL is a utility shipped with Adaptive Server Anywhere that lets you execute SQL statements, build scripts, and display database data.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About Interactive SQL</a>	218
<a href="#">Lesson 1: Start Interactive SQL</a>	219
<a href="#">Lesson 2: The Interactive SQL interface</a>	220
<a href="#">Lesson 3: Displaying data using Interactive SQL</a>	226
<a href="#">Lesson 4: Working with SQL statements</a>	232
<a href="#">Summary</a>	239

---

## About Interactive SQL

You can use Interactive SQL for the following purposes:

- ◆ sending SQL statements to the database server
- ◆ browsing the information in a database
- ◆ trying out SQL statements that you plan to include in an application
- ◆ loading data into a database
- ◆ carrying out administrative tasks

In addition, Interactive SQL can run **command files** or **script files**. For example, you can build repeatable scripts to run against a database and then use Interactive SQL to execute those scripts in a batch fashion.

## Lesson 1: Start Interactive SQL

In this section you start the Interactive SQL utility, connect to the sample database, and enter a command.

### ❖ To start Interactive SQL and connect to the sample database (Windows)

1. From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL.

The Connect dialog appears.

2. Select the ODBC Data Source Name option.
3. Click Browse, and select **ASA 9.0 Sample** from the list.
4. Click OK.
5. Click OK to connect to the sample database.

### ❖ To start Interactive SQL and connect to the sample database (Command prompt)

1. If you are using a Windows or UNIX operating system, enter the following command to start the personal database server:

```
dbeng9 -n asademo9
```

2. Type the following command at a command prompt to start Interactive SQL and connect to the sample database:

```
dbisql -c "DSN=ASA 9.0 Sample"
```

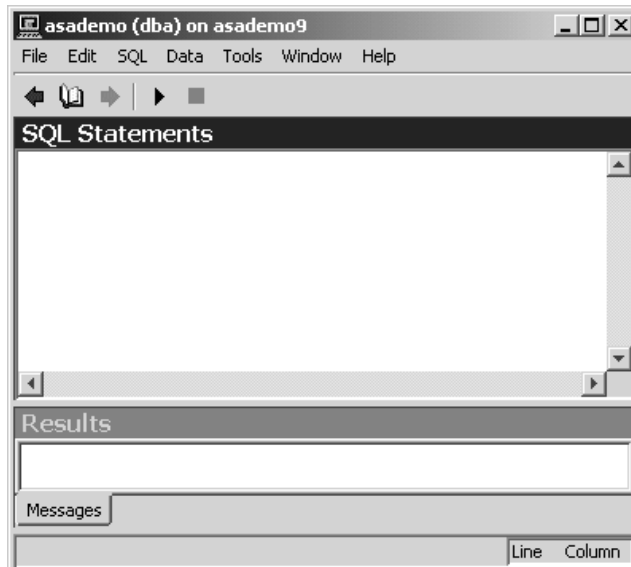
---

## Lesson 2: The Interactive SQL interface

In this section, you learn about the components of the Interactive SQL main window, Interactive SQL keyboard shortcuts, and tools that are available in Interactive SQL.

### Interactive SQL main window description

After connecting to the database, the Interactive SQL window appears.



The title bar at the top of the window displays connection information, as follows:

```
database-name ( userid ) on server-name
```

For example, if you connect to the sample database using the ASA 9.0 Sample ODBC data source, the title bar is as follows:

```
asademo ( dba ) on asademo9
```

The Interactive SQL panes

Interactive SQL has the following panes:

- ◆ **SQL Statements** This pane provides a place for you to type SQL statements to access and modify your data.
- ◆ **Results** This tab displays the results of commands that you execute. For example, if you use SQL statements to search for specific data in the database, the Results tab displays the columns and rows that match the search criteria in the pane above. If the information exceeds the size of



the pane, scroll bars automatically appear. You can edit the result set on the Results tab.

☞ For more information about editing the result set, see [“Editing table values in Interactive SQL” on page 227](#).

- ◆ **Messages** This tab displays messages from the database server about the SQL statements that you execute in Interactive SQL.
- ◆ **Plan** This tab, as well as the UltraLite Plan tab, displays the query optimizer’s execution plan for a SQL statement.

You can set options for the tabs and panes in the main Interactive SQL window from the Options dialog found in the Tools menu.

## Using the Interactive SQL toolbar

The Interactive SQL toolbar appears at the top of the Interactive SQL window. It provides you with buttons for executing common commands. With the buttons on this toolbar, you can:

- ◆ Recall the executed SQL statement immediately before your current position in the history list.
- ◆ View a list of up to 50 previously executed SQL statements.
- ◆ Recall the executed SQL statement immediately after your current position in the history list.
- ◆ Execute the SQL statement currently appearing in the SQL Statements pane.
- ◆ Interrupt the execution of the current SQL statement.

As an easy reminder of what these buttons do, you can hold your cursor over each button to see a popup description.

## Opening multiple windows

You can open multiple Interactive SQL windows. Each window corresponds to a separate database connection. You can connect simultaneously to two (or more) different databases on different servers, or you can open concurrent connections to a single database.

---

## ❖ To open a new Interactive SQL window

1. From the Window menu, choose New Window.

The Connect dialog appears.

### Tip

If the SQLCONNECT environment variable is set, or if you are already connected to a database, the server attempts to use this information to connect to a database before it prompts you for information in. If these attempts fail, or if you are not already connected to a database, the Connect dialog appears.

2. In the Connect dialog, enter connection options, and click OK to connect.


The connection information (including the database name, your user ID, and the database server name) appears on the title bar above the SQL Statements pane.

You can also connect to or disconnect from a database with the Connect and Disconnect commands in the SQL menu, or by executing a CONNECT or DISCONNECT statement.

## Interactive SQL keyboard shortcuts

Interactive SQL provides the following keyboard shortcuts:

Function key	Description
ALT+F4	Exits Interactive SQL.
ALT+LEFT CURSOR	Displays the previous SQL statement in the history list.
ALT+RIGHT CURSOR	Displays the next SQL statement in the history list.
CTRL+BREAK	Interrupts the SQL statement that is being executed.
CTRL+C	Copies the selected row(s) and column headings to the clipboard in the Results pane. In the SQL Statements pane, copies the selected text to the clipboard.
CTRL+END	Moves to the bottom of the current pane.
CTRL+H	Displays the history of your executed SQL.
CTRL+HOME	Moves to the top of the current pane.
CTRL+N	Clears the contents of the Interactive SQL window.

Function key	Description
CTRL+P	<p>Prints the contents of the SQL Statements pane. You can configure the appearance of the printed text in the Interactive SQL Options dialog.</p> <p> For information about setting print options, see “Print tab” [<a href="#">SQL Anywhere Studio Help, page 166</a>].</p>
CTRL+Q	<p>Displays the Query Editor.</p> <p>The Query Editor helps you build SQL queries. When you have finished building your query, click OK to export it back into the SQL Statements pane.</p>
CTRL+S	<p>Saves the contents of the SQL Statements pane.</p>
ESC	<p>Clears the SQL Statements pane.</p>
F1	<p>Opens Help.</p>
F2	<p>Edits the selected value in the result set. You can tab from column to column within the row.</p>
F5	<p>Executes all text in the SQL Statements pane.</p> <p>You can also perform this operation by clicking the Execute SQL Statement button on the toolbar.</p>
F7	<p>Displays the Lookup Table Name dialog.</p> <p>In this dialog, you can find and select a table and then press ENTER to insert the table name into the SQL Statements pane at the cursor position. Or, with a table selected in the list, press F7 again to display the columns in that table. You can then select a column and press ENTER to insert the column name into the SQL Statements pane at the cursor position.</p>
F8	<p>Displays the Lookup Procedure Name dialog.</p> <p>In this dialog, you can find and select a procedure and then press ENTER to insert the procedure name into the SQL Statements pane at the cursor position.</p>
F9	<p>Executes the text that is selected in the SQL Statements pane.</p> <p>If no text is selected, all of the statements are executed.</p>
PGDN	<p>Moves a page down in the current pane.</p>
PGUP	<p>Moves a page up in the current pane.</p>

---

<b>Function key</b>	<b>Description</b>
SHIFT+F5	Displays the plan for the statement in the SQL Statements pane without executing the statement.
SHIFT+F10	Displays the context menu for the area that has focus. This keyboard shortcut is an alternative to right-clicking on an area.

The following keyboard shortcuts are available when the SQL Statements pane has the focus:

<b>Function key</b>	<b>Description</b>
CTRL+]	Moves the cursor to the matching brace. Brace matching matches parentheses, braces, brackets, and angle brackets.
CTRL+BACKSPACE	Deletes the word to the left of the cursor.
CTRL+DEL	Deletes the word to the right of the cursor.
CTRL+G	Opens the Go To dialog where you can specify the line you want to go to.
CTRL+L	Deletes the current line from the SQL Statements pane and puts the line onto the clipboard.
CTRL+SHIFT+]	Extends the selection to the matching brace. Brace matching matches parentheses, braces, brackets, and angle brackets.
CTRL+SHIFT+L	Deletes the current line.
CTRL+SHIFT+U	Changes the selection to upper case characters.
CTRL+U	Changes the selection to lower case characters.
F3	Finds the next occurrence of the selected text.
HOME	Moves the cursor to the start of the current line or to the first word on the current line.
SHIFT+F3	Finds the previous occurrence of the selected text.
SHIFT+HOME	Extends the selection to the start of the text on the current line.

## Learning more about Interactive SQL

Interactive SQL provides a set of windows that let you configure its appearance and behavior.

☞ For more information, see [“Interactive SQL Help”](#) [*SQL Anywhere Studio Help*, page 153].

From Interactive SQL you can gain access to an Index Consultant, also available from Sybase Central, that helps you improve query performance.

☞ For more information, see [“Index Consultant overview”](#) [*ASA SQL User’s Guide*, page 67].

From Interactive SQL you can also gain access to a Query Editor, which helps you to design, analyze, and test all kinds of queries.

☞ For more information, see [“Query Editor Help”](#) [*SQL Anywhere Studio Help*, page 225].

---

## Lesson 3: Displaying data using Interactive SQL

One of the principal uses of Interactive SQL is to browse table data. This section shows how to query the information in the sample database. Queries take the form of SQL statements.

You can display database information in Interactive SQL using the `SELECT` statement. The following example shows the command to type in the SQL Statements pane. Once you have typed the command, you must click the Execute SQL Statement button on the toolbar to carry out the command.

After you execute the statement, the data (called a result set) appears on the Results tab in the Results pane. You can use the scroll bars to see areas of the table that are outside your current view of the pane. By default, row numbers appear to the left of the result set.

### ❖ To list all the columns and rows of the employee table

1. Connect to the sample database if you are not already connected.

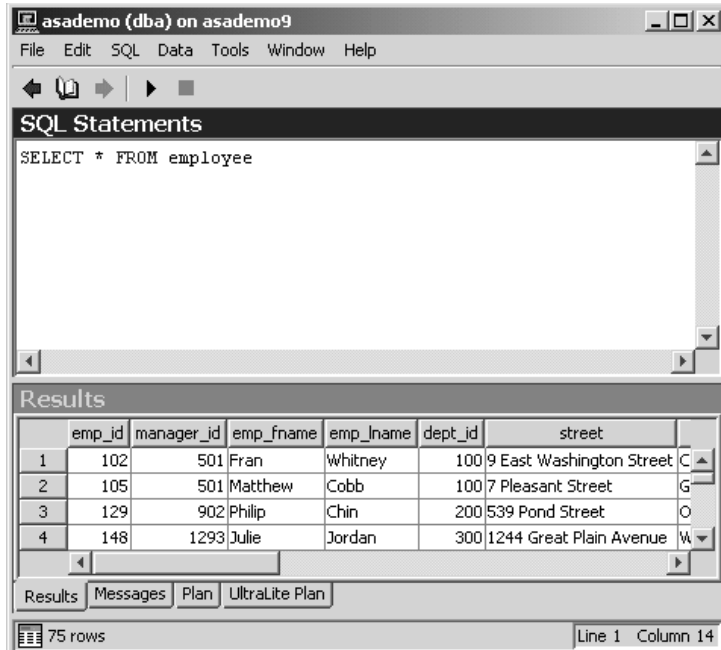
☞ For instructions on connecting to the sample database, see [“Lesson 1: Start Interactive SQL” on page 219](#).

2. In the SQL Statements pane, type the following SQL query.

```
SELECT * FROM employee
```

3. Press F5 to execute the statement.

The query displays all the columns of the table named `employee`. This table contains information about the employees of a fictitious merchandising company. The following appears:



Interactive SQL retrieves the information by sending a request to your database server. The database server, in turn, looks up the information in the employee table and returns it to Interactive SQL.

At this point, you may want to look at the data in some of the other tables in the sample database, such as product, department, or sales\_order.

You can add, delete, and update rows within the result set.

➔ For more information about editing the result set, see [“Editing table values in Interactive SQL” on page 227](#).

## Editing table values in Interactive SQL

Once you execute a query in Interactive SQL, you can edit the result set to modify the database. You can also select rows from the result set and copy them for use in other applications. Interactive SQL supports editing, inserting, and deleting rows. These actions have the same effect as executing UPDATE, INSERT, and DELETE statements.

To edit a row or value in the result set, you must have the proper permissions on the table or column you want to modify values from. For example, if you want to delete a row, then you must have DELETE permission for the table the row belongs to.

---

You cannot edit a result set if:

- ◆ you select columns from a table with a primary key, but do not select all of the primary key columns.
- ◆ you attempt to edit the result set of a JOIN (for example, if there is data from more than one table in the result set).

Editing the result set may fail if:

- ◆ you attempt to edit a row or column you do not have permission on.
- ◆ you enter an invalid value (for example, a string in a numeric column or a NULL in a column that does not allow NULLs).

When editing fails, an Interactive SQL error message appears explaining the error, and the database table values remain unchanged.

Once you make changes to table values, you must execute a COMMIT statement to make the changes permanent. If you want to undo your changes, you must execute a ROLLBACK statement.

☞ For more information, see “COMMIT statement” [ASA SQL Reference, page 329] and “ROLLBACK statement” [ASA SQL Reference, page 591].

## Editing table values from the Interactive SQL result set

From Interactive SQL you can change any or all of the values within existing rows in database tables. You must have UPDATE permission on the columns being modified. When you edit the result set, you can only make changes to the values in one row at a time.

### ❖ To edit a row in the result set

1. Click the value you want to change.
2. Right-click the value and choose Edit from the popup menu.  
Alternatively, press F2 to edit the result set.  
A blinking cursor appears in the table cell containing the value.
3. Enter the new value.

You cannot enter invalid data types into a column. For example, you cannot enter a string in a column that accepts the INT data type.

When you are done editing values in the row, press Enter to update the database. If you want to change other values in the row, press TAB or SHIFT+TAB to move to the other values.

You can press the ESC key to cancel the change that was made to the selected value.



4. Execute a COMMIT statement to make your changes to the table permanent.

Editing computed columns

Once you edit values in the result set, the database is updated with the modified values. Computed columns are recalculated based on the modified values whether or not they are part of the result set. If there are computed columns in your result set, and you modify a value in the computed column, the database is updated with the modified value.

## Inserting rows into the database from the Interactive SQL result set

Interactive SQL allows you to add new rows to result sets. You tab between columns in the result set to add values to the row. When you add values to the table, characters are stored in the same case as they are entered. You must have INSERT permission on the table to add new rows.

### ❖ To insert a new row into the result set

1. Right-click the result set and choose Add from the popup menu.

Alternatively, press CTRL+F10 to display the context menu and choose Add.

A new blank row appears with a blinking cursor in the first value in the row.

Press TAB to move the cursor from column to column across the row. You can also insert a value by clicking on the value within the selected row.

2. Enter the new value.

You cannot enter invalid data types into a column. For example, you cannot enter a string into a column that accepts the INT data type.

3. Press TAB to move to the next column.

4. Repeat steps 2 and 3 until all the column values are added.

5. Press ENTER to update the database.

6. Execute a COMMIT statement to make your changes to the table permanent.

Inserting values into computed columns

If the result set contains a computed column and you do not specify a value for the computed column, the value is calculated when the database is updated. However, if you specify a value for the computed column, the database is updated with the specified value, and a value is not calculated for the computed column.

Inserting new rows using the INPUT statement

An alternative to inserting new rows from the result set in Interactive SQL is to add rows using the INPUT statement with the PROMPT clause. When the

---

PROMPT clause is specified, Interactive SQL prompts you for the value for each column in the table. For example, to add a new row to the product table and be prompted for the values for each column, you would execute the following statement in Interactive SQL:

```
INPUT INTO product
PROMPT
```

## Deleting rows from the database using Interactive SQL

You can also delete rows from a database table in Interactive SQL. You must have DELETE permission on the table to delete rows. You can select only consecutive rows in the result set.

### ❖ To delete a row from the result set

1. Select the row(s) you want to delete. To select a row(s):
  - ◆ Press and hold the SHIFT key while clicking the row(s)
  - ◆ Press and hold the SHIFT key while using the Up or Down arrowIf you want to delete non-consecutive rows, you must delete each row individually.
2. Right-click the result set and choose Delete from the popup menu. You can also delete the selected row(s) by pressing the DELETE key.  
The selected row(s) are removed from the database table.
3. Execute a COMMIT statement to make your changes to the table permanent.

## Copying rows from the Interactive SQL result set

You can copy rows directly from the result set in Interactive SQL and then paste them into other applications. Copying rows also copies the column headings. Copied data is comma-delimited, which allows other applications, such as Microsoft Excel, to format the copied data correctly. Copied data is in ASCII format, and all of the strings are enclosed in single quotes. You can select only consecutive rows in the result set.

### ❖ To copy rows from the Interactive SQL result set

1. Select the row(s) you want to copy. To select a row(s):
  - ◆ Press and hold the SHIFT key while clicking the row(s)
  - ◆ Press and hold the SHIFT key while using the Up or Down arrow

2. Right-click the result set and choose Copy from the popup menu. You can also copy the selected row(s) by pressing CTRL+C.

The selected row(s), including their column headings, are copied to the clipboard. You can paste them into other applications.

**Copying individual values from the result set** You can copy a single value from the result set by selecting a value, right-clicking the result set and choosing Copy Cell from the popup menu. When you do this, no column headings are copied—only the data is copied.

---

## Lesson 4: Working with SQL statements

The following sections describe some of the commands you can use in Interactive SQL. This section describes general tasks for working with commands in Interactive SQL.

All SQL statements can be entered as commands in the top pane of the Interactive SQL window. When you are finished typing, you need to execute the statement to view its results.

☞ You can enter any SQL statement against the database from within Interactive SQL. For a complete list of SQL statements, see “SQL Statements” [*ASA SQL Reference*, page 253].

### ❖ To execute a SQL statement

1. Press the Execute SQL Statement button, or choose SQL ► Execute, or press F5.

### ❖ To clear the SQL Statements pane

1. Choose Edit ► Clear SQL or press ESCAPE.

## Canceling an Interactive SQL command

The Interrupt button on the Interactive SQL toolbar cancels a command.

A Cancel operation stops the current processing and prompts for the next command. If a command file was being processed, or if there is more than one statement in the SQL Statements pane, you are prompted for an action to take (Stop Command File, Continue, or Exit Interactive SQL). These actions can be controlled with the Interactive SQL ON\_ERROR option.

☞ For information about the ON\_ERROR option, see “ON\_ERROR option [Interactive SQL]” [*ASA Database Administration Guide*, page 674].

### Reported messages

When an interruption is detected, one of three different messages is reported depending upon when the interruption is detected.

1. If the interruption is detected when Interactive SQL is processing the request (as opposed to the database server), then the following message appears:

```
ISQL command terminated by user
```

Interactive SQL stops processing immediately and the current database transaction is left alone.

2. If the interruption is detected by the database server while processing a data manipulation command (SELECT, INSERT, DELETE, or UPDATE), then the following message appears:

```
Statement interrupted by user.
```

The effects of the current command are undone, but the rest of the transaction is left intact.

3. If the interruption is detected while the database server is processing a data definition command (CREATE, DROP, ALTER, and so on), the following message appears:

```
Terminated by user - transaction rolled back
```

Since data definition commands all perform a COMMIT automatically before the command starts, the effect of the ROLLBACK is to just cancel the current command.

This message also occurs when the database server is running in bulk operations mode executing a command that modifies the database (INSERT, UPDATE, or DELETE). In this case, ROLLBACK cancels not only the current command, but everything that has been done since the last COMMIT. In some cases, it may take a considerable amount of time for the database server to perform the automatic ROLLBACK.

## Executing multiple statements

The Interactive SQL environment allows multiple statements to be entered at the same time. This can be done by ending each statement with a command delimiter. The command delimiter is a configurable option in Interactive SQL that you can change using the `COMMAND_DELIMITER` option. By default, it is a semicolon (;).

☞ For more information, see “[COMMAND\\_DELIMITER option \[Interactive SQL\]](#)” [*ASA Database Administration Guide*, page 642].

---

❖ **To enter multiple statements in SQL Statements pane**

1. Enter the following three commands into the SQL Statements pane.

```
UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 467;

UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 195;

SELECT *
FROM employee
WHERE emp_id IN ( 195, 467 );
```

2. On the toolbar, click the Execute SQL Statement button. All three statements are executed. After execution, the commands remain in the SQL Statements pane.
3. Roll back your changes by typing `ROLLBACK` in the SQL Statements pane and executing the statement.

Using `go` as an alternative

An alternative to using the semicolon is to enter `go` on a line by itself, at the beginning of the line.

```
UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 467
go

UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 195
go

SELECT *
FROM employee
WHERE emp_id IN ( 195, 467 )
go
```

**Tip**

You can press F9 to execute only the selected text in the SQL Statements pane.

## Looking up tables, columns, and procedures

While you are entering commands in Interactive SQL, you can look up the names of tables, columns, or procedures stored in the current database and insert them at your cursor position.

### ❖ To look up the names of tables in the database

1. Choose Tools ► Lookup Table Name or press F7.
2. Find and select the table.
3. Click OK to insert the table name into the SQL Statements pane at the current cursor position.

### ❖ To look up column names in the database

1. Choose Tools ► Lookup Table Name or press F7.
2. Find and select the table containing the column.
3. Click Show Columns.
4. Select the column and click OK to insert the column name into the SQL Statements pane at the current cursor position.

### ❖ To look up the names of procedures in the database

1. Choose Tools ► Lookup Procedure Name or press F8.
2. Find and select the procedure.
3. Click OK to insert the procedure name into the SQL Statements pane at the current cursor position.

In the tables and procedures lookup dialogs, you can enter the first few characters of the table or procedure you are looking for. After you type something in the field, the dialog narrows the list to include only those items that start with the text you entered.

You can use the SQL wildcard characters ‘%’ (percent sign) and ‘\_’ (underscore) to help narrow your search. ‘%’ matches any string of zero or more characters, while ‘\_’ matches any one character.

For example, to list all the tables that contain the word profile, type **%profile%**.

If you want to search for a percent sign or underscore within a table name, you must prefix the percent sign or underscore with an escape character. The

---

escape character depends on the JDBC driver that you are using. If you are connected via jConnect, the escape character is ‘\’ (backslash) while the escape character for the iAnywhere JDBC driver is ‘~’ (tilde).

## Printing SQL statements

You can print the contents of the SQL Statements pane by pressing CTRL+P or by choosing Print from the File menu. You can add a header or footer and configure other formatting options in the Interactive SQL Options dialog.

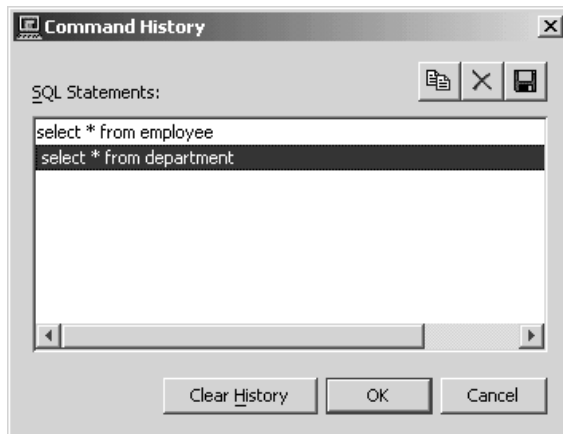
☞ For information about configuring print options, see “Print tab” [*SQL Anywhere Studio Help*, page 166].

☞ For information about print graphical plans in Interactive SQL, see “Graphical plans” [*ASA SQL User’s Guide*, page 461].

## Recalling commands

When you execute a command, Interactive SQL automatically saves it in a history list that persists between Interactive SQL sessions. Interactive SQL maintains a record of up to 50 of the most recent commands.

You can view the entire list of commands in the Command History dialog. To access the Command History dialog, press CTRL+H, or click the book icon on the toolbar.



The most recent commands appear at the bottom of the list. To recall a command, select it and then click OK. It appears in the SQL Statements pane of Interactive SQL.

You can also recall commands without the Command History dialog. Use the arrows in the toolbar to scroll back and forward through your commands,



or press ALT+RIGHT ARROW and ALT+LEFT ARROW.

**Note**

If you execute a SQL statement that contains password information (GRANT CONNECT, GRANT REMOTE DBA, CONNECT, or CREATE EXTERNLOGIN), the password information appears in the Command History dialog for the duration of the current Interactive SQL session.

When the command history is viewed in subsequent Interactive SQL sessions, passwords are replaced with ... in any of these statements that contain password information. For example, if you execute the following statement in Interactive SQL:

```
GRANT CONNECT TO testuser IDENTIFIED BY testpassword
```

the following appears in the Command History dialog in subsequent Interactive SQL sessions:

```
GRANT CONNECT TO testuser IDENTIFIED BY ...
```

When Interactive SQL saves the command history between sessions, it removes password information from the statements listed above.

