



# OS/390 Getting Started Guide Version 2.2

October, 2002

Copyright © 1993–2002 CrossAccess Corporation. All rights reserved.

This publication may not be reproduced, stored in a retrieval system or transmitted in whole or part, in any form or by any means, electronic, photocopy, recording or otherwise, without the prior written consent of CrossAccess Corporation.

The CrossAccess Corporation logo, eXadas Data Integrator, and XDi are trademarks of CrossAccess Corporation. Other products and services referred to herein are or may be the trademarks of their respective owners.

CrossAccess Corporation  
One Tower Lane, Suite 1700  
Oak Brook Terrace IL 60181  
(630) 928-3708 or (800) 427-6774  
FAX: (630) 954-0554

CrossAccess Corporation  
2900 Gordon Ave., Suite 100  
Santa Clara CA 95051  
(408) 735-7545 or (800) 982-9911  
FAX: (408) 735-0328

Technical Support: (800) 982-9911

# Table of Contents

---

<b>Preface</b> .....	<b>vii</b>
<b>Chapter 1 Overview</b> .....	<b>1</b>
<b>Introduction to eXadas</b> .....	<b>1</b>
Product Overview.....	2
<b>Operational Components</b> .....	<b>3</b>
eXadas Server .....	3
Region Controller .....	4
Connection Handlers.....	4
Query Processor.....	5
Logger.....	6
Initialization Services .....	6
eXadas Enterprise Server.....	7
Client Interface Module .....	7
Client Connectors.....	8
<b>Application Components</b> .....	<b>8</b>
<b>Administrative Components</b> .....	<b>9</b>
DataMapper.....	9
<b>Chapter 2 Concepts</b> .....	<b>13</b>
<b>Introduction to Concepts</b> .....	<b>13</b>
<b>Relational Access to Data</b> .....	<b>14</b>
<b>Client/Server Architecture</b> .....	<b>15</b>
<b>Data Sources</b> .....	<b>16</b>
<b>Non-Relational Data Mapping</b> .....	<b>16</b>
<b>eXadas Server and Client Components</b> .....	<b>18</b>
Server Components .....	18
Initialization Services .....	19
Connection Handler Services.....	19
Query Processor Services .....	19
MTO Interface .....	19
Logger Service.....	20
eXadas Server System Exits .....	20
SAF Security Exit .....	20

	SMF Accounting Exit .....	20
	CPU Resource Governor Exit.....	21
	Work Load Manager Exit .....	21
	DB2 Thread Management Exit.....	22
	Record Processing Exit.....	22
	Client Applications (Precompiler) .....	22
	Client Connectors.....	22
	<b>National Language Support .....</b>	<b>23</b>
	<b>Configuration Methodology .....</b>	<b>24</b>
	Server Configuration (CACDSCF).....	24
	Query Processor Configuration (CACQPCF).....	25
	Administrator Configuration (CACADMIN) .....	25
	Methods for Updating Configuration Members .....	25
<b>Chapter 3</b>	<b>System Requirements .....</b>	<b>27</b>
	<b>Introduction to System Requirements.....</b>	<b>27</b>
	<b>Minimum Disk Space Requirements for Installation .....</b>	<b>28</b>
	<b>Operational Disk Space Requirements .....</b>	<b>29</b>
	<b>Minimum Release Level Requirements.....</b>	<b>30</b>
	<b>APF Authorization Requirements .....</b>	<b>30</b>
	<b>Run-time Memory Storage Requirements .....</b>	<b>31</b>
	Storage Requirements for Client Applications .....	31
	Storage Requirements for a Server .....	32
	Temporary Storage Requirements Per User .....	33
	Fifty Concurrent User Storage Requirements.....	33
<b>Chapter 4</b>	<b>OS/390 Installation .....</b>	<b>35</b>
	<b>Introduction to Installation .....</b>	<b>35</b>
	Installing a New Server .....	37
	IBM MQ Series .....	41
	<b>Installing Additional Components.....</b>	<b>42</b>
	<b>Starting the eXadas Server.....</b>	<b>42</b>
	Copying the eXadas Server JCL.....	43
	eXadas Server Configuration Members.....	43
	Configure the eXadas Server for Communications .....	44
	TCP/IP .....	44
	Starting the eXadas Server.....	45
<b>Chapter 5</b>	<b>Bringing Adabas On Line .....</b>	<b>47</b>
	<b>Introduction to Bringing Adabas On Line With eXadas.....</b>	<b>47</b>
	<b>Bringing Adabas On Line.....</b>	<b>48</b>

	Set Up Adabas-Specific Environment .....	48
	Access File Definition Information And Create Logical Tables .....	48
	Loading the eXadas Catalogs.....	49
	Accessing the Adabas Data With SQL (Locally) .....	51
	Accessing the Adabas Data With SQL (Remotely).....	52
<b>Chapter 6</b>	<b>Bringing DB2 On Line.....</b>	<b>55</b>
	<b>Introduction to Bringing DB2 On Line With eXadas.....</b>	<b>55</b>
	<b>Bringing DB2 On Line .....</b>	<b>56</b>
	Setup Interface With DB2.....	56
	Import DB2 Table Definitions And Create Logical Tables .....	57
	Loading the eXadas Catalogs.....	57
	Accessing the DB2 Data With SQL (Locally).....	59
	Accessing the DB2 Data With SQL (Remotely) .....	61
<b>Chapter 7</b>	<b>Bringing Datacom On Line .....</b>	<b>63</b>
	<b>Introduction to Bringing Datacom On Line With eXadas .....</b>	<b>63</b>
	<b>Bringing Datacom On Line .....</b>	<b>64</b>
	Punch out file definition in COBOL format .....	64
	Mapping the Sample Datacom Copybook .....	65
	Loading the eXadas Catalogs.....	68
	Accessing the Datacom Data With SQL (Locally).....	70
	Accessing the Datacom data with SQL (remotely).....	72
<b>Chapter 8</b>	<b>Bringing IDMS On Line.....</b>	<b>73</b>
	<b>Introduction to Bringing IDMS On Line With eXadas .....</b>	<b>73</b>
	<b>Bringing IDMS On Line .....</b>	<b>74</b>
	Punching the Schema and Subschema.....	74
	Mapping the IDMS Schema and Subschema.....	75
	Loading the eXadas Catalogs.....	78
	Accessing the IDMS data with SQL (locally) .....	80
	Accessing the IDMS Data With SQL (remotely) .....	81
<b>Chapter 9</b>	<b>Bringing IMS On Line.....</b>	<b>83</b>
	<b>Introduction to Bringing IMS On Line With eXadas .....</b>	