Copying commands from the Command History dialog

You can copy a command to the clipboard by selecting it in the Command History dialog and then pressing CTRL+C or clicking the Copy button when the dialog has the focus. If you want to copy selected commands to the SQL Statements pane, click OK. When you copy multiple commands, they are separated by the command delimiter (a semicolon by default).

Removing commands from the Command History dialog

The contents of the Command History dialog persist between Interactive SQL sessions. You can remove commands from the dialog in one of two ways:

- ◆ Select one or more commands and click the Delete button or press the DELETE key to remove the selected command(s) from the dialog. This action cannot be undone.
- ◆ Remove all the commands from the dialog by clicking the Clear History button. This action cannot be undone.

You can also save commands in text files so that you can use them in a subsequent Interactive SQL session.

❖ **To save the command history to a file**

1. Open the Command History dialog.
2. Click the Save button or press CTRL+S.

- 
3. In the Save As dialog, specify a location and name for the file.  
The command history file has a *.SQL* extension.
  4. Click Save when finished.

## Logging commands

With the Interactive SQL logging feature, you can record commands as you execute them. Interactive SQL continues to record until you stop the logging process, or until you end the current session. The recorded commands are stored in a log file.

### ❖ To begin logging Interactive SQL commands

1. Choose SQL ► Start Logging.
2. In the Save As dialog, specify a location and name for the log file.
3. Click Save when finished.

### ❖ To stop logging Interactive SQL commands

1. Choose SQL ► Stop Logging.

#### **Tips**

You can also start and stop logging by typing in the SQL Statements pane. To start logging, type and execute **START LOGGING 'c:\filename.sql'**, where *c:\filename.sql* is the path, name, and extension of the log file. You only need to include the single quotation marks if the path contains embedded spaces. To stop logging, type and execute **STOP LOGGING**.  
Once you start logging, all commands that you try to execute are logged, including ones that do not execute properly.

## Summary

In this chapter, you learned how to start Interactive SQL, to execute SQL statements and to display database data.



---

## CHAPTER 15


# Managing Databases with Sybase Central

### About this chapter

This chapter introduces Sybase Central, the Adaptive Server Anywhere graphical database management tool. It provides a brief introduction to using Sybase Central for managing database properties.

You will learn how to start Sybase Central and connect to the sample database, how to view and edit the sample database, edit tables, manage users and groups, work with stored procedures, and back up your database.

Before you begin, be sure to make a copy of the sample database so you can restore it to its original form when you are finished.

 For more information about copying the sample database, see [“Lesson 1: Make a copy of the sample database”](#) on page 200.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About Sybase Central</a>	242
<a href="#">Lesson 1: Start Sybase Central and connect</a>	243
<a href="#">Lesson 2: The Sybase Central interface</a>	244
<a href="#">Lesson 3: Viewing the sample database</a>	246
<a href="#">Lesson 4: Create and edit tables</a>	251
<a href="#">Lesson 5: Manage users and groups</a>	253
<a href="#">Lesson 6: View and edit stored procedures</a>	255
<a href="#">Lesson 7: Back up your database</a>	258
<a href="#">Restore the sample database</a>	259
<a href="#">Summary</a>	260

---

## About Sybase Central

Sybase Central is a database management tool that provides Adaptive Server Anywhere database settings, properties, and utilities in a graphical user interface. Sybase Central can also be used for managing other Sybase products. This chapter describes how to use Sybase Central with the Adaptive Server Anywhere sample database.

Sybase Central provides an easy-to-use interface for two kinds of tasks.

- ◆ Tasks carried out by sending SQL statements to the server.
- ◆ Tasks carried out by Adaptive Server Anywhere utilities.

## Lesson 1: Start Sybase Central and connect

This section describes how to start Sybase Central and connect to the sample database using the ASA 9.0 Sample data source.

### ❖ To start Sybase Central

1. From the Start menu, choose Programs ► SQL Anywhere 9 ► Sybase Central.

The Sybase Central window opens.

### ❖ To connect to the sample database

1. From the Tools menu, choose Connect.
2. If a dialog prompts you to choose a plug-in, select Adaptive Server Anywhere 9, and then click OK.

The Connect dialog appears.

3. Select the ODBC Data Source Name option.
4. In the ODBC Data Source Name box, type **ASA 9.0 Sample**, or select it from the dropdown list if it is already there.

Alternatively, you can also click Browse and choose the ASA 9.0 Sample from the Data Source Names dialog, and then click OK.

5. Click OK to connect to the sample database.

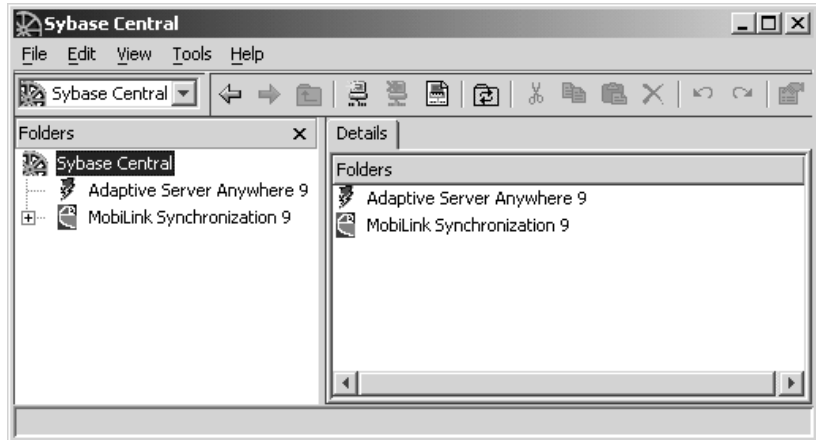
---

## Lesson 2: The Sybase Central interface

This section explains how to navigate the Sybase Central user interface.

### The Sybase Central main window

The Sybase Central main window:



#### Panes

The Sybase Central main window is split into two vertically-aligned panes.

The left pane displays a hierarchical view of database objects or **containers** in a tree-like structure. A container is a database object that can hold other database objects, including other containers.

The right pane displays the contents of the container that is selected in the left pane. You can change the appearance of both panes in the Options dialog (accessed through the Tools menu).

#### Toolbar

The Sybase Central toolbar provides you with buttons for common commands. To display or hide the toolbar, choose View ► Toolbar.

With the main toolbar, you can do the following:

- ◆ Navigate through the object tree.
- ◆ Connect to or disconnect from a database, server, or product module.
- ◆ Access the Connection Profiles dialog (also accessible through the Tools menu).
- ◆ Refresh the view of the current folder. You can also press F5 to refresh the folder.
- ◆ Cut, copy, paste, or delete objects.



- ◆ View the property sheet of a selected object.

As an easy reminder of what these buttons do, you can hold your cursor over each button to see a popup description.

Status bar

The status bar, which appears at the bottom of the main window, displays a brief summary of menu commands as you navigate through the menus. To display or hide the status bar, choose View ► Status Bar.

---

## Lesson 3: Viewing the sample database

With Sybase Central you can carry out many database administration tasks, including creating, deleting, and viewing tables, columns, and procedures; managing users and user groups; backing up your database; and modifying data in database tables. This section helps you explore the tables and other objects in the sample database.

### View the database schema

The database **schema** is a collection of all the objects in a database. Sybase Central displays a database schema as a hierarchy of containers and their contents. This section describes how to view the schema of a database.

#### Expanding a database container

Sybase Central offers a variety of methods for viewing the objects in a database, including the following:

- ◆ Click a container in the left pane to select it. The right pane then displays the contents of the selected container.
- ◆ Click once on the plus or minus icon next to a container in the left pane. This action expands or collapses the container. If no plus or minus icons appear next to a container, it means that the container holds no objects extending beyond the level of that container.
- ◆ Double-click a container in either pane. This action expands the tree in the left pane and displays the contents of the container in the right pane.

The left pane displays container objects only. The right pane displays the contents of the container object selected in the left pane. For example, when you select the Tables folder in the left pane, all of the tables within that folder appear in the right pane.

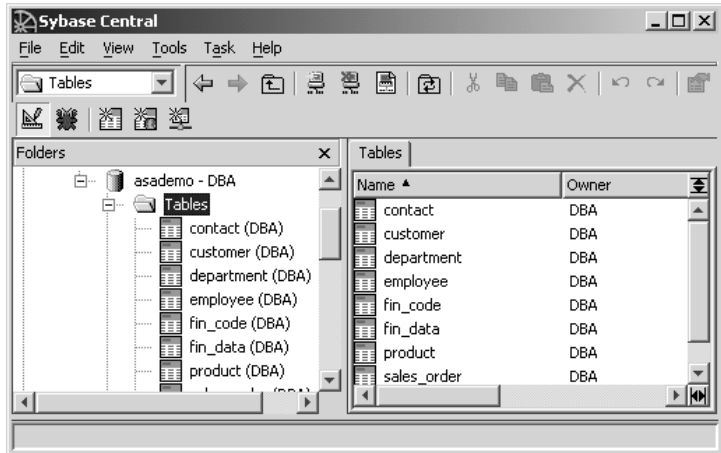
#### Viewing the tables in a database

Once you are connected to the sample database, you can open the folders in the left pane to view the tables and other objects that make up the database.

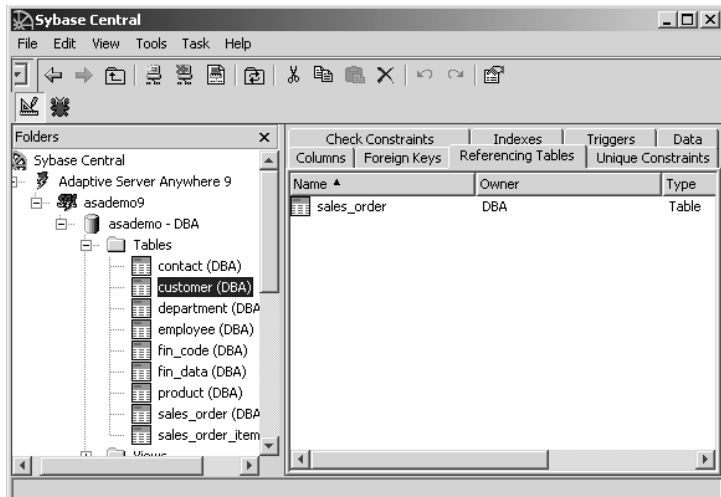
Each table has an **owner**, who is listed in parentheses beside the table name in the left pane. If you see more tables than appear in the figure, right-click the asademo database, and choose Filter Objects By Owner from the popup menu. Select DBA, and clear the other user IDs in the list. Click OK to restrict the objects displayed to those owned by DBA.

## ❖ To examine the tables in the sample database

1. Expand the Tables folder. You may have to expand asademo first.



2. Each table in the Tables folder is itself a container. Select a table in the left pane.
3. Click on the various tabs in the right pane to see the contents of that tab as it relates to the selected table.



Viewing database object properties

The properties of database objects, such as a database or a stored procedure, can be viewed using any of the following methods:

- ◆ Right-click a database object in the left pane and choose Properties from the popup menu.

- 
- ◆ Select a database object and choose File ► Properties.

You can navigate through a database schema by clicking or double-clicking items.

#### Database folders

Every Adaptive Server Anywhere database contains the following folders:

- ◆ **Tables** base tables and global temporary tables stored in the database.
- ◆ **Views** computed tables, stored in the database as a query and evaluated when accessed.
- ◆ **Indexes** indexes in the database that provide an ordering of the rows of a table on the basis of the values in some or all of the columns.
- ◆ **Triggers** special form of stored procedure executed automatically when a query that modifies data is executed.
- ◆ **System Triggers** triggers that implement referential integrity actions in the database.
- ◆ **Procedures & Functions** for using a module-based language consisting of SQL procedures.
- ◆ **Events** for creating and editing events.
- ◆ **Domains** for creating non-standard data types.
- ◆ **Users & Groups** for administering who is permitted to use the database.
- ◆ **Integrated Logins** for enabling users to connect to a database using their Windows user name and password.
- ◆ **SQL Remote Users** for administering SQL Remote replication of data in the database.
- ◆ **MobiLink Users** for managing MobiLink users and data synchronization.
- ◆ **Publications** database objects describing data to be replicated.
- ◆ **UltraLite Projects** for collecting SQL statements and table definitions used in UltraLite applications.
- ◆ **Dbspaces** for creating more than one *.db* file for the database.
- ◆ **Remote Servers** for identifying remote servers so that local users can execute remote procedure calls or retrieve information from the remote server.
- ◆ **Web Services** a list of web services currently available in the database.

You should explore the sample database until you are comfortable locating database objects in the Sybase Central main window.

## View the columns in the table

Each table has tabs in the right pane with information about a table's columns, foreign keys, indexes, triggers, and data. Here, you look only at the columns of the product table in the sample database.

### ❖ To view information about the columns of the product table

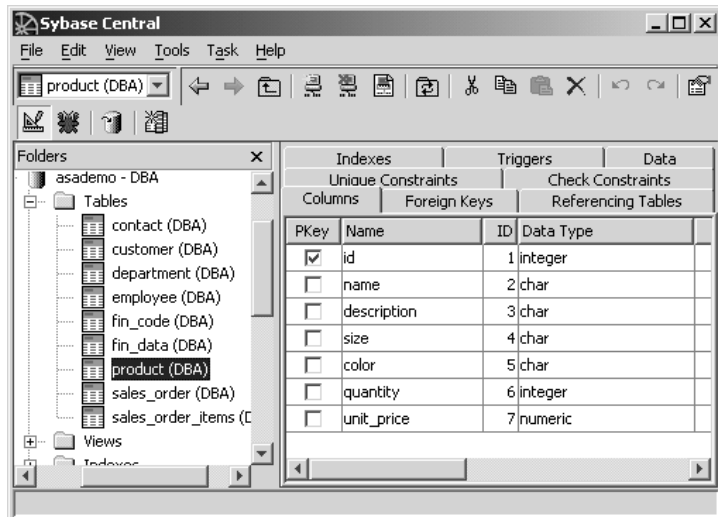
1. Open the product table.

In the left pane, expand the Tables folder, and then select the product table in the left pane.

Information about the product table appears on the tabs in the right pane.

2. In the right pane, click the Columns tab.

The name, data type, and other information about each column appears in the right pane.



3. View the properties of the color column.

Select the color column, and then from the File menu, choose Properties.

The color column property sheet appears. This property sheet contains a detailed description of the column. Some of the properties are useful for advanced users only, and are not described in this book.

---

## View the data in the database

In addition to viewing the database schema, you can view the data in the database tables from Sybase Central.

❖ **To view the contents of database tables**

1. Expand the Tables folder.
2. In the left pane, click the contact table.
3. In the right pane, click the Data tab.

The data in the table appears in the right pane.

## Lesson 4: Create and edit tables

This lesson takes you through the steps required to add a table to the sample database. This task includes adding columns to an existing table.

### Create a table in Sybase Central

In this lesson, you create a table of offices in the sample database.

#### ❖ Create a table named office

1. In the left pane of Sybase Central, click the Tables folder.
2. From the File menu, choose New ► Table.  
The Table Creation wizard appears.
3. Follow the instructions in the wizard, naming your table **office**, and choosing not to assign the table a primary key constraint. Use the default settings in the remainder of the wizard.
4. Name the first row **office\_id**, and assign a data type of smallint.
5. From the File menu, choose Save Table.

### Edit the Office table

This section describes how to edit an existing table. The steps show you how to add a primary key to the table created in the previous section by turning a column named `office_id` into a primary key.

#### ❖ Edit the Office table

1. Open the Tables folder and select the office table.
2. Place a checkmark in the PKey column to indicate that the column named `office_id` is the primary key for the table.

##### Primary key conditions

A column with a primary key constraint cannot contain Nulls. Thus, when you select the Primary Key checkbox, the checkmark in the Nulls checkbox is automatically removed from that column.

Similarly, if you select the Nulls checkbox, the primary key constraint is automatically removed from that column.

3. From the File menu, choose Save Table.
4. From the File menu, choose New Column.

- 
5. Edit the new column so that its name is `office_name`, its type is `char`, and its size is 20. Select Nulls so that `office_name` is allowed to be blank.
  6. From the File menu, choose Save Table.

The column is now in the database, although it contains no data.

#### Notes

- ◆ All objects in Sybase Central have property sheets, including tables, users, and stored procedures.
- ◆ Whenever you select an object in Sybase Central, commands related to that object appear in several places: in the File menu, in a popup menu that you can access by right-clicking the object, and as buttons on the toolbar.
- ◆ When you click on many of the folders, **New *item*** buttons appear in the toolbar. Clicking these buttons creates new objects for the database.
- ◆ Once you open a container, its contents are cached to improve performance. You can access two refresh commands in the View menu to update the view of either the current container or the entire window.

## Delete the office table

Tables can be deleted, or dropped, from a database. For example you can delete the office table to restore the sample database to its original state.

### ❖ To delete the office table from the sample database

1. Open the Tables folder.
2. Select the office table and from the Edit menu, choose Delete.  
The Confirm Delete dialog appears.
3. Click Yes to delete the office table from the database.



## Lesson 5: Manage users and groups

In Adaptive Server Anywhere, both users and groups are objects within the database. However, groups are also containers, capable of containing users and other groups. When users are contained within a group, they are members of that group.

In this structure, permissions granted to a group are inherited by all users and groups contained within. Adaptive Server Anywhere allows you to create users and groups with permission to use a database and grant membership to other groups. Users and groups can be members of multiple groups.

This section demonstrates how to create a group for the database, how to create an individual user, and how to make the user a member of the group.

### Add a group to the sample database

This section describes how to add a group to the sample database.

#### ❖ To add a group to the sample database

1. Connect to the sample database.
2. Select the Users & Groups folder in the left pane.
3. From the File menu, choose New ► Group.  
The Group Creation wizard appears.
4. Follow the instructions in the wizard, naming the new group **Sales**, and leaving the rest of the options at their default settings.

The Sales group now appears in both panes.

### Add a user to the database

This section describes how to add a user to the sample database.

#### ❖ To add a user to the sample database

1. Connect to the sample database.
2. Select the Users & Groups folder in the left pane.
3. From the File menu, choose New ► User.
4. Follow the instructions in the User Creation wizard, naming the new user **Sandy**, and using the name **Sandy** as the password. Leave the rest of the settings at their default values.

---

An icon for the new user appears in both panes.

## Add a user to a group

This section describes how to add two users to a group by copying and pasting them.

### ❖ To add users to a group

1. Open the Users & Groups folder in the left pane.
2. Click the Sales group.
3. Click the Members tab in the right pane.
4. From the File menu, choose New ► Members.  
The New Members dialog appears.
5. Choose the user Sandy, then click OK.

## Lesson 6: View and edit stored procedures

The Procedures & Functions folder holds stored procedures for a database. This lesson shows how to view and alter the contents of a procedure, as well as how to modify the procedure's properties.

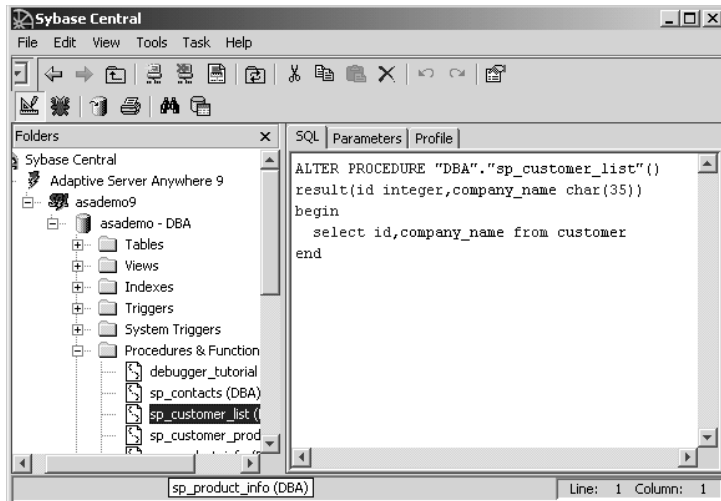
### Viewing Stored Procedures

There are several stored procedures included in the sample database. The following steps explain how to view and edit stored procedures.

#### ❖ To view the contents of a stored procedure

1. In the left pane of Sybase Central, open the Procedures & Functions folder in the sample database.
2. Click the `sp_customer_list` stored procedure.

You can view the text of the procedure called `sp_customer_list` on the SQL tab in the right pane:



This procedure is designed to return a set of customer IDs and company names from the customer table.

### Edit stored procedure properties

You can edit the properties of stored procedures using the stored procedure's property sheet. In this section, you alter the permissions for a stored procedure so that all salespeople, including the user you created in the

---

previous lesson (Sandy), can execute the procedure to obtain a list of customers.

Stored procedures have permissions associated with them. In order to execute a procedure, you either need to be granted permission to execute it, or you need to be a member of a user group that has permission to execute it.

❖ **View and alter the permissions on the sp\_customer\_list procedure**

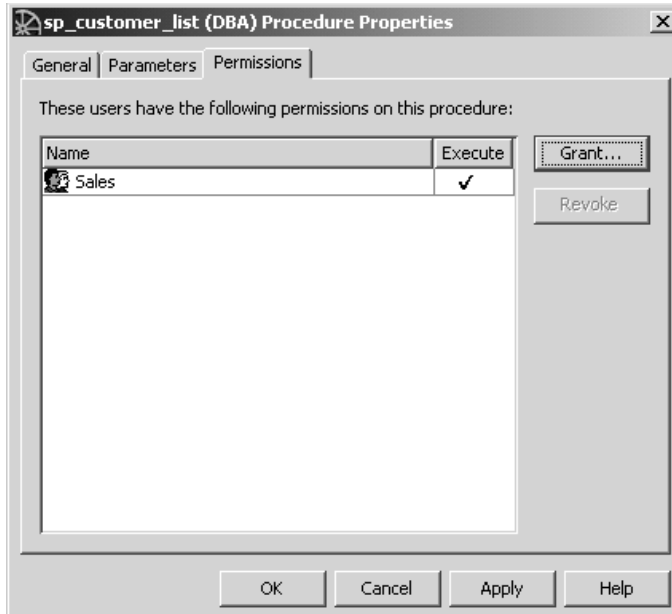
1. In the Procedures & Functions folder, select sp\_customer\_list.
2. Choose File ► Properties.
3. Click the Permissions tab of the property sheet to see which user IDs have been granted permissions for this procedure. Currently, none have since the only user for the sample database is DBA, who is the owner of the procedure and so automatically has execute permission for the procedure.
4. Click Grant.

The Grant Permissions dialog appears:



5. In the Grant Permission dialog, select Sales and then click OK to grant this group permission to execute the sp\_customer\_list procedure.

Sales is added to the Permissions tab of the Procedure property sheet with a checkmark in the Execute column:



You can revoke this group's permission by clicking the checkmark to make it disappear, or by selecting the group and clicking Revoke.

6. Click OK to accept the changes to the sp\_customer\_list permissions.

---

## Lesson 7: Back up your database

Utilities

Sybase Central includes a set of database utilities for carrying out common database administration tasks. Wizards walk you through the steps involved.

To see a list of those utilities that can be used on a running database, select the sample database container, and then click the File menu. Several options appear, including Backup Database, Upgrade Database, and Validate Database. Each of these menu items represents a utility.

In this section, you use a wizard to back up the sample database. This can be done on a running database.

### ❖ To back up a running database

1. In the left pane, right-click the asademo - DBA icon and choose Backup Database from the popup menu. The Backup Database wizard appears.
2. Read the introductory page of the wizard and then click Next.
3. Select asademo from the list of databases you can back up. Click Next.
4. Select On Disk, In The Following File.

Type a filename in the text box indicating where you want to back up the database to. As this is a tutorial, you may wish to choose a file in a temporary directory such as `c:\temp\backup`.

5. Click Finish to back up the database.

Notes

This kind of backup is called an **archive backup**. You can also make backups called **image backups**, which are physical copies of the database file and associated files.

An extension of `.1` is added to the filename you specify in the Backup Database wizard.

Wizards are available for several other database administration tasks. You may wish to try creating a database by selecting the server in the left pane and choosing Create Database from the File menu.

## Restore the sample database

Now that you have completed this tutorial, you should restore the sample database so that it can be used again in its original form. Replace the version of *asdemo.db* that you just changed with the copy you made before beginning the tutorial. Delete *asdemo.log*.

---

## Summary

In this chapter, you learned how to start Sybase Central and connect to the sample database. You also learned how to view and edit the sample database, edit tables, manage users and groups, work with stored procedures, and back up your database.



---

## CHAPTER 16

# Designing and Building a Database

### About this chapter

This chapter introduces some principles of database design, and describes how to create a database using Sybase Central. It uses the Adaptive Server Anywhere sample database to illustrate the principles involved, and applies skills acquired in the preceding chapters working with the sample database to creating a brand new database from scratch. You will design and build a very simple database, modeled on the product, sales\_order\_items, sales\_order, and customer tables of the sample database, but simplified.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About designing a new database</a>	262
<a href="#">Designing and building a simple database</a>	263
<a href="#">Lesson 1: Create a database file</a>	264
<a href="#">Lesson 2: Connect to your database</a>	266
<a href="#">Lesson 3: Design and create a table</a>	267
<a href="#">Lesson 4: Identify and create primary keys</a>	269
<a href="#">Lesson 5: Design column properties</a>	271
<a href="#">Lesson 6: Design and create relationships between tables</a>	274
<a href="#">Summary</a>	277

---

## About designing a new database

It is worth spending some time designing even the simplest database: what tables you will have, the keys that relate these tables, and so on.

Designing a database is not a difficult task for small and medium sized databases, but it is an important one. Bad database design can lead to an inefficient and possibly unreliable database system. Database applications are built to work on specific parts of a database, and rely on the database design, so a bad design can be difficult to revise at a later date.

Designing a large database is a complex task. There are formal approaches, such as conceptual data modeling, that help you to create efficient designs. Powerful tools, such as Sybase PowerDesigner and DataArchitect, enable you to apply these techniques to design and maintain large database designs.

This chapter does not attempt to tackle design issues for large databases. Instead, it helps you decide the kind of information you group together in a single table, and the way in which to think about and classify relationships between tables.

☞ For an elementary look at the principles of database design, see [“Designing Your Database”](#) [*ASA SQL User’s Guide*, page 3]. For more advanced treatments, see the Sybase PowerDesigner documentation or a book devoted to database design.

## Designing and building a simple database

When designing a database, you plan what items you want to store information about, and what information you will keep about each item. You also determine how these items are related. You classify the things in your database as entities; the links between these entities are called **relationships**.

---

# Lesson 1: Create a database file

In this lesson, you create a database file to hold your database.

☞ For more information, see [“The pieces of a database system” on page 123](#).

## Concepts

A database file is a container, ready to hold your database. The database file contains system tables and other system objects that are common to all databases, but you must add tables and the data they hold.

The collection of tables, indexes, and so on within the database, and all the relationships between them, make up the database **schema**. The schema is the database without the data. This tutorial describes how to design and create a very simple database schema.

☞ For a conceptual introduction to some of these pieces, see [“Relational database concepts” on page 118](#).

The name of each object within the database, including tables, columns, and indexes, is an **identifier**. There are rules governing what you can use as identifiers. You can use any set of letters, numbers, or symbols. However, you must enclose a column name in double quotes if it contains characters other than letters, numbers, or underscores, if it does not begin with a letter, or if it is the same as a keyword.

If the QUOTED\_IDENTIFIER database option is set to OFF, double quotes are used to delimit SQL strings and cannot be used for identifiers. However, you can always use square brackets to delimit identifiers, regardless of the setting of QUOTED\_IDENTIFIER.

☞ For more information about identifiers, see [“Identifiers” \[ASA SQL Reference, page 7\]](#).

## Exercise

### ❖ To create a new database file

1. Start Sybase Central.
2. In the left pane, select the Adaptive Server Anywhere plug-in, then click the Utilities tab in the right pane.
3. In the right pane, double-click Create Database.  
The Create Database wizard opens.
4. Read the information on the introductory page and click Next.
5. Select Create A Database On This Computer and click Next.

6. Choose a location and name for your database file.

Enter the filename `c:\temp\mysample`. If your temporary directory is somewhere other than `c:\temp`, specify the appropriate path.

7. Click Finish to create the database.

Other options are available when creating a database, which you could have viewed by clicking Next instead of Finish, but the default choices are good for many purposes.

8. The Creating Database window displays the progress of the task. When the file is created, click OK to close the window.

## Lesson 2: Connect to your database

In this lesson, you connect to the database file you created.

☞ For more information, see [“How the pieces fit together”](#) on page 125.

### Exercise

Once your database is created, you can connect to it in order to create tables and other database objects.

#### ❖ To connect to your database

1. Start Sybase Central.
2. From the Tools menu, choose Connect to open the Connect dialog.  
If you have multiple plug-ins loaded, you may need to choose the Adaptive Server Anywhere plug-in before the Connect dialog opens.

3. Specify the user ID and password.

On the Identification tab, enter a User ID of **DBA** and a Password of **SQL**. These are the default values for new databases.

Choose **None** in the profile options at the bottom of the tab.

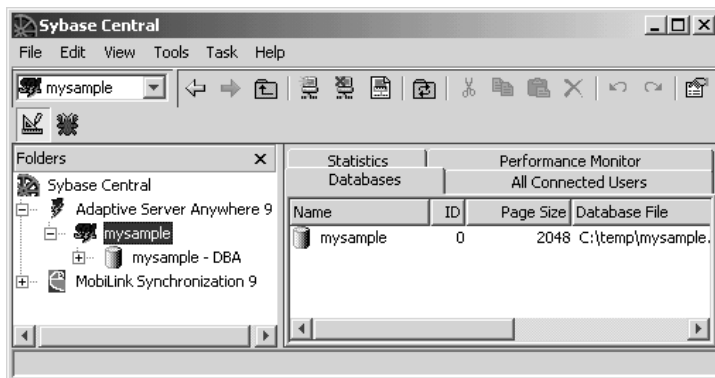
4. Specify your database file.

Click the Database tab. Enter the full path of your database file in the Database File field. For example, if you followed the suggestion in the previous lesson, you should enter the following:

```
c:\temp\mysample.db
```

5. Connect to the database.

Click OK. Sybase Central connects to the database.



6. Open the database server container in the left pane to see the mysample database.

## Lesson 3: Design and create a table

In this lesson, you add a table to your database.

### Concepts

Each table in your database should contain information about a single subject. In the language of database design, you are identifying **entities**. For example, the sample database holds information about employees in one table, and information about products in another table: employees and products are entities within the database design.

☞ For an introduction to tables, see [“Database tables” on page 118](#).

Each column in a table describes a particular characteristic of the thing that you would like to store information about. For example, in the sample database, the employee table has columns that hold an employee ID number, first and last names, an address, and other particular information that pertains to a particular employee.

In database diagrams, each table is depicted by a rectangular box. The name of the table is at the top of the box, and the column names are listed inside the box.

Product	
<u>ID</u>	Integer
name	char(15)
description	char(30)
size	char(18)
color	char(6)
quantity	integer
unit_price	numeric(15,12)

In the product table from the sample database, above, each product is an item of sports clothing.

### Exercise

Create a simplified version of the product table, containing only the identifier (id) and name columns.

#### ❖ To create the product table

1. In Sybase Central, connect to your database if you are not already connected.
2. Open the Tables folder in your database.  
First open the server and database containers, then open the Tables folder.
3. From the File menu, choose New ► Table.  
The Table Creation wizard appears.

- 
4. Name your table **product**.
  5. Click Finish.
  6. The first column of the table automatically appears in the right-hand pane. Define the first column in your table with the following values:
    - ◆ **PKey** Select the checkbox beside the column name so that a checkmark appears, indicating that the column is a **primary key**.
    - ◆ **Column Name** Give the column a name of **id**.
    - ◆ **Data Type** Give the column the **integer** data type.
  7. Create an additional column.

From the File menu, choose New Column and add a column with the following properties:

    - ◆ **Column Name** Give the column a name of **name**. This column holds the product name.
    - ◆ **Data Type** Give the column the **char** data type, which holds character strings.
    - ◆ **Size** Enter a maximum length of **15** in the Size column.
  8. From the File menu, choose Save Table.

You have now created a table in your database. The table data is held in the database file. At present, the table is empty.

The next two lessons have more to say about columns and data types.



## Lesson 4: Identify and create primary keys

In this lesson, you learn more about defining primary keys for your tables. There is no exercise associated with this lesson.

For more information, see “[Tables have a primary key](#)” on page 119.

### Concepts

The **primary key** is a special column or columns used to uniquely identify a row in a table. In the product table, the id column uniquely identifies each product.

Product	
<u>ID</u>	<u>Integer</u>
name	char(15)
description	char(30)
size	char(18)
color	char(6)
quantity	integer
unit_price	numeric(15,12)

Each row has a unique value for the id column, and the values in each row pertain only to a single product identifier by its id value. Two products might have the same name or the same size, but not the same id number. In the diagram, the id column is underlined to show that it is a primary key.

Creating a column specifically to hold an identifier which has no other meaning is common practice in database design. You will know from your bank, utility, or credit card statements that each account has a unique identifier.

### Using an AUTOINCREMENT primary key

You can make entering primary keys simple by assigning a primary key column a default value of AUTOINCREMENT. The value for this column is entered automatically each time a new row is added, and its value is one more than the field’s value for the last row added.

#### ❖ To create an AUTOINCREMENT primary key

1. Select the product table in the left pane, then select the Columns tab in the right pane.
2. Select the primary key column. From the File menu, choose Properties to open the property sheet for the column.
3. Click the Value tab.
4. Select the Default Value option.
5. Click System-defined, and choose Autoincrement from the dropdown list.

- 
6. Click OK to close the column property sheet.
  7. From the File menu, choose Save Table.

## Lesson 5: Design column properties

In this lesson, you learn more about choosing data types and other attributes for the columns of your tables.

### Concepts

Each column has a data type associated with it. The **data type** defines the type of information the column holds. Choose a data type for the column that is appropriate for the data in the column. For example, identifier columns commonly have an integer data type, while columns holding names or addresses must have character data types.

Data types are organized into the following categories:

- ◆ **Numeric data types** There are several numeric data types. Some are exact (not affected by round-off errors during operations) and some are approximate.

The data type of the column affects the maximum size of the column. For example, if you specify SMALLINT, a column can contain a maximum value of 32,767. If you specify INTEGER, the maximum value is 2,147,483,647.

☞ For a complete list, see “[Numeric data types](#)” [*ASA SQL Reference*, page 58].

- ◆ **Character data types** These are used to hold strings of text, such as names, addresses, and so on. These data types have a length indicating the maximum length of string that can be stored in them.

☞ For a list, see “[Character data types](#)” [*ASA SQL Reference*, page 54].

- ◆ **Binary data types** These can be useful to hold information that may be meaningful to an application, but is encoded in a binary format.

☞ For a list, see “[Binary data types](#)” [*ASA SQL Reference*, page 74].

- ◆ **Date/time data types** These hold times of the day, as well as dates.

☞ For a list, see “[Date and time data types](#)” [*ASA SQL Reference*, page 67].

- ◆ **Long data types** These are sometimes called blobs (binary large objects). They can be used to hold long strings of text (called memo fields in some databases), images, or other binary information.

☞ For more information, see “[LONG BINARY data type \[BINARY\]](#)” [*ASA SQL Reference*, page 74], and “[LONG VARCHAR data type \[Character\]](#)” [*ASA SQL Reference*, page 56].

In addition, Adaptive Server Anywhere supports user-defined data types and special Java data types. These are not discussed in this introductory book.

---

## NULL and NOT NULL

If every row must contain a value for this column, you should define the column as being NOT NULL. Otherwise, the column is allowed to contain NULL, which represents a missing value. The default is to allow NULL, but you should explicitly declare columns NOT NULL unless there is a good reason to allow NULL.

☞ For a complete description of the NULL value, see “NULL value” [ASA SQL Reference, page 49]. For information on its use in comparisons, see “Search conditions” [ASA SQL Reference, page 23].

### ❖ To specify a data type for a column

1. Select the product table in the left pane, then select the Columns tab in the right pane.
2. Select the Id column.
3. From the File menu, choose Properties to open the property sheet for the column.  
The column’s property sheet opens.
4. On the Data Type tab, select Integer from the Built-in Type dropdown list.
5. Click OK.

## Exercise

This lesson and the last lesson have introduced the basic concepts you need to know in order to create database tables. You can put these to work by adding some more tables to your database. These tables will be used in the subsequent lessons in this chapter.

Add the following tables to your database:

- ◆ **customer** Add a table named customer, with the following columns:
  - **id** An identification number for each customer. This column has **integer** data type, and is the **primary key**. Make this an autoincrement key.
  - **company\_name** The company name. This column is a **character** data type, with a maximum length of **35** characters.
- ◆ **sales\_order** Add a table named sales\_order, with the following columns:
  - **id** An identification number for each sales order. This column has **integer** data type, and is the **primary key**. Make this an autoincrement key.
  - **order\_date** The date on which the order was placed. This column has **date** data type.

- **cust\_id** The identification number of the customer who placed the sales order. This column has **integer** data type.
- ◆ **sales\_order\_items** Add a table named `sales_order_items` to hold line item information, with the following columns:
  - **id** The identification number of the sales order of which the line item is a part. This column has **integer** data type, and should be identified as a **primary key** column.
  - **line\_id** An identification number for each sales order. This column has **integer** data type, and should be identified as a **primary key** column.
  - **prod\_id** The identification number for the product being ordered. This column has **integer** data type.

You have now created four tables in your database. The tables are not yet related in any way. In the next lesson, you define foreign keys to relate the tables to one another.

---

## Lesson 6: Design and create relationships between tables

In this lesson, you learn about designing and creating relationships between tables, using foreign keys.

☞ For more information, see “[Tables are related by foreign keys](#)” on [page 120](#).

### Concepts

Although each table contains information about a single subject, two or more tables may contain related information. For example, an employee is a member of a department, and a sales order is comprised of a set of products. Relationships in a database may appear as **foreign key** relationships between tables, or may appear as separate tables themselves. You will see examples of each in this chapter.

You create relationships in your database to encode rules or practices that govern the data in your tables. Once a relationship is built into the structure of the database, there is no provision for exceptions.

Relationships among tables are classified as follows.

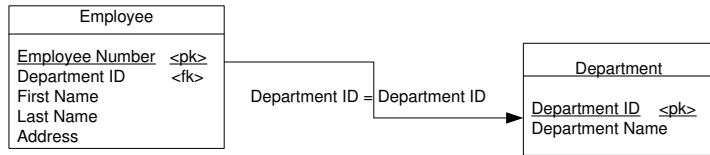
- ◆ **One-to-one relationships** Each item in one entity corresponds to either zero or one entity in another. For example, in the sample database, *one* employee manages *one* department. There is nowhere to put a second department manager. Duplicating the department entry would involve duplicating the department ID, which is not possible because it is the primary key.

It is often appropriate to combine the items in a one-to-one relationship into a single table. There is a column in the department table for a manager, rather than having a separate table named manager.

☞ For cases where it is appropriate to keep the items separate, see “[Designing Your Database](#)” [[ASA SQL User's Guide, page 3](#)].

- ◆ **Many-to-one relationships** A many-to-one relationship becomes a foreign key relationship between tables. In a many-to-one relationship, the primary key in the *one* entity appears as a new foreign key column in the *many* table.

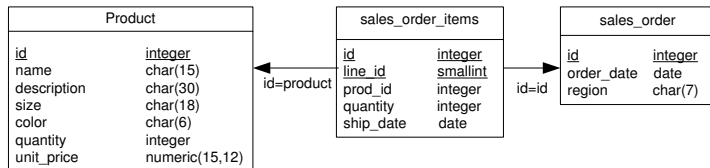
For example, in the database you just created, *one* customer can place *many* orders, but only one customer places each order. To represent the one-to-many relationship, you need a **foreign key column** in the `sales_order` table (`cust_id`) that maps to the primary key column in the customer table (`id`). It is often convenient to give the two columns the same name.



Each entry in the `cust_id` column of the `sales_order` table must match one of the entries in the `id` column of the `customer` table. The `sales_order` table (which contains the foreign key in the relationship) is called the foreign table or referencing table. The `customer` table (which contains the referenced primary key) is called the primary table or the referenced table.

- ❖ **Many to many relationships** A many-to-many relationship is represented by an intermediate table, and there is a foreign key relationship from the intermediate table to each of the related entities.

For example, in the sample database, there is a many-to-many relationship between products and sales orders. One sales order can be for many products, and one product can appear on many sales orders.



In some cases, the intermediate table (`sales_order_items`) contains additional information, such as the number of items of the product that were ordered and the date they were shipped. In this case, the intermediate table holds no additional information.

## Exercise

Add foreign keys to relate the tables in your database.

### ❖ To create a foreign key

1. Select the table for which you wish to create a foreign key.
2. Select the Foreign Keys tab in the right pane.
3. From the File menu, choose **New** ► **Foreign Key** to open the Foreign Key Creation wizard.
4. Follow the instructions in the wizard to add the following foreign key:
  - ❖ A foreign key from the `id` column in `sales_order_items`, referencing the `id` column in `sales_order`. This key builds the many-to-one relationship between sales orders and sales order items into the database.
5. Repeat steps 1 through 4 to create the following foreign keys:

- 
- ◆ A foreign key from the `prod_id` column in `sales_order_items`, referencing the `id` column in `product`. This key builds the many-to-one relationship between sales order items and products into the database.
  - ◆ A foreign key from the `cust_id` column in `sales_order`, referencing the `id` column in `customer`. This key builds the many-to-one relationship between sales orders and customers into the database.

The first two foreign keys taken together build the many-to-many relationship between sales orders and products into the database.

This completes this introductory section on designing and building relational databases. Remaining chapters in the book describe how to add and retrieve data from databases. These chapters use the Adaptive Server Anywhere sample database, which is a bigger database than the one you have just created.



## Summary

In this chapter, you learned the principles of database design. You then applied those principles of design in creating a brand new database using Sybase Central.



---

## CHAPTER 17

# Synchronizing Databases with MobiLink

### About this chapter

This chapter provides a tutorial to guide you through the process of synchronizing two Adaptive Server Anywhere databases via MobiLink. One of these databases is the consolidated database, and the other is a remote database. In this tutorial, you create these databases and then synchronize the two.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About MobiLink</a>	280
<a href="#">Introduction</a>	281
<a href="#">Lesson 1: Create your databases</a>	282
<a href="#">Lesson 2: Prepare the databases for synchronization</a>	286
<a href="#">Lesson 3: Start the MobiLink synchronization server</a>	289
<a href="#">Lesson 4: Run the MobiLink synchronization client utility</a>	290
<a href="#">Summary</a>	291

---

## About MobiLink

Data replication is the sharing of data among physically distinct databases. Sybase provides three distinct technologies for data replication:

- ◆ MobiLink
- ◆ SQL Remote
- ◆ Replication Server

MobiLink synchronization enables replication between an ODBC-compliant consolidated database and Adaptive Server Anywhere or UltraLite remote databases. In this tutorial an Adaptive Server Anywhere remote database is used. The consolidated database can be made by Sybase Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, Oracle, Microsoft SQL Server, or IBM DB2.

MobiLink is designed for synchronization involving a consolidated data server and large numbers of remote databases, typically including many mobile databases. Administration and resource requirements at the remote sites are minimal. The system is connection-based and a remote site can connect as often as desired. At the end of each connection, the databases are fully synchronized.

MobiLink works by lumping the results of multiple transactions on the remote database into one set of changes applied to the consolidated database. Since synchronization always occurs at a transaction boundary, referential integrity is preserved. The order of the individual changes made during the component transactions is not preserved: since uncommitted data is never replicated, data integrity is preserved.

- For more information about synchronization strategies, including complete MobiLink documentation, see the [MobiLink Administration Guide](#).
- For an introduction to SQL Remote synchronization, see “[Replicating Data with SQL Remote](#)” on page 293.

## Introduction

In this tutorial, you create a consolidated database and a remote database. You write synchronization publications and subscriptions. You then synchronize these databases using MobiLink synchronization technology.

### Requirements

The tutorial requirements are given below.

Requirement	Discussion
Timing	The tutorial should take 50 minutes.
Software	<ul style="list-style-type: none"> <li>◆ A full Adaptive Server Anywhere 9 installation.</li> <li>◆ A full installation of MobiLink synchronization server.</li> </ul>
Competencies and experience	<p>Knowledge and/or experience with command processing.</p> <p>Competent at connecting to your database using ODBC and Sybase Central.</p> <p>For more information on ODBC, see <a href="#">“Lesson 1: Create an ODBC data source”</a> on page 210.</p>

### Goals

The goals for the tutorial are to gain competence and familiarity with:

- ◆ The MobiLink synchronization server and client as an integrated system
- ◆ Executing MobiLink synchronization server and client commands
- ◆ The MobiLink synchronization server and client commands and options.

### Key concepts

The MobiLink synchronization server connects to the consolidated database using the ODBC interface. The MobiLink synchronization client connects to your remote database. The MobiLink synchronization server and client function as a pair, managing the upload and download of data from one database to another.

Important concepts you will learn in this tutorial include:

- ◆ MobiLink synchronization server, MobiLink synchronization client
- ◆ ODBC connection, ODBC data source, synchronization subscription and publication, consolidated server, remote databases.

---

## Lesson 1: Create your databases

MobiLink synchronization requires that you have compatible consolidated and remote databases, data in database tables, and ODBC data sources for each database.

Create your database files

The first step is to create each of the databases. In this procedure, you build a consolidated database and a remote database using the *dbinit* executable from the command prompt.

### Tip

Creating a database file using *dbinit* is similar to formatting a disk — you have a database file with no user tables or procedures. You create your database schema when you define, within the newly initialized file, various user-defined tables and procedures.

For more information on the *dbinit* utility, see “[Creating a database using the dbinit command-line utility](#)” [*ASA Database Administration Guide*, page 531].

### ❖ To create your database files

1. Create the directory where all of the tutorial files will be stored.

From the C:\ drive, at the command prompt type:

```
mkdir MLTutorial
```

2. Go to the directory you made.

At the command prompt type:

```
cd MLTutorial
```

3. Create a file for your consolidated database.

At a command prompt type:

```
dbinit consol.db
```

4. Create a file for your remote database.

At a command prompt type:

```
dbinit remote.db
```

5. Verify the successful creation of these database files by typing the following at a command prompt:

```
dir
```

Your database files should appear in the directory listing. If not, review your procedures and repeat Steps 1 or 2 as needed.

## Create ODBC data sources

☞ You are now ready to build ODBC data sources through which you can connect to your Adaptive Server Anywhere databases.

☞ For more information on creating ODBC data sources see “The Data Source utility” [*ASA Database Administration Guide*, page 510].

## ❖ To create ODBC data sources

1. Create your ODBC data source for a consolidated database by typing the following from a command prompt:

```
dbdsn -w test_consol -y -c "uid=DBA;pwd=SQL;dbf=C:\
MLTutorial\consol.db;eng=Consol"
```

2. Create an ODBC data source for a remote database by typing the following from a command prompt:

```
dbdsn -w test_remote -y -c "uid=DBA;pwd=SQL;dbf=C:\
MLTutorial\remote.db;eng=remote"
```

Now you can verify your data sources.

## ❖ To verify your new data sources

1. Start the ODBC Administrator.
 

From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► ODBC Administrator.

The ODBC Data Source Administrator appears.
2. Click the User DSN tab.
3. Scroll through the list to find your new data sources.
4. Select a data source and click Configure.
 

The ODBC Configuration for Adaptive Server Anywhere dialog opens.
5. On the ODBC tab, test your data source by clicking the Test Connection button.

The MobiLink synchronization server and the MobiLink synchronization client use the ODBC data sources to connect to the consolidated and remote databases, respectively.

## Populate your databases

You can now create tables for each of your newly initialized databases by executing SQL statements in scripts using Interactive SQL. The scripts contain SQL statements that create tables in the consolidated and remote databases and insert data. The scripts also create synchronization subscriptions and publications on the remote.

---

## ❖ To run scripts from Interactive SQL

### 1. Start Interactive SQL.

From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL

or

From a command prompt, type *dbisql*.

### 2. Connect to the consolidated database.

From the SQL menu, select Connect.

Type the default user ID **DBA** and the default password **SQL** in the appropriate fields.

Select the ODBC Data Source Name option and use Browse to select the test\_consol data source.

Click OK to connect to the database.

### 3. Create a table in the consolidated database and add some rows to the table:

#### ◆ Enter the following instructions:

```
CREATE TABLE cust (
    cust_id int default autoincrement primary key,
    emp_id int,
    cust_name varchar( 128 )
);
-- add data to cust table
INSERT INTO cust ( emp_id, cust_name ) VALUES ( 1,
    'cust1' );
INSERT INTO cust ( emp_id, cust_name ) VALUES ( 1,
    'cust2' );
INSERT INTO cust ( emp_id, cust_name ) VALUES ( 2,
    'cust3' );
COMMIT;
```

### 4. Verify the successful creation of the table:

Enter the following command and verify that three rows are returned:

```
SELECT * FROM cust
```

### 5. Now repeat steps 2, 3, and 4 for the remote database *remote.db*, using the following SQL statements. No rows are added to the table in the remote database.

```
CREATE TABLE cust (cust_id int default autoincrement primary
    key,
    emp_id int,
    cust_name varchar( 128 )
)
```



The query `SELECT * FROM cust` should display no rows.

6. Close both instances of Interactive SQL.

**Tip**

If you need to start your database at any time, the following commands, for the consolidated and remote databases, can be used from the `C:\MLTutorial` directory: `dbeng9 c:\MLTutorial\consol.db` and `dbeng9 c:\MLTutorial\remote.db`

---

## Lesson 2: Prepare the databases for synchronization

Synchronization is governed by the following:

- ◆ **Synchronization publications, users, and subscriptions** These are defined in each remote database.
- ◆ **Synchronization scripts** These are written in SQL and held in the consolidated database. Alternatively, you can write synchronization scripts in Java and store them in a location accessible by the MobiLink synchronization server. In this tutorial we use SQL scripts.

You can write, view and modify synchronization scripts as well as publications and subscriptions using Sybase Central.

Create a synchronization subscription and publication

The MobiLink synchronization publication, user, and subscription are necessary for MobiLink synchronization to happen. Each is defined in the remote database.

### ❖ To add a publication and synchronization subscription to the remote database

1. Start Sybase Central and connect to your remote database:
  - ◆ From the Start menu, select Programs ► SQL Anywhere 9 ► Sybase Central.
  - ◆ In the left pane, select the Adaptive Server Anywhere 9 plug-in. From the Tools menu, choose Connect.
  - ◆ Enter an ODBC data source name of test\_remote and click OK to connect.
2. Add a publication to the remote database:
  - ◆ In Sybase Central, open the remote database.
  - ◆ Open the Publications folder.
  - ◆ From the File menu, choose New ► Publication. The Publication Creation wizard appears.
  - ◆ Name the publication Customer and click Next.
  - ◆ Double-click the table **cust** to add it to the list of selected tables and click Finish to create the publication.
3. Add a MobiLink user name to the remote database:
  - ◆ In Sybase Central, open the remote database.
  - ◆ Open the MobiLink Users folder.

- ◆ From the File menu, choose New ► MobiLink User. The MobiLink User Creation wizard appears.
  - ◆ Name the user **ml\_user** and click Finish to create the user.
4. Subscribe the MobiLink user to the publication:
- ◆ In the left pane, expand the Publications folder so that the Customer publication is visible.
  - ◆ In the left pane, expand the MobiLink Users folder so that the ml\_user MobiLink user is visible.
  - ◆ Drag the MobiLink user onto the Customer publication.
  - ◆ You are asked whether to subscribe the user to the publication. Click OK.
5. Add address information to the subscription:
- Address information is used to enable the MobiLink client to locate the correct MobiLink synchronization server.
- ◆ In the left pane, select the MobiLink user ml\_user and from the File menu, choose Properties.
  - ◆ On the Connection tab, enter the following information, leaving the other fields blank:
    - **Host** localhost.  
If the MobiLink synchronization server were running on a different machine, you would enter the machine name or IP number instead of **localhost**.
  - ◆ Ensure that the Enable Certicom Security checkbox is clear.
  - ◆ Click OK to add the address information.

This completes the preparation of the remote database. The next step is to add synchronization scripts for the consolidated database.

Add synchronization scripts to the consolidated database

Each script belongs to a designated **script version**. You must add a script version to the consolidated database before you add scripts.

#### ❖ To add a script version

1. Start Sybase Central and connect to the test\_consol data source using the MobiLink plug-in.
2. Expand the test\_consol folder.
3. Open the Versions folder.
4. Double-click Add Version. Name the new version **default**.

---

### ❖ To add synchronized tables to your consolidated database

1. Open the Tables folder beneath the test\_consol icon.
2. Open the DBA container.
3. From the File menu, select Add to Synchronized Tables.

Now that you have designated these tables as tables to be synchronized, you can add a new table script for each upload and download to the consolidated database.

### ❖ To add table scripts to each synchronized table

1. Open the Synchronized Tables folder, and double-click the cust table.
2. Double-click Add Table Script in the right pane.  
The Add Synchronizing Table Script wizard appears.
3. Select the **upload\_insert** event from the dropdown list.
4. Click Finish.
5. In the right pane, double-click the **default** table script.

A dialog appears.

6. Type the following code into the dialog:

```
INSERT INTO cust ( cust_id, emp_id, cust_name )  
VALUES ( ?, ?, ? )
```

7. Save the script and close the dialog.
8. Repeat this step for the download\_cursor event, using the following script:

```
SELECT cust_id, emp_id, cust_name  
FROM cust
```

You have now generated scripts that perform a snapshot synchronization of your database, uploading new rows to the consolidated database and downloading rows from the consolidated database to the remote. In a complete MobiLink installation, you would add scripts for other events.

## Lesson 3: Start the MobiLink synchronization server

☞ In this lesson you start the MobiLink synchronization server so that you can synchronize the data in the consolidated and remote database.

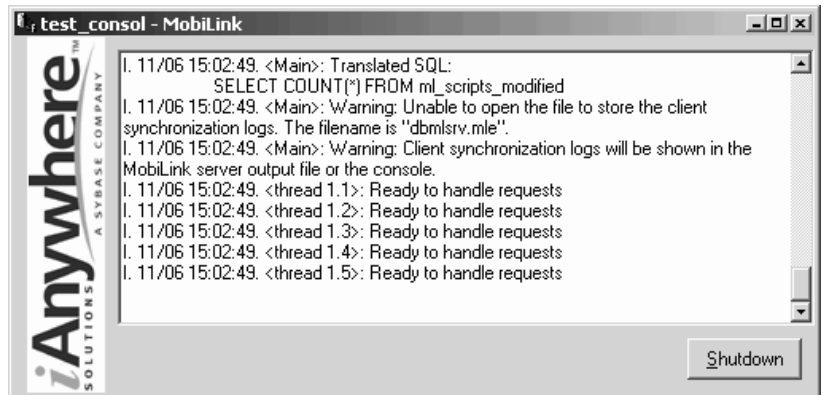
### ❖ To start the MobiLink synchronization server

1. From the command prompt type:

```
dbmlsrv9 -c "dsn=test_consol" -o mlserver.mls -v+ -dl -zu+
```

☞ For a detailed explanation of the meaning of the options, see “[MobiLink Synchronization Server Options](#)” [*MobiLink Administration Guide*, page 189].

Once you have executed the MobiLink synchronization server command, the output below appears.



You can check to see that this screen appears to ensure you are ready to proceed to the next lesson in the tutorial.

## Lesson 4: Run the MobiLink synchronization client utility

In this lesson you start the MobiLink synchronization client. You specify connection parameters at the command prompt using the `-c` option with the `dbmlsync` client utility. These parameters are for the remote database.

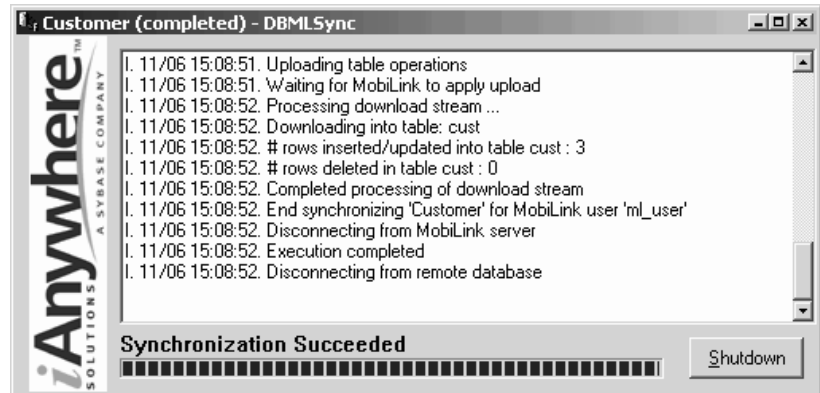
### ❖ To run the MobiLink synchronization client

1. Ensure the MobiLink synchronization server is started.
2. At the command prompt type:

```
dbmlsync -c "dsn=test_remote" -o dbmlsync.out -v
```

☞ For more information see [“MobiLink synchronization client”](#) [*MobiLink Clients*, page 96].

Once you have executed the MobiLink synchronization remote database command, the output below appears.



Check to see that this screen indicates that your synchronization has succeeded. You can then connect to the remote database and confirm that the `cust` table has three rows in it. These rows were added at the consolidated database.

You may wish to add more rows at the remote and consolidated database, and synchronize them.

#### **Clean up**

Be sure to delete all tutorial-related data sources and databases once you have finished this tutorial.

## Summary

Now that you have two databases and have synchronized their contents, you can do so any time by running the synchronization server and client.

During this tutorial, you accomplished the following tasks.

- ◆ Created new Adaptive Server Anywhere databases to serve as consolidated and remote databases
- ◆ Wrote synchronization publication and subscription definitions
- ◆ Created scripts to control data upload and download.
- ◆ Started a MobiLink synchronization server, the MobiLink synchronization client, and synchronized the remote database with the consolidated database

### Learning accomplishments

In this tutorial you:

- ◆ Gained familiarity with the MobiLink synchronization server and client as an integrated system and learned MobiLink synchronization server and client commands and options
- ◆ Acquired competence in executing MobiLink synchronization server and client commands
- ◆ Became competent at writing synchronization scripts.

### What's next?

You may need to learn more about MobiLink functioning to get the most out of MobiLink. The following areas are good starting points for further reading:

- ◆ Try the next tutorial in our series “[Introduction](#)” [*MobiLink Tutorials*, page 2].
- ◆ Read the introductory chapter on MobiLink “[Introducing MobiLink Synchronization](#)” [*MobiLink Administration Guide*, page 3].
- ◆ Read the chapter on running the MobiLink synchronization server “[Running MobiLink Outside the Current Session](#)” [*MobiLink Administration Guide*, page 157].
- ◆ Get to know the MobiLink client utility “[MobiLink synchronization client](#)” [*MobiLink Clients*, page 96].





---

## CHAPTER 18

# Replicating Data with SQL Remote

About this chapter

This chapter shows you how to set up a simple replication system using SQL Remote.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About SQL Remote</a>	294
<a href="#">Lesson 1: Getting Started</a>	295
<a href="#">Lesson 2: Set up the consolidated database</a>	296
<a href="#">Lesson 3: Set up the remote database</a>	300
<a href="#">Lesson 4: Replicate data</a>	302
<a href="#">Lesson 5: Restore the database and database settings</a>	305
<a href="#">Summary</a>	307

---

## About SQL Remote

Data replication is the sharing of data among physically distinct databases. Changes made to shared data in any one database are replicated in the other databases. Sybase provides three distinct strategies for data replication:

- ◆ SQL Remote
- ◆ MobiLink
- ◆ Replication Server

SQL Remote is designed for two-way replication involving a consolidated data server and a very large number of remote databases, typically including many mobile databases. Administration and resource requirements at the remote sites are minimal. This system is message-based.

In a SQL Remote installation, the central database must be either Sybase Adaptive Server Anywhere or Sybase Adaptive Server Enterprise.

SQL Remote replicates data by scanning the transaction log and preparing messages, as appropriate, for each transaction. It orders these messages and sends them to the remote or consolidated site. When processing received messages, SQL Remote always processes them in the same order as they were applied to the other database. When necessary, it automatically delays processing a message until all earlier messages have been applied.

☞ For more information about synchronization strategies, including complete SQL Remote documentation, see the “[About This Manual](#)” [*SQL Remote User's Guide*, page ix].

☞ For an introduction to MobiLink synchronization, see “[Synchronizing Databases with MobiLink](#)” on page 279.

## Lesson 1: Getting Started

This tutorial describes how to set up a simple SQL Remote replication system.

With SQL Remote, you can carry out two-way replication between a central database (called the **consolidated database**) and a set of **remote databases**. These remote databases may be on laptop computers, while the consolidated database may be on a network in an office. All the setup and administration is carried out at the consolidated database.

This tutorial describes a very simple case, replicating one table from the sample database to a single remote database on the same machine.

In the tutorial, you act as the system administrator of a consolidated Adaptive Server Anywhere database. The replication system consists of a simple table from the sample database. The table we replicate is the department table, which is one of the simplest in the database.

The tutorial takes you through the following steps:

1. Setting up the consolidated database.
2. Creating a file-sharing replication system with a single remote database.
3. Replicating data between the two databases.

### Create directories for the tutorial

Before you start, you need to create directories to hold the databases and other files you create during the tutorial. You should also create a copy of the sample database in case you need to recreate it in its original form.

#### ❖ Prepare for the tutorial

1. Create a directory to hold the files you make during this tutorial, for example, *c:\tutorial*.
2. Create two subdirectories of *tutorial* called *DBA* and *field*, for example, *c:\tutorial\DBA* and *c:\tutorial\field*. These subdirectories hold messages for each of the two user IDs in the replication system.
3. Create a copy of the sample database. Create a new folder for the copy, for example, *demoback*, and copy *asademo.db* into it.

What next?

Now you're ready to create the consolidated database.

---

## Lesson 2: Set up the consolidated database

This section shows you how to prepare the consolidated database of a simple SQL Remote replication system. You require DBA authority to do this.

To prepare a consolidated database for replication, this lesson takes you through the following steps:

1. Create a message type to use for replication.
2. Grant PUBLISH permissions to a user ID to identify the source of outgoing messages.
3. Grant REMOTE permissions to all user IDs that are to receive messages.
4. Create a publication describing the data to be replicated.
5. Create subscriptions describing who is to receive the publication.

### Add a SQL Remote message type

All messages sent as part of replication use a message type. A message type description has two parts:

- ◆ A message link supported by SQL Remote. In this tutorial, we use the FILE link. This is a file sharing method, where the messages are left in a file on disk, and read by the intended recipient. SQL Remote also supports ftp and e-mail protocols.
- ◆ An address for this message link, to identify the source of outgoing messages. In this tutorial we use a file path to specify where the messages will be left.

Message types are created in all new databases, but you do need to supply an address for the message type you will use.

#### ❖ Add an address to a message type

1. From Sybase Central, connect to the sample database as user ID **DBA** using the password **SQL**, and open the sample database container (**asademo**) in the left pane.
2. Click the SQL Remote Users folder in the left pane.
3. Click the Message Types tab in the right pane.
4. Click the FILE message type, and from the File menu choose Properties.

5. Enter a publisher address to provide a return address for remote users. The publisher address is the directory you created in Lesson 1 to hold messages for the consolidated database (*DBA*). For example, `c:\tutorial\DBA`.
6. Click OK to save the message type.

## Add the publisher to the database

Each database in a SQL Remote replication system needs a single user ID that identifies the **publisher** of the data. Here, we make the *DBA* user ID the publisher.

### ❖ Set the publisher

1. Click the Users & Groups folder in the left pane.
2. Click *DBA* in the right pane, and from the File menu, choose Change to Publisher from the popup menu.  
Publisher appears in the Type column beside *DBA*.

A database can have only one publisher. You can find out who the publisher is at any time by opening the Users & Groups folder.

## Add a remote user to the database

Each remote database is identified in the consolidated database by a user ID with REMOTE permissions.

When a remote user is added to a database, the message system they use and their address under that message system need to be stored along with their database user ID.

### ❖ Add a remote user

1. Click the SQL Remote Users folder in the left pane.
2. From the File menu, choose New ► SQL Remote User.  
The Remote User Creation wizard appears.
3. Type the name **field** as the name of the new remote user, and click Next.
4. Ensure that the user is allowed to connect, and type the password **field**. Confirm the password by entering it again. Click Next.
5. Select *DBA* permissions as well as Remote *DBA* permissions for the remote user. Click Next.

- 
6. Select the File message type and enter the remote address **field** in the text box. Click Next.
  7. Select Send Then Close. Click Next.
  8. Click Finish to create the remote user.

The remote user field appears in the SQL Remote Users folder.

#### Notes

If you forget to set DBA permission in the wizard, you can set it by clicking on the user, then selecting Properties from the File menu, and checking DBA on the Authorities tab.

You have now created the users who will use this system.

## Add publications and subscriptions

This section describes how to add a publication to a database, and how to add a subscription to that publication for a user. The publication replicates all rows of the table department.

### ❖ Add a publication

1. Click the Publications folder in the left pane.
2. From the File menu, choose New ► Publication.  
The Publication Creation wizard appears.
3. Type the name **DepartmentPub** as the name of the new publication, and click Next.
4. Select department from the list of Available Tables. Click Add.  
The table appears in the list of Selected Tables on the right.
5. Click Finish to create the publication.

#### Add a subscription

Each user ID that is to receive changes to a publication must have a **subscription** to that publication. Subscriptions can only be created for a valid remote user. You need to add a subscription to the DepartmentPub publication for the remote database user field.

### ❖ Add a subscription

1. Expand the Publications folder in the left pane.
2. Select the DepartmentPub publication in the left pane.
3. Click the SQL Remote Subscriptions tab in the right pane.

4. From the File menu, choose New ► SQL Remote Subscription.  
The Create a New SQL Remote Subscription dialog appears.
5. Click the user field in the dialog.
6. Click Finish to create the subscription.

You have now set up the consolidated database.

What next?

You can now create the remote database.

---

## Lesson 3: Set up the remote database

The remote database needs to be created and configured in order to send and receive messages and participate in a SQL Remote setup.

The database extraction utility enables you to carry out all the steps needed to create a remote database complete with subscriptions and required user IDs.

### Create the remote database

You need to extract a database from the consolidated database for remote user field.

#### ❖ Extract the remote database

1. Click the sample database asademo (DBA) in the left pane, and from the File menu, choose Extract Database from the popup menu.

The Extract Database wizard appears.

2. Click Next on the introductory page of the wizard.
3. Click asademo in the dialog. Click Next.
4. Set the isolation level to 3. Click Next.
5. Click the user field, and select the Start Subscriptions Automatically option. Click Next.
6. Choose Extract and Reload into a new database. Click Next.
7. Save the database to the file `c:\tutorial\field\field.db`. Click Next.
8. Choose Extract the structure and data.
9. Select the Order Data by Primary Key option. Click Next.
10. Select the Foreign Keys, Procedures & Functions, Triggers, and Views checkboxes.
11. Clear the Extract fully qualified publication definitions checkbox. Click Next.
12. Clear the Connect to the new database checkbox.
13. Click Finish to extract the remote database.

The Extracting Database window appears, displaying the progress of the extraction. When completed, close the window.



Note

In a proper SQL Remote setup, the remote database **field** would be loaded on to the computer, together with a database server and any client applications required. For this tutorial, we leave the database where it is and use Interactive SQL to input and replicate data.

## Verify that the database is created properly

To see what the extraction utility has done, connect to the *field* database and confirm that all the database objects are created.

### ❖ Browse through the remote database

1. In Sybase Central, click the Connect button.  
The New Connection dialog appears.
2. Choose Adaptive Server Anywhere 9 and click OK.
3. On the Identification tab, enter the User ID **field** and Password **field**.
4. On the Database tab, enter the database path `c:\tutorial\field\field.db`.
5. Click OK to connect to the database.  
The database field appears in the left pane of Sybase Central.
6. Expand the Tables folder in the left pane. The department table, owned by user DBA, is in the list.
7. In the left pane, click the Department table.
8. In the right pane, click the Data tab to show the five rows of the department table.
9. Open the SQL Remote Users folder.  
On the SQL Remote tab, you will see that user DBA is designated as a consolidated user. This means DBA is the publisher of the consolidated database, and therefore above the field database in the hierarchy.
10. Click the Message Types tab in the right pane.  
**Field** is designated as the publisher's address. Any data sent from this database comes from the user **field**, just as any data from the consolidated database comes from the user DBA.
11. Open the Publications folder. You will see that the DepartmentPub publication is present.

What next?

The system is now ready for replication.

---

## Lesson 4: Replicate data

You now have a replication system in place. In this section, data is replicated from the consolidated database to the remote database, and from the remote database to the consolidated database.

### Add data to the consolidated database

First, enter a row in the consolidated database.

#### ❖ Enter data in the consolidated database

1. In the left pane, expand the Tables folder of the consolidated database (asademo).
2. Select the department table in the left pane.
3. Click the Data tab in the right pane.
4. Click the plus sign in the toolbar to add a row.
5. Enter the following values:

Column	Value
dept_id	202
dept_name	Eastern Sales
dept_head_id	Leave as (NULL)

6. Press Enter.

If a confirmation dialog appears, click OK to update the row.

The next step is to send the new row to the remote database.

To send data to the remote database, you run the Message Agent at the consolidated database. The *dbremote* program is the Message Agent for Adaptive Server Anywhere.

#### ❖ Send the data to the remote database

1. At a command prompt, navigate to your tutorial directory. For example, navigate to *c:\tutorial*.
2. To run the Message Agent against the consolidated database, execute the following command:

```
dbremote -c "dbn=asademo;uid=DBA;pwd=SQL"
```

The SQL Remote window appears, and displays messages about the status of replication.

3. When SQL Remote displays the message `Execution completed`, click `Shutdown`.

To receive the insert statement at the remote database, you must run the Message Agent, `dbremote`, at the remote database.

#### ❖ Receive data at the remote database

1. At a command prompt, change to your tutorial directory. For example, navigate to `c:\tutorial`.
2. Execute the following command to run the Message Agent against the **field** database:

```
dbremote -c "dbn=field;uid=field;pwd=field"
```

The SQL Remote window appears, informing you of the status of replication. The window indicates that a message was received from DBA.

3. When SQL Remote displays the message `Execution completed`, click `Shutdown`.

#### Notes

- ◆ The SQL Remote window displays status information while running. This information can be output to a log file for record keeping. You will see that the Message Agent first receives a message from `asademo`, and then sends a message. This return message contains confirmation of successful receipt of the replication update; such confirmations are part of the SQL Remote message tracking system that ensures message delivery even in the event of message system errors.
- ◆ Depending on the current status of the database, there are three different connectivity parameters you can use to connect to a database:
  - **dbf** Connect to the database using the database file. This parameter requires you to specify the database file itself. If no server is currently running, a server will be started and the database will be loaded onto it. If a server is already running, the database will be loaded onto the default server.
  - **dbn** Connect to the database using the database name. This parameter requires you to specify the name of the database. You can only use this parameter when the database is already running.
  - **dsn** Connect to the database using the database source. A data source is a collection of parameters stored in the system registry or in a set of files. The source is referenced simply by its name.

---

☞ For more information about data sources, see “[DataSourceName connection parameter \[DSN\]](#)” [*ASA Database Administration Guide*, page 190].

## Verify that the data has arrived

From Sybase Central, inspect the department table to verify that the row has been received.

### ❖ Verify that the data has arrived

1. Open the *field* database container.
2. In the left pane, select the department table. In the right pane, click the Data tab to view the rows in the table.

You will see that the department table contains the Eastern Sales department entered at the consolidated database.

## Replicate from the remote database to the consolidated database

Now try entering data at the remote database and sending it to the consolidated database. Only the outlines are presented here.

### ❖ Replicate data from the remote database to the consolidated database

1. Insert a row at the remote database, for a department with a dept\_id of 203 and a dept\_name of Western Sales.
2. Run *dbremote* to send the message to the consolidated database:

```
dbremote -c "dbn=field;uid=field;pwd=field"
```

3. Run *dbremote* to receive the message at the consolidated database:

```
dbremote -c "dbn=asademo;uid=DBA;pwd=SQL"
```

4. Display the data in the department table at the consolidated database: It contains the Western Sales row.

What next?

The tutorial is now complete, but you should continue to the next section to restore the sample database to its original form.

## Lesson 5: Restore the database and database settings

Once you have completed the tutorial it is important to undo any changes you have made to the sample database. Make sure that the following steps are completed in order to ensure that the settings are reset properly.

### ❖ Delete the remote user

1. In the left pane, click the Users & Groups folder for the sample database.
2. Click the user field, and from the Edit menu, choose Delete. Click Yes to remove the user from the list.

### ❖ Delete the publication

1. Click the Publications folder in the left pane.
2. Click DepartmentPub in the left pane, and from the Edit menu, choose Delete. Click Yes to confirm the deletion.

### ❖ Revoke the publishing status from the sample database

1. Click the Users and Groups folder in the left pane.
2. Click the DBA user in the left pane, and from the File menu, select Revoke Publisher.

#### **Revoking a status while running the database**

The status of a user cannot be modified while the user is running the database. Make sure that the user is idle before revoking or invoking a status.

### ❖ Restore the original message type settings

1. Select the SQL Remote Users folder and click the Message Types tab in the right pane.
2. Right-click the File message type in the right pane and choose Properties.
3. Delete the publisher address and click OK to restore the settings.

---

### ❖ Shut down the remote database

1. In the left pane, select the field database container.
2. From the Tools menu, select Disconnect.  
The Disconnect dialog appears.
3. Select the connection that corresponds to the field database.
4. Click Disconnect.

### ❖ Delete the remote database

1. In the left pane, select the Adaptive Server Anywhere plug-in.
2. In the right pane, click the Utilities tab.
3. In the right pane, double-click Erase Database.  
The Erase Database wizard appears.
4. Follow the instructions in the wizard to erase the field database.
5. Enter the name of the database you want to erase or click Browse to search for the database. For example, enter `c:\tutorial\field\field.db`. Click Finish to erase the database file.

#### Notes

Now that the tutorial is complete, it might be a good idea to delete the tutorial directory (`c:\tutorial`) in order to save space.

## Restore the data in the database

The most important part of the cleanup process is to ensure that the changes to the sample database are reversed. The integrity of the sample database is very important in order to carry out other tutorials in later chapters of the manual.

### ❖ Delete the inserted data from the sample database

1. From Sybase Central, display the data in the department table of the asademo database.
2. Delete the rows with dept\_id values of 202 and 203 to restore the table to its original state.

## Summary

In this tutorial you learned how to

- ◆ Prepare the consolidated database of a simple replication system.
- ◆ Create and configure a remote database.
- ◆ Replicate data in both directions between the two databases.
- ◆ Restore your database and database settings.
- ◆ Verify all the steps.

## Where do I go from here?

☞ For more information about SQL Remote, see the [“About This Manual” \[SQL Remote User’s Guide, page ix\]](#).

☞ For an introduction to MobiLink synchronization, see [“Synchronizing Databases with MobiLink” on page 279](#).





---

## CHAPTER 19

# Designing Databases with PowerDesigner

### About this chapter

SQL Anywhere Studio includes PowerDesigner Physical Architect, a module of Sybase's powerful database design tool, PowerDesigner. This module provides ways to generate and modify databases using a graphical representation of the database structure. You can optimize your database by customizing tables, columns, indexes, referential integrity, views, physical storage, triggers, and stored procedures.

### Contents

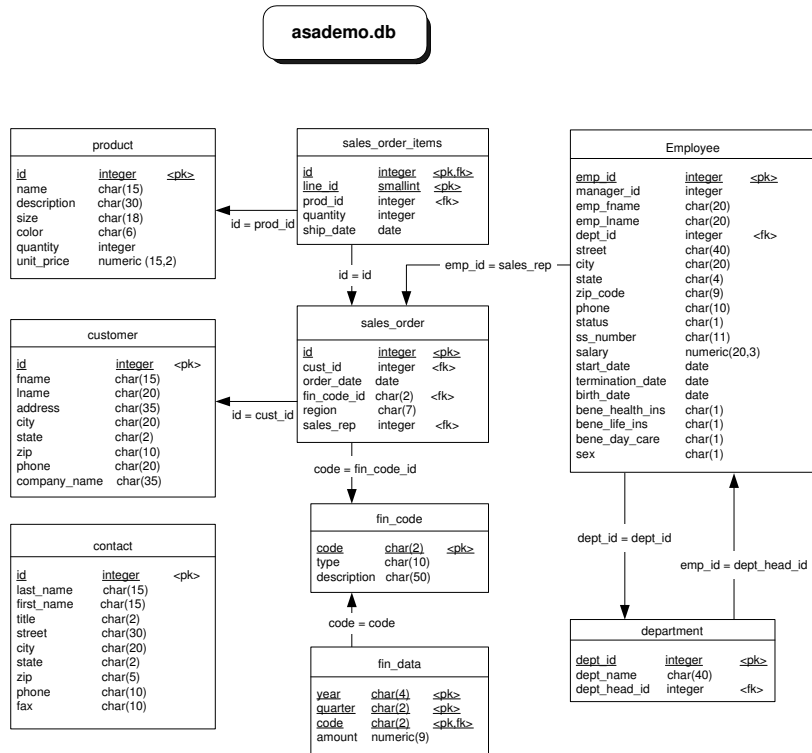
<b>Topic:</b>	<b>page</b>
<a href="#">About PowerDesigner</a>	310
<a href="#">Lesson 1: Getting Started</a>	312
<a href="#">Lesson 2: Add a column</a>	317
<a href="#">Lesson 3: Check your work</a>	320
<a href="#">Lesson 4: Save changes and generate database</a>	321
<a href="#">Summary</a>	324

# About PowerDesigner

The structure of your database, such as the tables, relationships, views, and triggers, is called the database **schema**. You use SQL statements to create and arrange these elements to your liking, but doing so without a graphical tool can be confusing.

PowerDesigner gives you a graphical representation of the structure of your database. Better still, you can modify the structure of the database or create an entirely new one simply by drawing new tables or entering information. Once your design is complete, PowerDesigner can generate a SQL script to generate your new database.

The following diagram, which displays the structure of the sample database, is easily created using PowerDesigner.



The performance of your database depends heavily on your design. In general, you should store information about different distinct types of objects, such as employees or products, in separate tables.

You can identify relationships between these tables using references,

meaning that foreign keys in one table identify particular rows in another table. Many-to-one and one-to-many relationships can be represented by a reference. Many-to-many relationships require two references and another table.

☞ For more information about database design, see “[Designing Your Database](#)” [*ASA SQL User’s Guide*, page 3].

☞ For more information about PowerDesigner, click Help on the toolbar of the PowerDesigner main window to access three books: *PowerDesigner General Features Guide*, *PowerDesigner PDM User’s Guide*, and *PowerDesigner Report User’s Guide*.

☞ For more PowerDesigner tutorials, open PowerDesigner. From the Help menu, choose Where to Start ► PDM Getting Started. This accesses the book *Physical Data Model Getting Started*.

---

## Lesson 1: Getting Started

PowerDesigner can read the structure of a database from a script file that creates the database. However, it is generally easier to just connect to your database from PowerDesigner and let it extract the design directly with the reverse engineering feature.

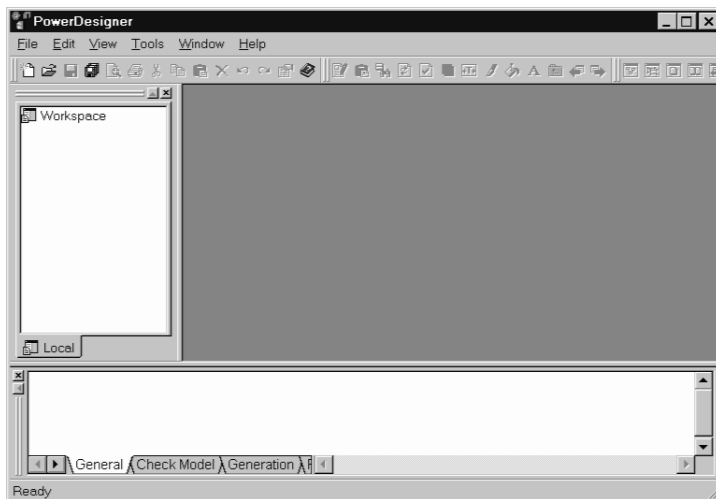
The following tutorial uses the sample database as a starting point. The tutorial illustrates PowerDesigner by implementing a modification that improves the design of the sample database.

Currently, the price of each product is always read from the product table. As a result, updating the price effectively changes the sale price of that item on all previous orders. Adding a `unit_price` column to the `sales_order_items` table will correct this problem. The actual selling price to each customer can then be stored separately. The price in the product table records the current list price.

### ❖ To start PowerDesigner

1. From the Start menu, choose Programs ► SQL Anywhere 9 ► PowerDesigner 9 ► PowerDesigner.

The PowerDesigner main window appears:



The PowerDesigner main window includes an object browser docked to the left, and an output window docked at the bottom.

2. From the File menu, choose New.

The New dialog appears.

3. In the New dialog, choose Physical Data Model and then click OK.  
The New Physical Data Model dialog appears.
4. On the General tab, from the dropdown list choose Sybase AS Anywhere 9.
5. Use the default settings for the remaining options, and then click OK.  
The model name PhysicalDataModel\_1 appears in the browser and in the title bar.

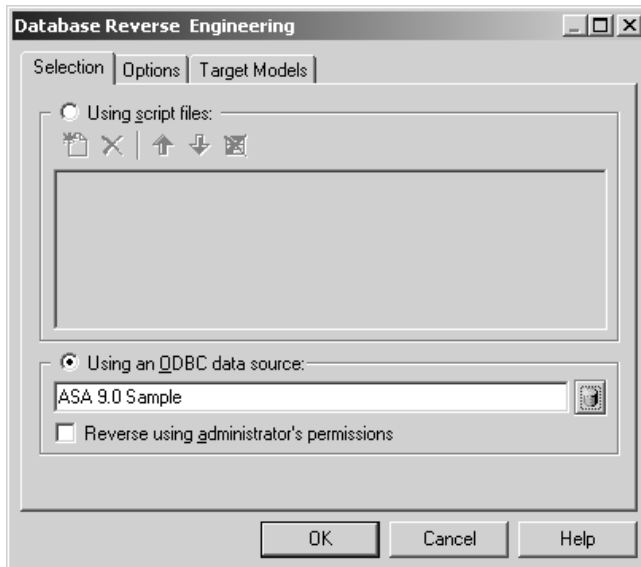
## Reverse engineer the database

In this section, you generate a Physical Data Model (PDM) of the sample database by reverse engineering it.

### ❖ To reverse engineer the database

1. Click the diagram window (the large central pane of PowerDesigner).
2. From the Database menu, choose Reverse Engineering Database.

The Database Reverse Engineering dialog appears:



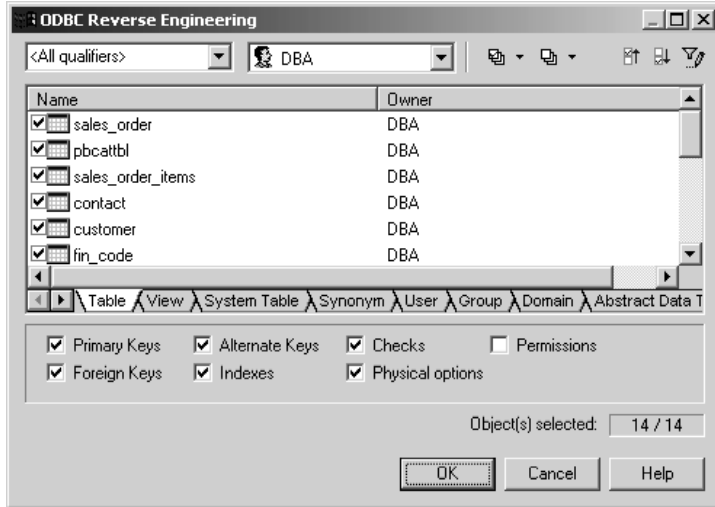
3. Ensure that Using an ODBC Data Source is selected, and that the data source is ASA 9.0 Sample.  
If ASA 9.0 Sample does not appear, click the icon to the right of the data source field. The Connect to an ODBC Data Source dialog appears.

---

Select Machine Data Source and choose ASA 9.0 Sample from the dropdown list. Enter user ID **DBA** and password **SQL**. Click Connect to return to the Database Reverse Engineering dialog.

4. In the Database Reverse Engineering dialog, click OK.

The ODBC Reverse Engineering dialog appears:



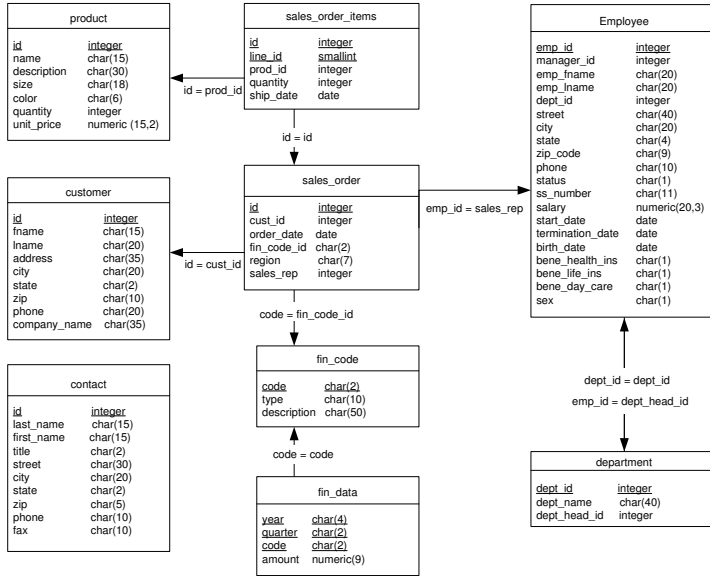
In the lower half of the dialog, there are seven checkboxes for selecting keys, indexes, and so on. These are the **reverse engineering options**. Ensure that all options are selected, except for the Permissions checkbox (the default). You should also ensure that all tables are selected (also the default).

5. Click OK to reverse engineer the database.

A graphic representation of the sample database appears in the diagram window, and the model objects appear in the browser:



asademo.db





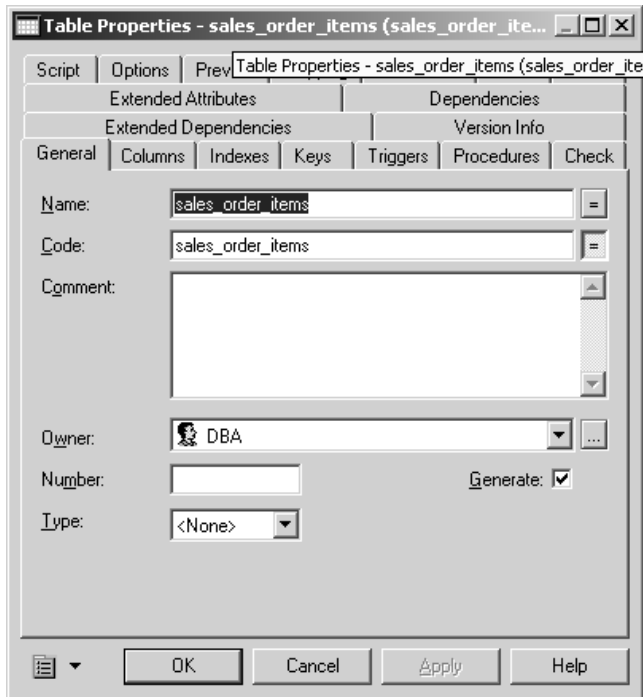
## Lesson 2: Add a column

You are now ready to add the `unit_price` column to the `sales_order_items` table. You can accomplish this task by accessing the list of columns through the Table property sheet.

### ❖ To add a column

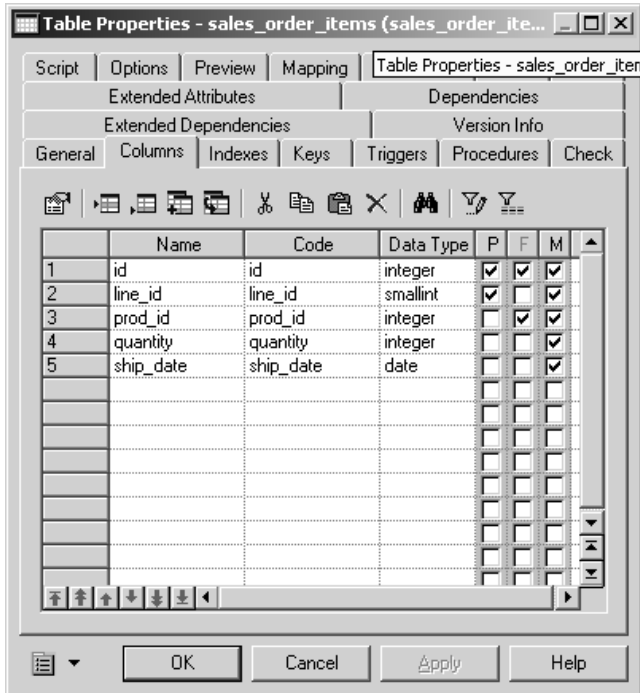
1. Select the `sales_order_items` table.
2. From the View menu, choose Properties.

The Table Properties dialog appears.



3. Click the Columns tab.

The list of columns appears.



4. Add a new column to hold the unit price.

Click the Insert a Row button.

An arrow appears at the beginning of the line and a column with a default name (Column\_6) appears.

5. Type **unit\_price** in the name column. The name is automatically duplicated as the code.

In the Data Type column, choose Numeric from the dropdown list.

The Data field may be too narrow to read. You can pull the sides of the column to expand it.

6. The column properties P, F, and M stand for Primary Key, Foreign Key, and Mandatory, as follows:

- ◆ Primary key designates a column whose values uniquely identify a row in the table.
- ◆ Foreign key designates a column that depends on and migrates from a primary key column in another table.
- ◆ Mandatory indicates a column that must be assigned a value.

7. Select Mandatory and then click OK.

8. Examine the effect of your changes on the diagram of the database.  
The sales\_order\_items table now includes a new column called unit\_price.

---

## Lesson 3: Check your work

PowerDesigner lets you quickly detect database design errors in your new model.

### ❖ To check your new schema

1. From the Tools menu, choose Check Model.

The Check Model Parameters dialog appears. You can use the default parameters.

2. Click OK.

The results of Check Model appear in the Result List dialog.

## Lesson 4: Save changes and generate database

In PowerDesigner, models that depict the physical components of your database design, including tables and columns, are called **Physical Data Models (PDM)**. PowerDesigner stores these in files with the extension *.PDM*.

### ❖ To save the physical data model (PDM)

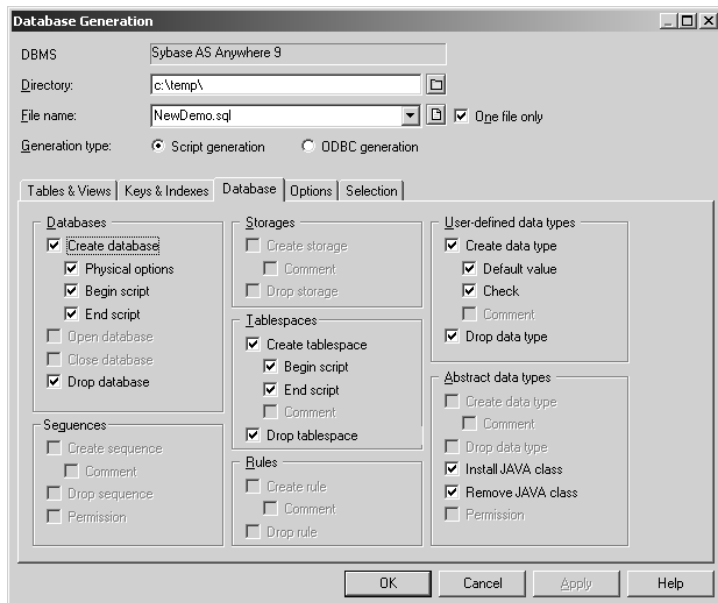
1. From the File menu, choose Save As.
2. Type the file name `c:\Temp\NewDemo.pdm`.
3. Click Save.

You can use PowerDesigner to generate a SQL script that implements all the components of your model. You can then use the SQL script to generate a database.

### ❖ To generate a SQL script to create your new database

1. From the Database menu, choose Generate Database.

The Database Generation dialog appears:



2. Type `c:\Temp\` in the Directory field and `NewDemo.sql` in the File Name field.

3. Ensure that Script Generation is selected.
4. Click the Database tab, and ensure that Create Database is selected.  
Explore the other tabs to observe options that give you control over many other properties of the generated script.
5. Click OK.

When the script is created, the Result dialog appears.

6. Click Edit to view the script.

Check that your changes are reflected in the script. For example, the definition of the new office table appears below.

```

/* ===== */
/* Table: office */
/* ===== */
create table office
(
    id            integer            not null
                default autoincrement
                check (
                    id >= 100),
    name          char(15)           not null,
    street        char(30)           not null,
    city          char(20)           not null,
    state         char(2)            not null,
    zip           char(5)            not null,
    phone         char(10)           ,
    fax           char(10)           ,
                primary key (id)
);

```

7. When you are finished, close the dialog by clicking Close in the Result dialog.

You can now create your new database from Interactive SQL.

### ❖ To create the new database

1. Start Interactive SQL.  
From the Start menu, choose Programs ► SQL Anywhere 9 ► Adaptive Server Anywhere ► Interactive SQL.
2. Connect to the sample database using the ASA 9.0 Sample ODBC data source.
3. Create an empty database:

- ◆ Execute the following SQL statement, substituting any convenient directory.

```
CREATE DATABASE 'c:\\Temp\\newdemo.db'
```

**Tip**

To execute a SQL statement in Interactive SQL, type or copy the statement in the SQL Statements pane, and then press F5. Alternatively, from the SQL menu, choose Execute.

4. Close the connection to the sample database.

From the SQL menu, choose Disconnect.

5. Connect to the new database.

From the SQL menu, choose Connect.

- ◆ Enter **DBA** as the User ID
- ◆ Enter **SQL** as the Password
- ◆ Click the Database tab, and in the Database File box, enter the full path and file name of the new database file.
- ◆ Click OK.

6. Use the read statement. Remember that this statement demands that you enclose the file name in double quotes.

- ◆ Execute the SQL statement:

```
READ "c:\\Temp\\newdemo.SQL"
```

You can use these basic steps to modify other databases.

---

## Summary

This tutorial introduced only some of the basic features of PowerDesigner. In fact, it is capable of handling the complete design or modification of your database schema, including all tables, views, indexes, references, triggers, and procedures.

### Domains

Other features greatly simplify the task of designing larger databases. For example, you can specify **domains**. A domain holds a particular type of data, such as a phone number. It has a data type associated with it, but is more specific. For example, you can create a domain of identification numbers. Whenever you need an identification number in a table, you can associate that column with the identification number domain. All properties and checks associated with that domain are attached automatically.

Domains reduce repetitive definitions. In doing so, they not only reduce your work, but also reduce the chance that you will erroneously use a different type definition or check procedure. Rather than identify a column simply as an integer, you specify what specific type of data that column contains. All instances of that data type share a common definition.

☞ For more information, see “Using domains” [*ASA SQL User’s Guide*, page 92].

### Business rules

A business rule is a written expression of the way a business operates. For example, *the order shipped date must be greater than or equal to the order date* is a business rule.

Business rules fall into four categories:

- ◆ **Definition** Expresses inherent properties of an object. Definitions typically describe entities.
- ◆ **Fact** Expresses certainty or existence. Facts typically describe relationships.
- ◆ **Validation** A constraint on a value.
- ◆ **Formula** Calculation used to produce values.

Business rules are particularly handy because they relate directly to the task that a customer requires a database to perform. By recording business rules and attaching them to particular objects, you can ensure that a database performs the required tasks.

## Where do I go from here?

☞ For more information about PowerDesigner, select Help on the toolbar



of the PowerDesigner main window to access three books: *PowerDesigner General Features Guide*, *PowerDesigner PDM User's Guide*, and *PowerDesigner Report User's Guide*.

☞ For more PowerDesigner tutorials, open PowerDesigner. From the Help menu, choose Where to Start ► PDM Getting Started. This accesses the book *Physical Data Model Getting Started*.

☞ For more information about database design, see “[Designing Your Database](#)” [*ASA SQL User's Guide*, page 3].



---

## CHAPTER 20

# Creating Reports with InfoMaker

About this chapter

This chapter includes a brief tutorial of InfoMaker report making that gets you started and teaches you the basic skills required to be productive in the InfoMaker environment.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About InfoMaker</a>	328
<a href="#">Lesson 1: Getting Started</a>	329
<a href="#">Lesson 2: Create a basic report</a>	330
<a href="#">Lesson 3: Enhance your report</a>	333
<a href="#">Summary</a>	337

---

## About InfoMaker

InfoMaker is a powerful reporting and data maintenance tool. With InfoMaker, you can create the following objects:

- ◆ Reports to view data.
- ◆ Forms to view and change data.
- ◆ Queries to automatically retrieve data for reports or forms.
- ◆ Pipelines to pipe data from one database (or DBMS) to another.
- ◆ Applications to bundle reports and forms and distribute them to users.

InfoMaker provides built-in connectivity to a broad range of desktop and server-based databases. When you work in InfoMaker, you work in a graphical environment—and working with data in this environment means that you don't need to understand SQL. InfoMaker creates all SQL statements behind the scenes as you build your reports and other objects graphically.

☞ For more information about InfoMaker, from the Start menu, choose Programs ► Sybase ► InfoMaker 9.0 ► Online Help Files. Complete documentation is also available within the application by accessing the Help menu.

## Lesson 1: Getting Started

To use this InfoMaker tutorial, you must be connected to the Adaptive Server Anywhere sample database. To connect to the SQL Anywhere Studio 9.0 sample database you need to create a **database profile**.

### ❖ To start InfoMaker

1. From the Start menu, choose Programs ► Sybase ► InfoMaker 9.0 ► InfoMaker.

The InfoMaker main window appears. It includes the PowerBar, which has buttons for accessing the InfoMaker painters and online Help.

### ❖ To create a database profile for the SQL Anywhere Studio sample database with InfoMaker

1. In InfoMaker, open the Database Profiles dialog.

From the Tools menu, choose Database Profile.

2. Select ODB ODBC and then click New.

The Database Profile Setup dialog appears.

3. On the Connection tab, enter the following values:

- ◆ **Profile Name** Anywhere
- ◆ **Data Source** ASA 9.0 Sample
- ◆ **User ID** DBA
- ◆ **Password** SQL

4. Click OK.

The profile Anywhere should now be listed under ODB ODBC.

### ❖ To connect to the Adaptive Server Anywhere sample database once a data source has been created

1. If it is not already open, open the Database Profiles window by choosing Database Profile from the Tools menu.

2. Select Anywhere from the ODBC group and then click Connect.

What next?

Next, you will create a report using an InfoMaker wizard.

---

## Lesson 2: Create a basic report

This section shows you how to create a basic report, how to preview and save it, and how to change the settings in your design environment.

### Create a report

There are many types of reports that you can create. This section shows you how to create a report from a single table.

#### ❖ To create the report

1. From the File menu, choose New.
2. In the New dialog, click the Object tab.
3. Double-click the icon labeled Tabular to select the tabular presentation style.  
The Tabular Report Generator wizard appears.
4. Select the Quick Select data source, and ensure that Retrieve on Preview is selected. Click Next.  
The Quick Select dialog appears. This allows you to select a database table and some or all of the table's columns.
5. Select the contact table.
6. Select the following columns: last\_name, first\_name, title, phone, and fax. (You may need to scroll down.)  
InfoMaker moves the columns you select to the grid at the bottom of the dialog. You can use this grid for reordering the columns, and for providing sort and selection criteria.
7. Click OK.  
The Select Color and Border Settings dialog appears.
8. Click Next to use the default settings.  
A dialog summarizing your specifications appears.
9. Click Finish.

### Preview your report

In this section you will view your report to see what it looks like before printing it.

The main InfoMaker window has toolbars in the top section. Underneath the toolbars is the preview pane, which occupies the rest of the window. At the top of the preview pane is a title bar. At the left of the title bar is the name of the report you are previewing, and at the right are Minimize/Maximize buttons for adjusting the size of the preview pane.

❖ **To preview your report**

1. Position your pointer in the upper right corner of the preview pane and click the Maximize icon.

The Preview pane fills the entire window. Notice that it now contains the header information for the report, as well as information about the database.

2. If you want to turn the rulers on or off, click inside the section that contains the data. Then, from the File menu, choose Print Preview Rulers.
3. Click the Minimize icon in the upper right corner of the preview pane to return the preview pane to its original size and location.

## Save the report

❖ **To save the report**

1. From the File menu, choose Save.

The Save Report dialog appears.

2. In the Reports field, type **contacts\_by\_jobrole**. This becomes the name of the report.
3. Optionally, you can add further detail about this report by adding notes in the Comments box.

Click in the Comments box and type **This report shows my contacts grouped by job role..**

4. Click OK to save the report.

## Set up the design environment

In this section you make modifications to the controls, grid, and ruler.

---

❖ **To set up the design environment**

1. From the Design menu, choose Options.  
The Report Options dialog appears.
2. Make sure that the following options are selected: Show Grid, Show Ruler, Show Edges, and Retrieve on Preview.
3. Make sure that Snap to Grid is not selected.
4. You can click Help to see descriptions of all options.
5. Click OK.

What next?

You are now ready to customize the look of your report.



## Lesson 3: Enhance your report

In this section you'll learn how to make a number of enhancements to your report, including the following:

- ◆ Sorting the data.
- ◆ Creating and formatting headers and titles.
- ◆ Adding computed fields such as dates, page numbers, and totals.

### Define sorting and grouping

In this section you group contacts by title. To do this, you sort the data by title and then specify grouping by title.

First, you define the sorting.

#### ❖ To sort contacts

1. From the Rows menu, choose Sort.  
The Specify Sort Columns dialog appears.
2. Drag title from the Source Data box to the Columns box.
3. Drag last\_name and then first\_name into the Columns box.
4. Leave the checkboxes selected to accept the default Ascending sort order.
5. Click OK.

Next, you define the grouping based on the title column, so that all contacts with the same title are grouped together:

#### ❖ To group contacts

1. From the Rows menu, choose Create Group.  
The Specify Group Columns dialog appears.
2. Drag title to the Columns box and then click OK.

The grouping is completed. Maximize the Preview pane, and you will see that the data is sorted. The grouping does not yet appear.

### Enhance the report

In this section you enhance the report by rearranging controls, adding a title and date, adding page numbers, and adding a total.

---

### ❖ To adjust the group header

1. Place your mouse pointer on the gray bar called Header Group Title.  
The pointer changes to a double-arrow.
2. Drag the band down by five grid dots.
3. Drag the title column from the bottom of the preview pane into the bottom of the band for the group header.
4. With the title column still selected, click B and I on the style bar.
5. Review the report in the Preview view.

### ❖ To arrange the headers

1. Drag the First Name and Last Name text boxes into the right side of the header band to fill the empty space.
2. Drag the first\_name and last\_name columns into the right side of the detail band to fill the empty space.
3. From the Edit menu, choose Select ► Select Text.  
This selects all the headers. It may be difficult to see that they are selected, because the band is so narrow.
4. On the stylebar, click the Left Justification button. If your headers are not already bold, click the Bold button.
5. Review the report in the Preview pane.

### ❖ To add a title to the report

1. Drag the gray bar marked Header down about 10 grid dots.
2. From the Edit menu, choose Select ► Select Text. This selects all the headers.
3. Drag all the headers down close to the gray band.
4. From the Insert menu, choose Control ► Text.
5. Move the point of the pointer above the First Name box and down one grid dot from the top of the page. Click once. This positions the text box for the title.
6. Type **My Contacts**
7. Select size 14 from the dropdown list in the stylebar.

❖ **To add a date to the report**

1. From the Insert menu, choose Control ► Today().
2. Move the point of the pointer to the upper left corner of the report and click. This adds the date to the report.

❖ **To add page numbers to the report**

1. Drag the gray bar marked Footer down about four grid dots. If necessary, use the scrollbar to make room at the bottom of the Design view. Dragging down the footer bar makes space for the page number in the footer band.
2. From the Insert menu, choose Control ► Page n of n.
3. Place the pointer about two grid dots below the center of the footer band. Click.
4. Select size 10 in the stylebar.

❖ **To add a total by counting the number of last names**

1. Drag the gray bar marked Summary down about six grid dots. This makes space for the total.
2. From the Insert menu, choose Control ► Computed Field.
3. Place the pointer about four grid dots below the center of the summary band. Click.  
The Modify Expression dialog appears.
4. In the Functions box, click Count( #x for all ).
5. In the Columns box, click last\_name.
6. Click Verify.  
You receive a message stating that your expression is OK. Click OK.
7. Click OK to complete the definition of the expression for the computed field.

---

❖ **To format the total**

1. From the Insert menu, choose Control ► Text.
2. Position the pointer to the left of the computed field and click.
3. Type **Total contacts:**
4. Drag the text box to line it up with the computed field.
5. Lasso the text and computed field by circling them with the pointer while pressing the left mouse button, and then releasing the mouse button.
6. On the Stylebar, select the font size 10, and click B and I.

❖ **To print, save, and close the report**

1. From the File menu, choose Print Report.
2. In the Windows Print dialog, click OK.
3. From the File menu, click Close.
4. If you are prompted to save changes, click Yes.

## Summary

In this tutorial you learned how to

- ◆ Connect to InfoMaker.
- ◆ Create a basic report.
- ◆ Preview your report.
- ◆ Format the report by sorting the data, and adding headers, titles, dates, page numbers and a total.

## Where do I go from here?

In addition to reports, InfoMaker provides functionality for creating graphs, queries, and data entry forms.

☞ For more information about InfoMaker, choose Start ► Programs ► Sybase SQL Anywhere 9 ► InfoMaker 9.0 ► Online Help Files. Complete documentation is also available within the application by clicking the Help menu.



---

CHAPTER 21

# Microsoft Visual Basic Quick Start

About this chapter

This chapter describes how to develop a simple database application using Adaptive Server Anywhere and Microsoft Visual Basic.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Tutorial: Developing a Visual Basic application</a>	340

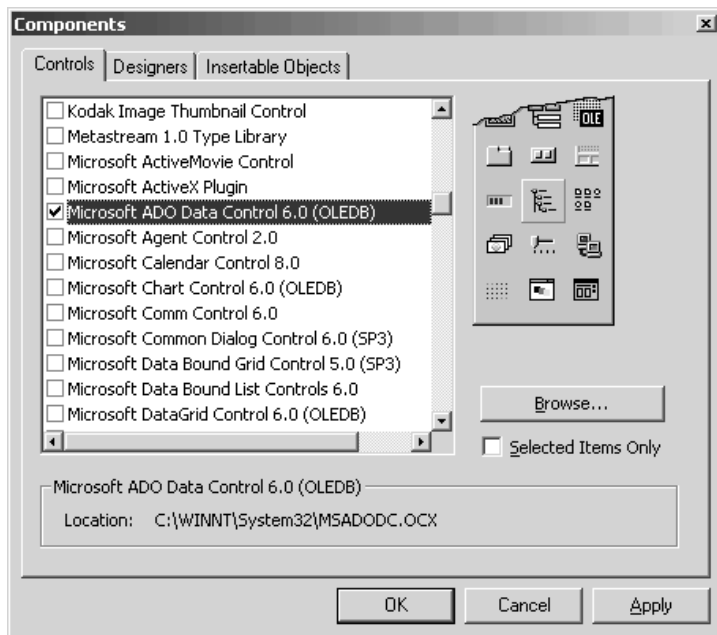
# Tutorial: Developing a Visual Basic application

This brief tutorial is based on Visual Basic 6.0. The complete application can be found in the Visual Basic project *Samples\ASA\VBStarter\ASASTarter.vbp*.

Visual Basic provides several data access technologies. In this tutorial, we use the Microsoft ADO Data Control with the Adaptive Server Anywhere OLE DB provider to access the Adaptive Server Anywhere sample database from Visual Basic.

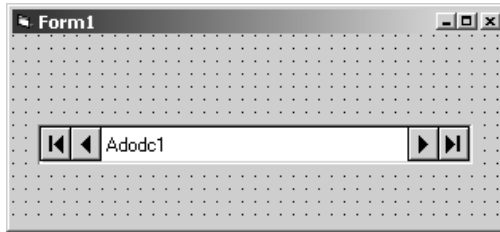
## ❖ To develop a database application with Visual Basic

1. Start Visual Basic, choosing a Standard Executable project.
2. Add the Microsoft ADO Data Control 6.0 to your tool palette:
  - ◆ From the Project menu, choose Components.
  - ◆ Select the Microsoft ADO Data Control 6.0 component from the list.



- ◆ Click OK to add the control to the palette.
3. Add the ADO Data Control to the form, as follows:



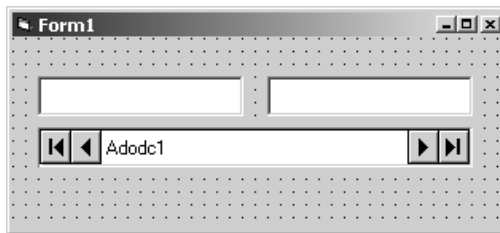


4. Configure the ADO Data Control:

Property	Value
ConnectionString	Provider=ASAPROV;DSN=ASA 9.0 Sample
CursorLocation	2 - asUseServer
CursorType	1 - adOpenKeyset
RecordSource	SELECT * FROM EMPLOYEE

The ConnectionString uses the Adaptive Server Anywhere OLE DB Provider (ASAProv) to connect to the ASA 9.0 Sample data source. The cursor settings take advantage of Adaptive Server Anywhere cursors rather than using the client-side cursors.

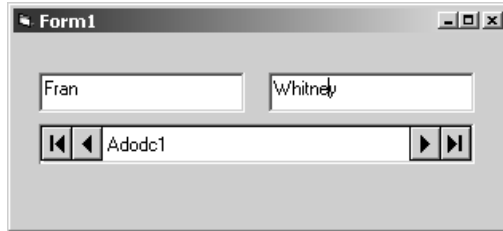
5. Add two text boxes to the form, as follows:



6. Bind the text boxes to the ADO Data Control:
- ◆ Set the DataSource property for each to be **Adodc1**.
  - ◆ Set the DataField property for the left-hand text box to **emp\_fname**, which is the column holding the employee's first name.
  - ◆ Set the DataField property for the right-hand text box to **emp\_lname**, which is the column holding the employee's last name.
7. Save the project.
8. Run the sample:

- 
- ◆ Choose Run ► Start to run the application.

The application connects to the Adaptive Server Anywhere sample database and puts the name of the first employee in the text boxes, as follows:



- ◆ You can use the buttons on the ADO Data Control to scroll through the rows of the result set.

You have now created a simple Visual Basic application that works with Adaptive Server Anywhere.

PART V

# APPENDIX





The appendix contains a glossary of terms pertinent to SQL Anywhere Studio.





---

## CHAPTER 22

# Glossary

Adaptive Server Anywhere (ASA)	The relational database server component of SQL Anywhere Studio, intended for use in mobile and embedded environments or as a server for small and medium-sized businesses.
article	In SQL Remote or MobiLink, an article is a database object that represents a whole table, or a subset of the columns and rows in a table. Articles are grouped together in a publication.  See also: <a href="#">“replication” on page 357</a> , <a href="#">“publication” on page 355</a> .
atomic transaction	A situation when commands in the transaction do not process properly and changes are undone or rolled back.
base table	A permanent table for data. Tables are sometimes called <b>base tables</b> to distinguish them from temporary tables and views.  See also: <a href="#">“temporary table” on page 359</a> , <a href="#">“view” on page 360</a> .
business rule	A guideline based on real-world requirements. Business rules are typically implemented through check constraints, user-defined data types, and the appropriate use of transactions.  See also: <a href="#">“constraint” on page 347</a> , <a href="#">“user-defined data type” on page 360</a> .
check constraint	A restriction that enforces specified conditions on a column or set of columns.  See also: <a href="#">“constraint” on page 347</a> , <a href="#">“foreign key constraint” on page 350</a> , <a href="#">“primary key constraint” on page 355</a> , <a href="#">“unique constraint” on page 360</a> .
checkpoint	The point at which all changes to the database are saved to the database file. At other times, committed changes are saved only to the transaction log.
client/server	A software architecture where one application (the client) obtains information from and sends information to another application (the server). The two applications often reside on different computers connected by a network.
collation	A combination of a character set and a sort order that defines the properties

---

	of text in the database. For Adaptive Server Anywhere databases, the default collation is determined by the operating system and language on which the server is running; for example, the default collation on English Windows systems is 1252LATIN1. A collation, also called a collating sequence, is used for comparing and sorting strings.
command file	A text file containing SQL statements. Command files can be built manually, or they can be built automatically by database utilities. The dbunload utility, for example, creates a command file consisting of the SQL statements necessary to recreate a given database.
communication stream	In MobiLink, the network protocol used for communication between the MobiLink client and the MobiLink synchronization server.
compressed database file	A database file that has been compressed to a smaller physical size using the dbshrink utility. Compressed databases are read-only. To make changes to a compressed database file, you must use an associated write file. You can expand compressed database files into normal database files using the dbexpand utility.
concurrency	The simultaneous execution of two or more independent, and possibly competing, processes. Adaptive Server Anywhere automatically uses locking to isolate transactions and ensure that each concurrent application sees a consistent set of data.   See also: <a href="#">“transaction” on page 359</a> , <a href="#">“lock” on page 352</a> , <a href="#">“isolation level” on page 351</a> .
conflict trigger	In SQL Remote replication, a trigger that fires when an update conflict is detected, before the update is applied. Conflict triggers are fired when the values in the VERIFY clause of an UPDATE statement fail to match the current values in the database.   See also: <a href="#">“replication” on page 357</a> , <a href="#">“trigger” on page 360</a> .
connection ID	A unique number that identifies a given connection between a client application and the database. You can determine the current connection ID using the following SQL statement:  <pre>SELECT connection_property( 'Number' )</pre>
connection profile	A set of parameters that are required to connect to a database, such as user name, password, and server name, that is stored and used as a convenience.
consolidated database	In database replication, a database that stores the master copy of the data. The consolidated database contains all of the data, while remote databases usually contain only subsets of the data. In case of conflict or discrepancy, the consolidated database is considered to have the primary copy of all data.

In MobiLink, the consolidated database can be Oracle, IBM DB2, Microsoft SQL Server, Adaptive Server Anywhere, or Adaptive Server Enterprise.

☞ See also: [“replication” on page 357](#).

constraint	<p>A restriction on the values contained in a particular database object, such as a table or column. For example, a column may have a uniqueness constraint, which requires that all values in the column be different. A table may have a foreign key constraint, which specifies how the information in the table relates to data in some other table.</p> <p>☞ See also: <a href="#">“check constraint” on page 345</a>, <a href="#">“foreign key constraint” on page 350</a>, <a href="#">“primary key constraint” on page 355</a>, <a href="#">“unique constraint” on page 360</a>.</p>
contention	<p>The act of competing for resources. For example, in database terms, two or more users trying to edit the same row of a database contend for the rights to edit that row.</p>
convoy	<p>A situation where all database connections but one are waiting for a shared resource.</p>
correlation name	<p>The name of a table or view that is used in the FROM clause of a query—either its original name, or an alias that is defined in the FROM clause.</p>
cursor	<p>A named linkage to a result set, used to access and update rows from a programming interface. In Adaptive Server Anywhere, cursors support forward and backward movement through the query results. Cursors consist of two parts: the cursor result set, typically defined by a SELECT statement; and the cursor position.</p> <p>☞ See also: <a href="#">“cursor result set” on page 347</a>, <a href="#">“cursor position” on page 347</a>.</p>
cursor position	<p>A pointer to one row within the cursor result set.</p> <p>☞ See also: <a href="#">“cursor” on page 347</a>, <a href="#">“cursor result set” on page 347</a>.</p>
cursor result set	<p>The set of rows resulting from a query that is associated with a cursor.</p> <p>☞ See also: <a href="#">“cursor” on page 347</a>, <a href="#">“cursor position” on page 347</a>.</p>
data definition language (DDL)	<p>The subset of SQL statements for modeling the structure of a database. DDL statements create, modify, and remove database objects, including users.</p>
data type	<p>The format of data, such as CHAR or NUMERIC. In the ANSI SQL standard, data types can also include a restriction on size, character set, and collation.</p> <p>☞ See also: <a href="#">“user-defined data type” on page 360</a>.</p>

---

data manipulation language (DML)	The subset of SQL statements for retrieving and updating the contents of a database.
database	<p>A collection of tables that are related by primary and foreign keys. The tables hold the information in the database. The tables and keys together define the structure of the database. A database-management system accesses this information.</p> <p>☞ See also: <a href="#">“foreign key” on page 349</a>, <a href="#">“primary key” on page 355</a>, <a href="#">“database-management system (DBMS)” on page 348</a>, <a href="#">“relational database-management system (RDBMS)” on page 356</a>.</p>
database administrator (DBA)	The user with the permissions required to maintain the database. The DBA is generally responsible for all changes to a database schema, and for managing users and user groups. The role of database administrator is automatically built into databases as user ID DBA with password SQL.
database connection	A communication channel between a client application and the database. A valid user ID and password are required to establish a connection. The privileges granted to the user ID determine the actions that can be carried out during the connection.
database file	<p>A database is held in one or more database files. There is an initial file, and subsequent files are called dbspaces. Each table, including its indexes, must be contained within a single database file.</p> <p>☞ See also: <a href="#">“dbspace” on page 349</a>.</p>
database-management system (DBMS)	<p>A collection of programs that allow you to create and use databases.</p> <p>☞ See also: <a href="#">“relational database-management system (RDBMS)” on page 356</a>.</p>
database name	<p>The name given to a database when it is loaded by a server. The default database name is the root of the initial database file.</p> <p>☞ See also: <a href="#">“database file” on page 348</a>.</p>
database object	A component of a database that contains or receives information. Tables, indexes, views, procedures, and triggers are database objects.
database owner (dbo)	<p>A special user that owns the system objects not owned by SYS.</p> <p>☞ See also: <a href="#">“database administrator (DBA)” on page 348</a>, <a href="#">“SYS” on page 359</a>.</p>
database server	A computer program that regulates all access to information in a database. Adaptive Server Anywhere provides two types of servers: network servers and personal servers.
DBA authority	The level of permission that enables a user to carry out administrative



---

	activity in the database. The DBA user has DBA authority by default. ☞ See also: <a href="#">“database administrator (DBA)” on page 348.</a>
dbspace	An additional database file that creates more space for data. A database can be held in up to 13 separate files (an initial file and 12 dbspaces). Each table, together with its indexes, must be contained in a single database file. The SQL command CREATE DBSPACE adds a new file to the database. ☞ See also: <a href="#">“database file” on page 348.</a>
deadlock	A state where a set of transactions arrives at a place where none can proceed.
download	The stage in synchronization where data is transferred from the consolidated database to a remote database.
embedded SQL	A programming interface for C programs. Adaptive Server Anywhere embedded SQL is an implementation of the ANSI and IBM standard.
external login	An alternate login name and password used when communicating with a remote server. By default, Adaptive Server Anywhere uses the names and passwords of its clients whenever it connects to a remote server on behalf of those clients. However, this default can be overridden by creating external logins. External logins are alternate login names and passwords used when communicating with a remote server.
extraction	In SQL Remote replication, the act of unloading the appropriate structure and data from the consolidated database. This information is used to initialize the remote database.  In MobiLink synchronization, the act of unloading the appropriate structure and data from a reference database. ☞ See also: <a href="#">“replication” on page 357.</a>
failover	Switching to a redundant or standby server, system, or network on failure or unplanned termination of the active server, system, or network. Failover happens automatically, and is often built-in to continuously available systems.
FILE	In SQL Remote replication, a message system that uses shared files for exchanging replication messages. This is useful for testing and for installations without an explicit message-transport system (such as MAPI). ☞ See also: <a href="#">“replication” on page 357</a> , <a href="#">“MAPI” on page 352.</a>
file-based download	In MobiLink, a way to synchronize data in which downloads are distributed as files, allowing offline distribution of synchronization changes.
foreign key	One or more columns in a table that duplicate the primary key values in another table. Foreign keys establish relationships between tables.

---

	<p>☞ See also: <a href="#">“primary key” on page 355</a>, <a href="#">“foreign table” on page 350</a>.</p>
foreign key constraint	<p>A restriction on a column or set of columns that specifies how the data in the table relates to the data in some other table. Imposing a foreign key constraint on a set of columns makes those columns the foreign key.</p> <p>☞ See also: <a href="#">“constraint” on page 347</a>, <a href="#">“check constraint” on page 345</a>, <a href="#">“primary key constraint” on page 355</a>, <a href="#">“unique constraint” on page 360</a>.</p>
foreign table	<p>The table containing the foreign key.</p> <p>☞ See also: <a href="#">“foreign key” on page 349</a>.</p>
full backup	<p>A backup of the entire database, and optionally, the transaction log. A full backup contains all the information in the database and thus provides protection in the event of a system or media failure.</p> <p>☞ See also: <a href="#">“incremental backup” on page 350</a>.</p>
generated join condition	<p>A restriction on join results based on the keyword <code>KEY</code> or <code>NATURAL</code>. For a natural join, the generated join condition is based on common column names in the two tables. For a key join, the condition is based on a foreign key relationship between the two tables.</p> <p>☞ See also: <a href="#">“join” on page 351</a>, <a href="#">“join condition” on page 351</a>.</p>
global temporary table	<p>A type of temporary table for which data definitions are visible to all users until explicitly dropped. Global temporary tables let each user open their own identical instance of a table. By default, rows are deleted on commit, and rows are always deleted when the connection is ended.</p> <p>☞ See also: <a href="#">“temporary table” on page 359</a>, <a href="#">“local temporary table” on page 352</a>.</p>
grant option	<p>The level of permission that allows a user to grant permissions to other users.</p>
identifier	<p>A string of characters used to reference a database object, such as a table or column. An identifier may contain any character from A through Z, a through z, 0 through 9, underscore (<code>_</code>), at sign (<code>@</code>), number sign (<code>#</code>), or dollar sign (<code>\$</code>).</p>
incremental backup	<p>A backup of the transaction log only, typically used between full backups.</p> <p>☞ See also: <a href="#">“transaction log” on page 359</a>.</p>
index	<p>A sorted set of keys and pointers associated with one or more columns in a base table. An index on one or more columns of a table can improve performance.</p>
InfoMaker	<p>A reporting and data maintenance tool that lets you create sophisticated forms, reports, graphs, cross-tabs, and tables, as well as applications that use</p>

---

	these reports as building blocks.
inner join	<p>A join in which rows appear in the result set only if both tables satisfy the join condition. Inner joins are the default.</p> <p>☞ See also: <a href="#">“join” on page 351</a>, <a href="#">“outer join” on page 354</a>.</p>
integrated login	<p>A login feature that allows the same single user ID and password to be used for operating system logins, network logins, and database connections.</p>
integrity	<p>Adherence to rules that ensure that data is correct and accurate, and that the relational structure of the database is intact.</p> <p>☞ See also: <a href="#">“referential integrity” on page 356</a>.</p>
Interactive SQL	<p>An Adaptive Server Anywhere application that allows you to query and alter data in your database, and modify the structure of your database. Interactive SQL provides a pane for you to enter SQL statements, as well as panes that display information about how the query was processed and the result set.</p>
isolation level	<p>The degree to which operations in one transaction are visible to operations in other concurrent transactions. There are four isolation levels, numbered 0 through 3. Level 3 provides the highest level of isolation. Level 0 is the default setting.</p>
JAR file	<p>Java archive file. A compressed file format consisting of a collection of one or more packages used for Java applications. It includes all the resources necessary to install and run a Java program in a single compressed file.</p>
Java class	<p>The main structural unit of code in Java. It is a collection of procedures and variables grouped together because they all relate to a specific, identifiable category.</p>
jConnect	<p>A Java implementation of the JavaSoft JDBC standard. It provides Java developers with native database access in multi-tier and heterogeneous environments.</p> <p>☞ See also: <a href="#">“JDBC” on page 351</a>.</p>
JDBC	<p>Java Database Connectivity. A SQL-language programming interface that allows Java applications to access relational data.</p>
join	<p>A basic operation in a relational system that links the rows in two or more tables by comparing the values in specified columns.</p>
join condition	<p>A restriction that affects join results. You specify a join condition by inserting an ON clause or WHERE clause immediately after the join. In the case of natural and key joins, Adaptive Server Anywhere generates a join condition.</p>

---

	<p>☞ See also: <a href="#">“join” on page 351</a>, <a href="#">“generated join condition” on page 350</a>.</p>
join type	<p>Adaptive Server Anywhere provides four types of joins: cross join, key join, natural join, and joins using an ON clause.</p> <p>☞ See also: <a href="#">“join” on page 351</a>.</p>
local temporary table	<p>A type of temporary table that exists only for the duration of a compound statement or until the end of the connection. Local temporary tables are useful when you need to load a set of data only once. By default, rows are deleted on commit.</p> <p>☞ See also: <a href="#">“temporary table” on page 359</a>, <a href="#">“global temporary table” on page 350</a>.</p>
lock	<p>A concurrency control mechanism that protects the integrity of data during the simultaneous execution of multiple transactions. Adaptive Server Anywhere automatically applies locks to prevent two connections from changing the same data at the same time, and to prevent other connections from reading data that is in the process of being changed.</p> <p>You control locking by setting the isolation level.</p> <p>☞ See also: <a href="#">“isolation level” on page 351</a>, <a href="#">“concurrency” on page 346</a>, <a href="#">“integrity” on page 351</a>.</p>
log file	<p>A log of transactions maintained by Adaptive Server Anywhere. The log file is used to ensure that the database is recoverable in the event of a system or media failure, to improve database performance, and to allow data replication using SQL Remote.</p> <p>☞ See also: <a href="#">“transaction log” on page 359</a>, <a href="#">“transaction log mirror” on page 360</a>, <a href="#">“full backup” on page 350</a>.</p>
LTM	<p>Log Transfer Manager. See <a href="#">“Replication Agent” on page 357</a>.</p>
MAPI	<p>Microsoft’s Messaging Application Programming Interface. A message system used in several popular e-mail systems such as Microsoft Mail.</p>
message system	<p>In SQL Remote replication, a protocol for exchanging messages between the consolidated database and a remote database. Adaptive Server Anywhere includes support for the following message systems: FILE, MAPI, FTP, SMTP, and VIM.</p> <p>☞ See also: <a href="#">“replication” on page 357</a>, <a href="#">“FILE” on page 349</a>, <a href="#">“MAPI” on page 352</a>.</p>
message type	<p>In SQL Remote replication, a database object that specifies how remote users communicate with the publisher of a consolidated database. A consolidated database may have several message types defined for it; this allows different</p>

---

	remote users to communicate with it using different message systems. ☞ See also: <a href="#">“replication” on page 357</a> , <a href="#">“consolidated database” on page 346</a> , <a href="#">“MAPI” on page 352</a> .
metadata	Data about data. Metadata describes the nature and content of other data. ☞ See also: <a href="#">“schema” on page 358</a> .
MobiLink	A session-based synchronization technology designed to synchronize UltraLite and Adaptive Server Anywhere databases with many industry-standard SQL database-management systems from Sybase and other vendors. ☞ See also: <a href="#">“UltraLite” on page 360</a> .
MobiLink client	There are two kinds of MobiLink clients. For Adaptive Server Anywhere remote databases, the MobiLink client is the dbmsync command line utility. For UltraLite remote databases, the MobiLink client is built in to the UltraLite runtime library.
MobiLink user	A MobiLink user is a name that uniquely identifies a MobiLink remote database in the synchronization system. The client supplies this name and, optionally, an associated password when it connects to the MobiLink synchronization server. MobiLink user names are entirely independent of database user names.
NetWare	A widely-used network operating system defined by Novell. NetWare generally employs the IPX/SPX protocol, although the TCP/IP protocol may also be used.
network server	A database server that accepts connections from computers sharing a common network. ☞ See also: <a href="#">“personal server” on page 354</a> .
normalization	The refinement of a database structure to eliminate redundancy and improve organization according to rules based on relational database theory.
object tree	In Sybase Central, the hierarchy of database objects. The top level of the object tree shows all products that your version of Sybase Central supports. Each product expands to reveal its own sub-tree of objects. ☞ See also: <a href="#">“Sybase Central” on page 359</a> .
ODBC	Open Database Connectivity. A standard Windows interface to database-management systems. ODBC is one of several interfaces supported by Adaptive Server Anywhere.
ODBC Administrator	A Microsoft program included with Windows operating systems for setting up ODBC data sources.

---

ODBC data source	A specification of the data a user wants to access via ODBC, and the information needed to get to that data.
outer join	A join that preserves all the rows in a table. Adaptive Server Anywhere supports left, right, and full outer joins. A left outer join preserves the rows in the table to the left of the join operator, and returns a null when a row in the right table does not satisfy the join condition. A full outer join preserves all the rows from both tables.  ☞ See also: <a href="#">“join” on page 351</a> , <a href="#">“inner join” on page 351</a> .
package	In Java, a collection of sets of related classes.
passthrough	In SQL Remote replication, a mode by which the publisher of the consolidated database can directly change remote databases with SQL statements. Passthrough is set up for specific remotes. In normal passthrough mode, all database changes made at the consolidated database are passed through to the selected remote databases. In passthrough only mode, the changes are made at the remote databases, but not at the consolidated database.
performance statistic	A value reflecting the performance of the database system. The CURRREAD statistic, for example, represents the number of file reads issued by the engine that have not yet completed.
personal server	A database server that runs on the same computer as the client application. A personal database server is typically used by a single user on a single computer, but it can support several concurrent connections from that user.
plug-in module	In Sybase Central, a way to access and administer a product. Plug-ins are usually installed and registered automatically with Sybase Central when you install the respective product. Typically, a plug-in appears as a top-level container, in the Sybase Central main window, using the name of the product itself; for example, Adaptive Server Anywhere.  ☞ See also: <a href="#">“Sybase Central” on page 359</a> .
PowerDesigner	A database modeling application. PowerDesigner provides a structured approach to designing a database or data warehouse.
PowerDynamo	A Sybase product for building and managing a Web application linked to a database.
PowerJ	A Sybase product for developing Java applications.
predicate	A conditional expression that is optionally combined with the logical operators AND and OR to make up the set of conditions in a WHERE or HAVING clause. In SQL, a predicate that evaluates to UNKNOWN is interpreted as FALSE.

---

primary key	<p>A column or list of columns whose values uniquely identify every row in the table.</p> <p>☞ See also <a href="#">“foreign key” on page 349</a>.</p>
primary key constraint	<p>A uniqueness constraint on the primary key columns. A table can have only one primary key constraint.</p> <p>☞ See also: <a href="#">“constraint” on page 347</a>, <a href="#">“check constraint” on page 345</a>, <a href="#">“foreign key constraint” on page 350</a>, <a href="#">“unique constraint” on page 360</a>, <a href="#">“integrity” on page 351</a>.</p>
primary table	<p>The table containing the primary key in a foreign key relationship.</p>
proxy table	<p>A local table containing metadata used to access a table on a remote database server as if it were a local table.</p> <p>☞ See also: <a href="#">“metadata” on page 353</a>.</p>
publication	<p>In SQL Remote or MobiLink, a database object that identifies replicated data. In MobiLink, publications exist only on the clients. A publication consists of articles. Periodically, the changes made to each publication are replicated to all subscribers to that publication. SQL Remote users can receive a publication by subscribing to it. MobiLink users can synchronize a publication by creating a synchronization subscription to it.</p> <p>☞ See also: <a href="#">“replication” on page 357</a>, <a href="#">“article” on page 345</a>, <a href="#">“publication update” on page 355</a>.</p>
publication update	<p>In SQL Remote replication, a list of changes made to one or more publications in one database. A publication update is sent periodically as part of a replication message to the remote database(s).</p> <p>☞ See also: <a href="#">“replication” on page 357</a>, <a href="#">“publication” on page 355</a>.</p>
publisher	<p>In SQL Remote replication, the single user in a database who can exchange replication messages with other replicating databases.</p> <p>☞ See also: <a href="#">“replication” on page 357</a>.</p>
query	<p>A SQL statement or group of SQL statements that access and/or manipulate data in a database.</p> <p>☞ See also: <a href="#">“SQL” on page 358</a>.</p>
Redirector	<p>A web server plug-in that routes requests and responses between a client and the MobiLink synchronization server. This plug-in also implements load-balancing and failover mechanisms.</p>
reference database	<p>In MobiLink, an Adaptive Server Anywhere database used in the development of UltraLite clients. You can use a single Adaptive Server</p>

---

	<p>Anywhere database as both reference and consolidated database during development. Databases made with other products cannot be used as reference databases.</p>
referential integrity	<p>Adherence to rules governing data consistency, specifically the relationships between the primary and foreign key values in different tables. To have referential integrity, the values in each foreign key must correspond to the primary key values of a row in the referenced table.</p> <p>☞ See also: <a href="#">“primary key” on page 355</a>, <a href="#">“foreign key” on page 349</a>.</p>
relational database-management system (RDBMS)	<p>A type of database-management system that stores data in the form of related tables.</p> <p>☞ See also: <a href="#">“database-management system (DBMS)” on page 348</a>.</p>
remote database	<p>In SQL Remote replication or MobiLink synchronization, a database that exchanges data with a consolidated database. Remote databases may share all or some of the data in the consolidated database.</p> <p>☞ See also: <a href="#">“replication” on page 357</a>, <a href="#">“consolidated database” on page 346</a>.</p>
remote DBA authority	<p>In SQL Remote, a level of permission required by the Message Agent. In MobiLink, a level of permission required by the Adaptive Server Anywhere synchronization client (<i>dbmsync</i>). When the Message Agent or synchronization client connects as a user who has this authority, it has full DBA access. The user ID has no additional permissions when not connected through the Message Agent or synchronization client.</p> <p>☞ See also: <a href="#">“DBA authority” on page 348</a>.</p>
remote permission	<p>In SQL Remote replication, the permission to exchange replication messages with the publishing database. Granting remote permissions to a user makes that user a remote user. You must specify a message type, an appropriate remote address, and a replication frequency. In general terms, remote permissions can also refer to any user involved in SQL Remote replication (for example, the consolidated publisher and remote publisher).</p> <p>☞ See also: <a href="#">“replication” on page 357</a>.</p>
remote user	<p>In SQL Remote replication, a database user in the consolidated database that has been granted remote permissions and is associated with one particular remote database in the replication setup. To create a remote user, an ordinary database user is granted remote permissions. Doing so not only identifies the existence of a particular remote database, but also specifies the message type and address with which to communicate with that particular remote site.</p> <p>When remote databases are created by means of extraction from a</p>



---

	<p>consolidated database, each remote user in the consolidated database becomes the publisher of the data in one particular remote database.</p> <p>☞ See also: <a href="#">“SQL Remote” on page 358</a>, <a href="#">“consolidated database” on page 346</a>, <a href="#">“publisher” on page 355</a>.</p>
replication	<p>The sharing of data among physically distinct databases. Sybase has three replication technologies: MobiLink, SQL Remote, and Replication Server.</p>
Replication Agent	<p>In Replication Server, the program, also called Log Transfer Manager (LTM), that reads a database transaction log and sends committed changes to Replication Server.</p>
replication frequency	<p>In SQL Remote replication, a setting for each remote user that determines how often the publisher’s message agent should send replication messages to that remote user.</p> <p>☞ See also: <a href="#">“replication” on page 357</a>, <a href="#">“remote user” on page 356</a>.</p>
replication message	<p>In SQL Remote or Replication Server, a communication sent between a publishing database and a subscribing database. Messages contain data, passthrough statements, and information required by the replication system.</p> <p>☞ See also: <a href="#">“passthrough” on page 354</a>, <a href="#">“replication” on page 357</a>, <a href="#">“publication update” on page 355</a>.</p>
Replication Server	<p>A Sybase connection-based replication technology that works with Adaptive Server Anywhere and Adaptive Server Enterprise. It is intended for near-real time replication between a small number of databases.</p>
role	<p>In conceptual database modeling, a verb or phrase that describes a relationship from one point of view. You can describe each relationship with two roles. Examples of roles are “contains” and “is a member of.”</p>
role name	<p>The name of a foreign key. This is called a role name because it names the relationship between the foreign table and primary table. By default, the role name is the table name, unless another foreign key is already using that name, in which case the default role name is the table name followed by a three-digit unique number. You can also create the name yourself.</p> <p>☞ See also: <a href="#">“foreign key” on page 349</a>.</p>
rollback log	<p>A record of the changes made during each uncommitted transaction. In the event of a ROLLBACK request or a system failure, uncommitted transactions are reversed out of the database, returning the database to its former state. Each transaction has a separate rollback log, which is deleted when the transaction is complete.</p> <p>☞ See also: <a href="#">“transaction” on page 359</a>.</p>

---

row-level trigger	<p>A trigger that executes once for each row that is changed.</p> <p>☞ See also: <a href="#">“trigger” on page 360</a>, <a href="#">“statement-level trigger” on page 358</a>.</p>
schema	<p>The structure of a database, including tables, columns, and indexes, and the relationships between them.</p>
scripts	<p>In MobiLink, code written to handle MobiLink events. Scripts programmatically control data exchange to meet business needs.</p>
server-initiated synchronization	<p>A way to initiate MobiLink synchronization programmatically from the consolidated database.</p>
service	<p>In Windows operating systems, a way of running applications when the user ID running the application is not logged on.</p>
session-based synchronization	<p>A type of synchronization where synchronization results in consistent data representation across both the consolidated and remote databases. MobiLink is session-based.</p>
SQL	<p>The language used to communicate with relational databases. ANSI has defined standards for SQL, the latest of which is SQL-99 (also called SQL3). SQL stands, unofficially, for Structured Query Language.</p>
SQL Remote	<p>A message-based replication technology for two-way replication between consolidated and remote databases. The consolidated database must be Adaptive Server Anywhere or Adaptive Server Enterprise. The remote databases must be Adaptive Server Anywhere.</p>
SQL statement	<p>A string containing SQL keywords designed for passing instructions to a DBMS.</p> <p>☞ See also: <a href="#">“schema” on page 358</a>, <a href="#">“SQL” on page 358</a>, <a href="#">“database-management system (DBMS)” on page 348</a>.</p>
statement-level trigger	<p>A trigger that executes after the entire triggering statement is completed.</p> <p>☞ See also: <a href="#">“trigger” on page 360</a>, <a href="#">“row-level trigger” on page 358</a>.</p>
stored procedure	<p>A program comprised of a sequence of SQL instructions, stored in the database and used to perform a particular task.</p>
subquery	<p>A SELECT statement that is nested inside another SELECT, INSERT, UPDATE, or DELETE statement, or another subquery.</p> <p>There are two types of subquery: correlated and nested.</p>
subscription	<p>In SQL Remote replication, a link between a publication and a remote user, allowing the user to exchange updates on that publication with the consolidated database.</p> <p>In MobiLink synchronization, a synchronization subscription is a link in a</p>

---

	<p>client database between a publication and a MobiLink user allowing the data described by the publication to be synchronized.</p> <p>☞ See also: <a href="#">“publication” on page 355</a>, <a href="#">“remote user” on page 356</a>, <a href="#">“MobiLink user” on page 353</a>.</p>
Sybase Central	<p>A database management tool that provides Adaptive Server Anywhere database settings, properties, and utilities in a graphical user interface. Sybase Central can also be used for managing other Sybase products, including MobiLink.</p>
synchronization	<p>The process of replicating data between databases using MobiLink technology.</p> <p>In SQL Remote, synchronization is used exclusively to denote the process of initializing a remote database with an initial set of data.</p> <p>☞ See also: <a href="#">“MobiLink” on page 353</a>, <a href="#">“SQL Remote” on page 358</a>.</p>
SYS	<p>A special user that owns most of the system objects. You cannot log in as SYS.</p>
system object	<p>Database objects owned by SYS or dbo.</p>
system table	<p>A table, owned by SYS or dbo, that holds metadata. System tables, also known as data dictionary tables, are created and maintained by the database server.</p>
system view	<p>A type of view, included in every database, that presents the information held in the system tables in an easily understood format.</p>
temporary table	<p>A table that is created for the temporary storage of data. There are two types: global and local.</p> <p>☞ See also: <a href="#">“local temporary table” on page 352</a>, <a href="#">“global temporary table” on page 350</a>.</p>
transaction	<p>A sequence of SQL statements that comprise a logical unit of work. A transaction is processed in its entirety or not at all. Adaptive Server Anywhere supports transaction processing, with locking features built in to allow concurrent transactions to access the database without corrupting the data. Transactions end either with a COMMIT statement, which makes the changes to the data permanent, or a ROLLBACK statement, which undoes all the changes made during the transaction.</p>
transaction log	<p>A file storing all changes made to a database, in the order in which they are made. It improves performance and allows data recovery in the event the database file is damaged. For best results, the transaction log should be kept on a different device from the database files.</p>

---

transaction log mirror	<p>An optional identical copy of the transaction log file, maintained simultaneously. Every time a database change is written to the transaction log file, it is also written to the transaction log mirror file.</p> <p>A mirror file should be kept on a separate device from the transaction log, so that if either device fails, the other copy of the log keeps the data safe for recovery.</p> <p>☞ See also: <a href="#">“transaction log” on page 359</a>.</p>
transactional integrity	<p>In MobiLink, the guaranteed maintenance of transactions across the synchronization system. Either a complete transaction is synchronized, or no part of the transaction is synchronized.</p>
trigger	<p>A special form of stored procedure executed automatically when a user executes a query that modifies the data.</p> <p>☞ See also: <a href="#">“row-level trigger” on page 358</a>, <a href="#">“statement-level trigger” on page 358</a>, <a href="#">“conflict trigger” on page 346</a>, <a href="#">“integrity” on page 351</a>.</p>
UltraLite	<p>A deployment technology for Adaptive Server Anywhere databases, aimed at small, mobile, and embedded devices. Intended platforms include cell phones, pagers, and personal organizers.</p>
unique constraint	<p>A restriction on a column or set of columns requiring that all non-null values are different. A table can have multiple unique constraints.</p> <p>☞ See also: <a href="#">“foreign key constraint” on page 350</a>, <a href="#">“primary key constraint” on page 355</a>, <a href="#">“constraint” on page 347</a>.</p>
unload	<p>Unloading a database exports the structure and/or data of the database to text files (SQL command files for the structure, and ASCII comma-separated files for the data). You unload a database with the Unload utility.</p> <p>In addition, you can unload selected portions of your data using the UNLOAD statement.</p>
upload	<p>The stage in synchronization where data is transferred from a remote database to a consolidated database.</p>
user-defined data type	<p>A data type that users create to specify a base data type, and optionally a default value, check condition, and nullability. User-defined data types, also called user-defined domains, can be applied to columns to enforce consistency throughout the database.</p> <p>☞ See also: <a href="#">“data type” on page 347</a>.</p>
validate	<p>To test for particular types of file corruption of a database, table, or index.</p>
view	<p>A SELECT statement that is stored in the database as an object. It allows users to see a subset of rows or columns from one or more tables. Each time</p>

a user uses a view of a particular table, or combination of tables, it is recomputed from the information stored in those tables. Views are useful for security purposes, and to tailor the appearance of database information to make data access straightforward.

- Windows The Microsoft Windows family of operating systems, including Windows 95, Windows 98, Windows Me, Windows CE, Windows NT, Windows 2000, and Windows XP.
- Windows CE A family of operating systems produced by Microsoft for mobile devices.
- work table An internal storage area for interim results during query optimization.
- write file A file used to record changes to a read-only database. Often used with compressed databases.

☞ See also: [“compressed database file” on page 346](#).



---

# Index

## Symbols

-? server option  
    unsupported on Windows CE 84

## A

accessibility  
    accessibility enablement 5  
ActiveSync  
    required version for ASA 39  
    supported versions 109  
Adaptive Server Anywhere  
    about 18  
    applications 15  
    compared to UltraLite 18, 19  
    configuring databases for Windows CE 53  
    design goals 18  
    features not supported on Windows CE 80  
    glossary definition 345  
    hallmarks 12  
    installing SQL Anywhere Studio 38  
    intended uses 11  
    internals 136  
    introduction 10  
    programming interfaces 131  
    system requirements 13  
    using SQL Anywhere Studio 37  
add synchronizing table script wizard  
    using 288  
add version wizard  
    using 287  
adding  
    new rows in Interactive SQL 229  
    rows 185  
administration tools  
    features not supported on Windows CE 83  
administration utilities  
    using on Windows CE 69  
ADO  
    data control 340  
    development tools 135  
ADOCE Sample

    using 48  
ADO.NET Sample  
    using 46  
aggregate functions  
    applying to grouped data 178  
    introduction 177  
aliases  
    for columns 145  
alphabetical order  
    ORDER BY clause 148  
ALTER DATABASE statement  
    limited on Windows CE 89  
ALTER statement  
    automatic commit 188  
ALTER WRITEFILE statement  
    unsupported on Windows CE 87  
APIs  
    Adaptive Server Anywhere 131  
    ADO 135  
    embedded SQL 133  
    JDBC 135  
    ODBC 133  
    OLE DB 135  
    Open Client 134  
ARM chip  
    Windows CE 111  
ARM processors  
    support for 111  
ARM V4T mode  
    Windows CE 111  
ARMV4T  
    platform support 111  
articles  
    glossary definition 345  
ASA  
    glossary definition 345  
ASA Server Example  
    using 45  
asademo.db file  
    about 198  
atomic transaction  
    glossary definition 345

attributes			
tables	119		
AUTO_COMMIT option			
grouping changes in Interactive SQL	188		
automatic commit			
ALTER statement	188		
COMMENT statement	188		
data definition statements	188		
DROP statement	188		
availability			
Adaptive Server Anywhere components	15		
<b>B</b>			
backup and data recovery			
strategies for Windows CE	69		
backup database wizard			
unsupported on Windows CE	90		
using	258		
BACKUP statement			
limited on Windows CE	89		
backups			
introduction	258		
performing on Windows CE	69		
running database	258		
Sybase Central	258		
base tables	121		
glossary definition	345		
BETWEEN conditions			
WHERE clause	155		
binary large objects			
about	271		
bitmaps			
storing as blobs	271		
blobs			
about	271		
business rules	324		
glossary definition	345		
<b>C</b>			
-ca option			
unsupported on Windows CE	84		
canceling Interactive SQL commands	232		
cardinality			
relationships and	274		
case sensitivity			
SQL	143		
table names	143		
CE			
processors supported	111		
versions supported	111		
Certicom			
ordering encryption software	5		
-ch option			
unsupported on Windows CE	84		
challenges for replication technologies	22		
change log file settings wizard			
unsupported on Windows CE	90		
characteristics			
MobiLink synchronization	32		
Replication Server	33		
SQL Remote	32		
check constraints			
glossary definition	345		
checking			
data integrity	191		
checkpoints			
glossary definition	345		
choosing a replication technology	29		
clearing			
SQL Statements pane	232		
client/server			
defined	127		
glossary definition	345		
collations			
glossary definition	345		
columns			
about	118		
aliases	145		
allowing NULL	271		
calculated	145		
looking up in Interactive SQL	235		
ordering	145		
primary key	251		
selecting from a table	145		
combining			
multiple statements in Interactive SQL	233		
command files			
building	233		
glossary definition	346		
overview	233		
SQL Statements pane	233		
command history dialog			
recalling commands in Interactive SQL	236		
using in Interactive SQL	236		



command line utilities			
introduction	15		
command sequence communication protocol			
about	132		
diagram	131		
commands			
canceling in Interactive SQL	232		
editing in Interactive SQL	236		
executing in Interactive SQL	143, 232		
getting in Interactive SQL	143		
interrupting in Interactive SQL	232		
loading in Interactive SQL	143		
logging in Interactive SQL	238		
recalling in Interactive SQL	236		
saving in Interactive SQL	143		
stopping in Interactive SQL	232		
COMMENT statement			
automatic commit	188		
COMMIT statement			
about	189		
transactions	188		
communication protocols			
Adaptive Server Anywhere	132		
communication stream			
glossary definition	346		
comparisons			
about	151		
introduction	152		
using subqueries	170		
completing transactions			
about	188		
components			
availability	15		
compound search conditions			
using	155		
compress database wizard			
unsupported on Windows CE	90		
compressed database files			
glossary definition	346		
computed columns			
adding to new rows in Interactive SQL	229		
recalculating in Interactive SQL	229		
updating in Interactive SQL	229		
conceptual database models			
definition of	263		
concurrency			
glossary definition	346		
conditions			
GROUP BY clause	180		
pattern matching	153		
search	151, 155		
configuring			
databases for Windows CE	53		
configuring databases			
Windows CE	53		
conflict triggers			
glossary definition	346		
connecting			
Windows CE	39		
connecting to a database on Windows CE			
about	39		
connecting to your Windows CE device			
about	39		
connecting your application to its database			
about	207		
connection IDs			
glossary definition	346		
connection profiles			
glossary definition	346		
connections			
connecting to a database on Windows CE	39		
from Sybase Central	243		
introduction	208		
sample database asademo.db	220		
consolidated databases			
glossary definition	346		
setting up	296		
supported RDBMSs	107		
constraints			
glossary definition	347		
containers			
in Sybase Central	246		
contention			
glossary definition	347		
conventions			
documentation	xvi		
convoy			
glossary definition	347		
copying			
databases to your Windows CE device	59		
rows in Interactive SQL	230		
copying databases			
to your Windows CE device	59		
correlated subqueries			

defined	172	current publisher	
correlation names		setting	297
glossary definition	347	cursor positions	
COUNT function		glossary definition	347
applying to grouped data	178	cursor result sets	
CREATE COMPRESSED DATABASE statement		glossary definition	347
unsupported on Windows CE	87	cursors	
CREATE DATABASE statement		glossary definition	347
creating databases for Windows CE	58	-cw option	
limited on Windows CE	89	unsupported on Windows CE	84
create database wizard		<b>D</b>	
unsupported on Windows CE	90	-d option	
using	264	unsupported on Windows CE	84
CREATE EVENT statement		data recovery	
limited on Windows CE	89	transactions	190
CREATE EXTERNLOGIN statement		data sources	
unsupported on Windows CE	87	introduction	209
CREATE FUNCTION statement		Windows CE	42
limited on Windows CE	89	data types	
CREATE SERVER statement		glossary definition	347
unsupported on Windows CE	87	database administrator	
CREATE TABLE statement		glossary definition	348
limited on Windows CE	89	database connections	
CREATE WRITEFILE statement		glossary definition	348
unsupported on Windows CE	87	database definitions	
creating		PowerDesigner	309
databases	264	database encryption	
databases for Windows CE	55	ordering encryption software	5
forms with InfoMaker	327	Windows CE	54
groups	253	database engine	
InfoMaker database profile	329	defined	125
remote databases for SQL Remote	300	database files	
reports with InfoMaker	327, 330	glossary definition	348
simple ODBC data sources	209	introduction	137
tables	251	database names	
users	253	glossary definition	348
creating databases		database objects	
for Windows CE	55	about	121
creating databases for Windows CE		glossary definition	348
CREATE DATABASE statement	58	database owner	
dbinit utility	57	glossary definition	348
Interactive SQL	58	database profiles	
Sybase Central	55	InfoMaker	329
cross products		database properties	
introduction	159	Windows CE	53
-cs option		database schemas	
unsupported on Windows CE	84		

viewing	246	server	123
database server		SQL	140
options not supported on Windows CE	84	system tables	141
stopping on Windows CE	67	viewing the schema	246
Windows CE	62	dates	
database servers		combining	155
about	198	compound	155
connecting to	197, 207	search conditions	155
differences between personal and network	14	search conditions introduction	152
glossary definition	348	DBA authority	
interface	202	glossary definition	348
internals	136	dbeng9	
running	197, 207	limitations	14
starting	201	starting database servers	201
stopping	205	dbinit utility	
window	202	creating databases for Windows CE	57
database sizes		dbisql utility	
multi-gigabyte databases	11	about	217
database utilities		DBMS	
wizards	258	glossary definition	348
databases		dbremote	
administering	241	supported UNIX platforms	104
backing up	258	supported Windows platforms	101
client application	123	dbspaces	
components	123	glossary definition	349
configuring for Windows CE	53	dbsrv9	
connecting	220	Windows CE	62
connecting to	243	dbxtract	
copying to your Windows CE device	59	unsupported on Windows CE	83
creating	264	DDL	
creating for Windows CE	55	automatic commit	188
design concepts	263	glossary definition	347
designing	261	deadlocks	
erasing from your Windows CE device	60	glossary definition	349
files	137	DELETE statement	
glossary definition	348	about	187
language interface	123	errors	192
managing	241	examples	192
managing on Windows CE using Sybase Central	69	deleting	
managing with Interactive SQL on Windows CE	74	rows from tables	230
		rows using Interactive SQL	230
		tables	252
object properties	247	deployment edition	
objects	121	about	96
queries	140	deployment releases	
relational	118	about	96
reverse engineering	312	designing	

databases	261, 263	reports with InfoMaker	333
using PowerDesigner	309	entering	
designing and building your database		Interactive SQL commands	232
about	261	multiple statements in Interactive SQL	233
developing		erase database wizard	
SQL statements	218	unsupported on Windows CE	90
development platforms		erasing	
UltraLite	99	databases from your Windows CE device	60
dialog boxes		erasing databases	
command history	236	from your Windows CE device	60
distributing reports and forms		errors	
with InfoMaker	327	Interactive SQL	232
DML		ESQL Sample	
glossary definition	348	using	50
documentation		executing	
conventions	xvi	commands in Interactive SQL	232
SQL Anywhere Studio	x	queries more than once	149
domains		external logins	
introduction to	324	glossary definition	349
downloads		external stored procedures	
glossary definition	349	unsupported on Windows CE	80
DROP DATABASE statement		extraction	
unsupported on Windows CE	87	glossary definition	349
DROP SERVER statement		extraction utility	
unsupported on Windows CE	87	unsupported on Windows CE	83
DROP statement		<b>F</b>	
automatic commit	188	failover	
dropping		glossary definition	349
tables	252	Federal Rehabilitation Act	
dynamic cache sizing		section 508	5
unsupported on Windows CE	80	feedback	
<b>E</b>		documentation	xx
editing		providing	xx
table values in Interactive SQL	227, 228	FILE	
embedded databases		glossary definition	349
defined	125	FILE message type	
requirements	11	glossary definition	349
embedded SQL		file sharing message type	
development tools	133	SQL Remote supported UNIX platforms	104
glossary definition	349	SQL Remote supported Windows platforms	101
encryption		file-based downloads	
ordering encryption software	5	glossary definition	349
Windows CE	54	finishing transactions	
ending transactions		about	188
about	188	FIPS	
enhancing		ordering encryption software	5

foreign key constraints			
glossary definition	350		
foreign key creation wizard			
using	275		
foreign keys			
about	120		
defined	119		
glossary definition	349		
inserts	191		
foreign tables			
glossary definition	350		
forms			
creating with InfoMaker	327		
FTP message type			
SQL Remote supported UNIX platforms	104		
SQL Remote supported Windows platforms	101		
full backups			
glossary definition	350		
full releases			
about	96		
function keys			
Interactive SQL	222		
functions			
SOUNDEX function	154		
<b>G</b>			
-gb option			
unsupported on Windows CE	84		
-ge option			
unsupported on Windows CE	84		
generated join conditions			
glossary definition	350		
global temporary tables			
glossary definition	350		
glossary			
Introducing SQL Anywhere Studio	345		
go			
statement delimiter	234		
grant options			
glossary definition	350		
GRANT PUBLISH statement			
introduction	297		
GRANT REMOTE statement			
introduction	297		
GROUP BY clause			
aggregate functions	178		
errors	178		
group creation wizard			
using	253		
grouped data	177		
grouping			
InfoMaker reports	333		
grouping changes into transactions			
about	188		
groups			
adding	253		
adding users, using Sybase Central	254		
creating	253		
-gss option			
unsupported on Windows CE	84		
-gx option			
unsupported on Windows CE	84		
<b>H</b>			
Handheld PC			
Windows CE	111		
hardware requirements			
Adaptive Server Anywhere hallmarks	12		
HAVING clause			
GROUP BY clause and	180		
WHERE clause and	180		
help on help			
accessing information	xiii		
hierarchical database configurations			
about	23		
host platforms			
UltraLite Windows supported platforms	99		
<b>I</b>			
iAnywhere JDBC driver			
unsupported on Windows CE	80		
icons			
used in manuals	xvii		
identifiers			
glossary definition	350		
IN conditions			
using as shortcut for compound search conditions	155		
incremental backups			
glossary definition	350		
Index Consultant			
Interactive SQL	225		
indexes			
glossary definition	350		

introduction	150	executing only selected text in SQL Statements	
inequality		pane	234
testing for	152	function keys	222
InfoMaker		getting commands	143
about	327	glossary definition	351
glossary definition	350	grouping changes into transactions	188
tutorial	327	inserting rows	229
Initialization utility		interrupting commands	232
creating databases for Windows CE	57	introduction	15
inner joins		keyboard shortcuts	222
glossary definition	351	loading commands	143
INPUT statement		logging commands	238
inserting new rows in Interactive SQL	229	looking up column names	235
INSERT statement		looking up procedure names	235
examples	191	looking up table names	235
introduction	185	main window description	220
inserting		managing databases on Windows CE	74
rows into tables in Interactive SQL	229	opening multiple windows	221
INSTALL JAVA statement		options	225
unsupported on Windows CE	87	overview	218
installation		quick start	219
SQL Anywhere Studio	6	recalling commands	236
installing		reported errors	232
SQL Anywhere Studio on Windows CE	38	saving commands	143
integrated logins		SQL Statements pane	143
glossary definition	351	starting	220
integrity		stopping commands	232
checking	191	toolbar description	221
glossary definition	351	updating computed columns	229
Interactive SQL		Interactive SQL	
about	217	creating databases for Windows CE	58
canceling commands	232	interface	
combining multiple statements	233	database server	202
command history dialog	236	internals	
commands overview	232	Adaptive Server Anywhere	136
configuring	225	database server	136
copying rows	230	Introducing Adaptive Server Anywhere and	
deleting rows	230	UltraLite	
displaying a list of procedures	235	overview	9
displaying a list of tables	158, 235	IP address	
displaying data	226	determining on CE devices	39
displaying the Query Editor	222	isolation levels	
editing table values	227, 228	glossary definition	351
effects of exiting	188		
executing all text in SQL Statements pane	222	<b>J</b>	
executing commands	143, 232	JAR files	
		glossary definition	351

Java		CREATE FUNCTION statement	89
separate licensing	5	CREATE TABLE statement	89
Java classes		jConnect	82
glossary definition	351	ODBC	82
Java in the database		Linux	
purchasing	5	Adaptive Server Anywhere components	103
separate licensing	5	Replication Agent supported platforms	105
Java synchronization logic		local temporary tables	
supported UNIX platforms	103	glossary definition	352
supported Windows platforms	100	locks	
jConnect		glossary definition	352
glossary definition	351	log files	
limited on Windows CE	82	glossary definition	352
Windows CE	53	logging	
JDBC		commands in Interactive SQL	238
development tools	135	looking up	
glossary definition	351	columns in Interactive SQL	235
join conditions		procedures in Interactive SQL	235
glossary definition	351	tables in Interactive SQL	235
join types		lookup table name dialog	
glossary definition	352	displaying a list of tables	158
joins		LTM	
glossary definition	351	glossary definition	357
introduction	158		
or subqueries	172		
<b>K</b>		<b>M</b>	
key joins		Mac OS X	
introduction	162	versions supported	111
keyboard shortcuts		Macintosh	
Interactive SQL	222	versions supported	111
keys		managing databases	
about	119	from Interactive SQL for Windows CE	74
foreign	119	from Sybase Central for Windows CE	69
primary	119	MAPI	
<b>L</b>		glossary definition	352
licensing		MAPI message type	
Java	5	glossary definition	352
security option	5	SQL Remote supported UNIX platforms	104
LIKE conditions		SQL Remote supported Windows platforms	101
introduction	153	unsupported on Windows CE	83
limited on Windows CE		Message Agent	
ALTER DATABASE statement	89	supported UNIX platforms	104
BACKUP statement	89	supported Windows platforms	101
CREATE DATABASE statement	89	message systems	
CREATE EVENT statement	89	glossary definition	352
		message types	
		glossary definition	352
		message-based replication	

about	26	glossary definition	353
metadata		network server	
glossary definition	353	glossary definition	353
system tables	141	platform support	14
Microsoft Visual Basic quick start	339	software requirements	13
MIPS chip		newsgroups	
Windows CE	111	technical support	xx
mobile computing		normalization	
requirements	11	glossary definition	353
MobiLink		Novell NetWare	
glossary definition	353	Adaptive Server Anywhere components	98
Tutorial - Introductory	279	n-tier computing	
MobiLink clients		introduction	128
glossary definition	353	NULL	
MobiLink consolidated databases		allowing in columns	185, 271
supported RDBMSs	107	<b>O</b>	
MobiLink synchronization		object trees	
alternative approaches	21	glossary definition	353
features compared	29	ODBC	
supported UNIX platforms	103	about	210
supported Windows platforms	100	data sources	210
tutorials	279	development tools	133
writing publications tutorial	281	glossary definition	353
writing scripts tutorial	281	introduction to data sources	209
writing subscriptions tutorial	281	limited on Windows CE	82
MobiLink user creation wizard		ODBC Administrator	
using	286	glossary definition	353
MobiLink users		ODBC data sources	
glossary definition	353	glossary definition	354
modeling database designs		Windows CE	42
using PowerDesigner	309	ODBC Sample	
multiple databases		using	51
running on a single server	129	OLE DB	
multi-tier computing		development tools	135
introduction	128	OLE DB and ADO programming interfaces	339
multi-user database		OLE DB drivers	
defined	127	supported processors	111
<b>N</b>		ON phrase	
natural joins		introduction	160
errors	164	one-to-many relationships	
introduction	164	definition of	274
.NET synchronization logic		one-to-one relationships	
supported UNIX platforms	103	definition of	274
supported Windows platforms	100	online books (PDF)	
NetWare		using	xiii
Adaptive Server Anywhere components	98	online help	



using	xiii	Palm Computing Platform	
Open Client		UltraLite development environments	99
development tools	134	parameters	
unsupported on Windows CE	80	to functions	177
opening multiple Interactive SQL windows		passthrough mode	
about	221	glossary definition	354
operating systems		passwords	
Adaptive Server Anywhere on UNIX	103	connecting to a new database	266
Adaptive Server Anywhere on Windows	98	using	220
administration utilities on UNIX	103	pattern matching	
administration utilities on Windows	98	introduction	153
Embedded SQL on UNIX	103	PDF	
Embedded SQL on Windows	98	accessing online books	xiii
Interactive SQL on UNIX	103	performance	
Interactive SQL on Windows	98	benefits of replication	22
Java on Windows	98, 103	performance monitoring	
MobiLink UNIX supported platforms	103	Adaptive Server Anywhere hallmarks	12
MobiLink Windows supported platforms	100	performance statistics	
remote data access on UNIX	103	glossary definition	354
remote data access on Windows	98	permissions	
Replication Agent	101, 105	PUBLISH	296
SQL Remote on UNIX	104	REMOTE	296
SQL Remote on Windows	101	setting	255
supported	13, 96	personal server	
Sybase Central on UNIX	103	glossary definition	354
Sybase Central on Windows	98	limitations	14
UltraLite	109	platform support	14
UltraLite Windows supported platforms	99	unsupported on Windows CE	80
versions supported	111	Physical Architect	
optimization of queries		see PowerDesigner	309
Adaptive Server Anywhere hallmarks	12	physical data models	
options dialog		in PowerDesigner	309
Interactive SQL	225	pipelines	
ORDER BY clause		creating with InfoMaker	327
examples	148	platforms	
required to ensure rows always appear in same		MobiLink UNIX operating systems	103
order	149	MobiLink Windows operating systems	100
using indexes to improve performance	150	supported	13
outer joins		supported operating systems	96
glossary definition	354	plug-in modules	
introduction	166	glossary definition	354
outer references		Pocket PC	
defined	172	Windows CE	111
<b>P</b>		PowerDesigner	
packages		about	309
glossary definition	354	glossary definition	354
		tutorial	309

PowerDynamo			
glossary definition	354		
PowerJ			
glossary definition	354		
predicates			
glossary definition	354		
introduction	155		
previewing			
reports with InfoMaker	330		
primary key constraints			
glossary definition	355		
primary keys			
creating	251		
defined	119		
glossary definition	355		
primary tables			
glossary definition	355		
printing			
InfoMaker reports	336		
procedure editor			
Sybase Central	255		
procedures			
looking up in Interactive SQL	235		
program group			
Adaptive Server Anywhere	6		
programming interfaces			
Adaptive Server Anywhere	131		
ADO	135		
embedded SQL	133		
JDBC	135		
ODBC	133		
OLE DB	135		
Open Client	134		
supported in Adaptive Server Anywhere	131		
projections			
defined	141		
properties			
database objects	247		
proxy tables			
glossary definition	355		
introduction	129		
publication creation wizard			
creating MobiLink publications	286		
creating SQL Remote publications	298		
publication updates			
glossary definition	355		
publications			
glossary definition	355		
publish permissions			297
remote permissions			297
publisher			
creating			297
glossary definition			355
<b>Q</b>			
-qi option			
unsupported on Windows CE			84
queries			
defined			140
glossary definition			355
Interactive SQL			232
SELECT statement			142
Query Editor			
displaying in Interactive SQL			222
Interactive SQL			225
quick start			
developing a Visual Basic application			340
Interactive SQL			219
<b>R</b>			
RDBMS			
defined			118
glossary definition			356
recalling			
commands in Interactive SQL			236
recovery			
Adaptive Server Anywhere hallmarks			12
strategies for Windows CE			69
Redirector			
glossary definition			355
supported Windows platforms			100
reference databases			
glossary definition			355
referential integrity			
glossary definition			356
registry			
configuring to run ASA on Windows CE			41
relational database-management system			
defined			118
relational databases			
about			119
concepts			118
terminology			119
relations			

entities	119	enhancing with InfoMaker	333
relationships		grouping InfoMaker reports	333
about	274	previewing with InfoMaker	330
cardinality of	274	printing InfoMaker reports	336
many-to-many	275	saving with InfoMaker	331
one-to-many	274	sorting InfoMaker reports	333
one-to-one	274	requirements	
remote data access		Adaptive Server Anywhere	13
introduction	129	restrictions	
supported UNIX platforms	103	defined	141
supported Windows platforms	98	result sets	
unsupported on Windows CE	80	copying rows	230
remote databases		deleting rows	230
glossary definition	356	editing table values in Interactive SQL	227, 228
setting up for SQL Remote (tutorial)	300	executing a query more than once	149
remote DBA authority		inserting rows	229
glossary definition	356	troubleshooting	149
remote permissions		retrieving	
glossary definition	356	commands in Interactive SQL	236
remote users		reverse engineering	
glossary definition	356	databases	312
REMOVE JAVA statement		role names	
unsupported on Windows CE	87	glossary definition	357
REORGANIZE TABLE statement		roles	
unsupported on Windows CE	87	glossary definition	357
replication		rollback logs	
alternative approaches	21	glossary definition	357
comparison of technologies	29	ROLLBACK statement	
connection-based	26	about	189
glossary definition	357	introduction	187
hierarchical database configurations	23	transactions	188
message-based	26	rolling back	
SQL Remote tutorial	293	transactions	188
Replication Agent		row-level locking	
supported operating systems	105	Adaptive Server Anywhere hallmarks	12
supported platforms	101	row-level triggers	
replication frequency		glossary definition	358
glossary definition	357	rows	
replication messages		about	118
glossary definition	357	adding	185
Replication Server		adding using Interactive SQL	229
characteristics	33	copying in Interactive SQL	230
glossary definition	357	deleting using Interactive SQL	230
replication technologies	29	editing values in Interactive SQL	228
characteristics	31	inserting in Interactive SQL	229
reports		selecting from a table	151
creating with InfoMaker	327, 330	running	

Interactive SQL commands	232	introduction	142
		subqueries	168
<b>S</b>		selecting aggregate data	
-s option		overview	175
unsupported on Windows CE	84	selecting data from database tables	
sample application		overview	139
ADOCE Sample	48	selecting data from multiple tables	
ADO.NET Sample	46	overview	157
ASA Server Example	45	selecting data using subqueries	
ESQL Sample	50	overview	167
ODBC Sample	51	selecting rows from a table	
Windows CE	45	about	151
sample database		selections	
about	198	defined	141
connecting to asademo.db	220	separately-licensable software	
starting asademo.db	201	Java and security	5
Windows CE	45	server name	
saving		introduction	203
reports with InfoMaker	331	server options	
transaction results	188	unsupported option on Windows CE	84
-sc option		server-initiated synchronization	
unsupported on Windows CE	84	glossary definition	358
schema		service creation wizard	
designing databases	310	unsupported on Windows CE	90
viewing	246	services	
schemas		glossary definition	358
defined	264	session-based synchronization	
glossary definition	358	glossary definition	358
scripts		setting up	
glossary definition	358	consolidated databases	296
search conditions		remote databases for SQL Remote (tutorial)	300
date comparisons	152	SMTP message type	
GROUP BY clause	180	SQL Remote supported UNIX platforms	104
introduction	151	SQL Remote supported Windows platforms	101
pattern matching	153	sorting	
shortcuts for	155	InfoMaker reports	333
subqueries	168	query results	148
section 508		SOUNDEX function	
compliance	5	about	154
security option		SPX	
purchasing	5	unsupported on Windows CE	80
separate licensing	5	SQL	
select list		about	140
calculated columns	145	developing queries	218
column names	145	glossary definition	358
SELECT statement		statements not supported on Windows CE	86
Interactive SQL	226	SQL and database computing	

about	140	STOP JAVA statement	
SQL Anywhere Studio		unsupported on Windows CE	87
documentation	x	stopping	
features not supported on Windows CE	80	database servers	205
SQL Modeler		the server on Windows CE	67
see PowerDesigner	309	store-and-forward	
SQL Remote		SQL Remote replication	26
characteristics	32	stored procedures	
features not supported on Windows CE	83	glossary definition	358
glossary definition	358	setting permissions	255
granting PUBLISH permissions	297	viewing	255
granting REMOTE permissions	297	strong encryption	
setting up a consolidated database	296	ordering encryption software	5
setting up a remote database (tutorial)	300	subqueries	
supported UNIX platforms	104	comparisons	170
supported Windows platforms	101	correlated subqueries	172
tutorial	293	glossary definition	358
SQL statements		introduction	168
glossary definition	358	or joins	172
unsupported statements on Windows CE	86	troubleshooting	171
SQL Anywhere Studio		subscriptions	
components	4	glossary definition	358
introduction	4	summarizing data	176
using on Windows CE	37	support	
ssqueue		newsgroups	xx
supported UNIX platforms	104	supported platforms	
supported Windows platforms	101	about	95
ssremote		Adaptive Server Anywhere	13
supported UNIX platforms	104	Adaptive Server Anywhere editions	96
supported Windows platforms	101	MobiLink UNIX supported platforms	103
standalone applications		MobiLink Windows supported platforms	100
defined	125	SQL Anywhere Studio components on UNIX	103
standards		SQL Anywhere Studio components on Windows	98
section 508	5	supported operating systems	96
START JAVA statement		Sybase Central	
unsupported on Windows CE	87	about	241
Start menu		adding table scripts to the consolidated database	288
Adaptive Server Anywhere	6	adding users to groups	254
starting		backing up databases	258
InfoMaker	329	code editor	255
Interactive SQL	220	connecting to a database	243
Sybase Central	243	creating a script version	287
transactions	188	creating databases for Windows CE	55
starting multiple databases		creating groups	253
Windows CE	65	creating MobiLink users	286
statement-level triggers			
glossary definition	358		

creating SQL Remote publications	298	foreign keys	120
creating synchronization subscriptions	286	looking up in Interactive SQL	235
creating users	253	viewing	246
glossary definition	359	target platforms	
introduction	15	UltraLite	109
main window	246	TDS communication protocol	
managing databases on Windows CE	69	diagram	131
starting	243	technical support	
tutorial	241	newsgroups	xx
viewing data	250	temporary files	
wizards	258	introduction	137
wizards not supported on Windows CE	90	temporary tables	
synchronization		glossary definition	359
alternative approaches	21	three-tier computing	
comparison of technologies	29	introduction	128
glossary definition	359	-tmf option	
MobiLink features compared	29	unsupported on Windows CE	84
MobiLink hierarchical database configurations	23	-tmt option	
MobiLink introductory tutorial	279	unsupported on Windows CE	84
SYS		toolbars	
glossary definition	359	Interactive SQL	221
system failures		transaction log	
transactions	190	glossary definition	359
system objects		introduction	137
glossary definition	359	Windows CE	53
system requirements		transaction log mirror	
Adaptive Server Anywhere	13, 98	glossary definition	360
system tables		transaction processing	
defined	141	Adaptive Server Anywhere hallmarks	12
glossary definition	359	data recovery	190
system views		transactional integrity	
glossary definition	359	glossary definition	360
<b>T</b>		transactions	
table values		completing	188
editing in Interactive SQL	227, 228	data recovery	190
tables		glossary definition	359
about	118	grouping changes	188
adding	251	starting	188
adding columns	251	translate log file wizard	
characteristics	118	unsupported on Windows CE	90
creating	251	transport-layer security	
creating a primary key	251	ordering	5
deleting	252	ordering encryption software	5
designing	267	triggers	
dropping	252	glossary definition	360
editing	251	viewing	255
		troubleshooting	

GROUP BY clause	178	-? server option	84
natural joins	164	ALTER WRITEFILE statement	87
result set appears to change	149	backup database wizard	90
subqueries	171	-ca option	84
tuples		-ch option	84
about	119	change log file settings wizard	90
tutorials		compress database wizard	90
connecting to a sample database	197	CREATE COMPRESSED DATABASE statement	87
designing a database	263	87	
developing a Visual Basic application	340	create database wizard	90
InfoMaker	327	CREATE EXTERNLOGIN statement	87
MobiLink synchronization basics	279	CREATE SERVER statement	87
PowerDesigner	309	CREATE WRITEFILE statement	87
SQL Remote	293	-cs option	84
Sybase Central	241	-cw option	84
two-way replication		-d option	84
about	24	dbxtract	83
<b>U</b>		DROP DATABASE statement	87
-u option		DROP SERVER statement	87
unsupported on Windows CE	84	dynamic cache sizing	80
-ua option		erase database wizard	90
unsupported on Windows CE	84	external stored procedures	80
-ud option		extraction utility	83
unsupported on Windows CE	84	-gb option	84
UltraLite		-ge option	84
about	19	-gss option	84
compared to Adaptive Server Anywhere	18, 19	-gx option	84
design goals	19	iAnywhere JDBC driver	80
development platforms	99	INSTALL JAVA statement	87
glossary definition	360	MAPI message type	83
supported Windows platforms	99	Open Client	80
target platforms	109	personal server	80
uncompress database wizard		-qi option	84
unsupported on Windows CE	90	remote data access	80
unique constraints		REMOVE JAVA statement	87
glossary definition	360	REORGANIZE TABLE statement	87
UNIX		-s option	84
Adaptive Server Anywhere components	103	-sc option	84
Replication Agent supported platforms	105	service creation wizard	90
unload		SPX	80
glossary definition	360	START JAVA statement	87
unload database wizard		STOP JAVA statement	87
unsupported on Windows CE	90	-tmf option	84
unsupported features		-tmt option	84
Windows CE limitations	80	translate log file wizard	90
unsupported on Windows CE		-u option	84
		-ua option	84

-ud option	84	values	
uncompress database wizard	90	editing in Interactive SQL	228
unload database wizard	90	viewing	
upgrade database wizard	90	data using Sybase Central	250
-ut option	84	views	
VIM message type	83	glossary definition	360
-y option	84	viewing	255
UPDATE statement		VIM message type	
errors	193	SQL Remote supported UNIX platforms	104
examples	193	SQL Remote supported Windows platforms	101
introduction	186	unsupported on Windows CE	83
updating		Visual Basic	
data	183	quick start	340
values in Interactive SQL	228		
updating the database		<b>W</b>	
overview	183	WHERE clause	
upgrade database wizard		BETWEEN conditions	155
unsupported on Windows CE	90	date comparisons introduction	152
upload		deleting rows	187
glossary definition	360	examples	151
user creation wizard		HAVING clause and	180
using	253	modifying rows in a table	186
user IDs		pattern matching	153
about	220	wildcards	
new databases	266	pattern matching	153
using	220	Windows	
user-defined data types		glossary definition	361
glossary definition	360	supported operating systems	13
users		Windows 98/Me	
adding	253	Adaptive Server Anywhere components	98
adding to groups, using Sybase Central	254	Replication Agent supported platforms	101
creating	253	Windows CE	
managing	253	Adaptive Server Anywhere components	98
using Interactive SQL		glossary definition	361
overview	217	OLE DB support	111
using Interactive SQL to display data		processors supported	111
about	226	Replication Agent supported platforms	101
-ut option		versions supported	111
unsupported on Windows CE	84	Windows NT/2000/XP	
utilities		Adaptive Server Anywhere components	98
introduction	15	Replication Agent supported platforms	101
<b>V</b>		Windows CE	
V4T mode		configuring databases	53
ARM processors	111	configuring SQL Anywhere Studio	38
validate		connecting from a PC	39
glossary definition	360	copying databases to your device	59
		creating databases	55



database properties	53	working with SQL statements in Interactive SQL	
database server	62	about	232
encryption	54	write files	
erasing databases	60	glossary definition	361
ESQL Sample	50	<b>X</b>	
installing SQL Anywhere Studio	38	x86 chip	
jConnect	53	Windows CE	111
managing databases from Interactive SQL	74	XScale processors	
managing databases from Sybase Central	69	support for	111
ODBC data sources	42	<b>Y</b>	
ODBC Sample	51	-y option	
sample applications	45	unsupported on Windows CE	84
sample database	45		
starting multiple databases	65		
starting the server	64		
stopping the server	67		
transaction log	53		
unsupported Adaptive Server Anywhere features			
80			
unsupported administration tools	83		
unsupported database server options	84		
unsupported SQL Anywhere Studio features	80		
unsupported SQL Remote features	83		
unsupported SQL statements	86		
unsupported Sybase Central wizards	90		
using SQL Anywhere Studio	37		
using the administration utilities	69		
using the ADOCE Sample	48		
using the ADO.NET Sample	46		
using the ASA Server Example	45		
wizards			
add synchronizing table script	288		
add version	287		
backup database	258		
create database	264		
foreign key creation	275		
group creation	253		
MobiLink user creation	286		
publication creation	286, 298		
Sybase Central	258		
unsupported Sybase Central wizards on			
Windows CE	90		
user creation	253		
work tables			
glossary definition	361		
workgroup computing			
requirements	11		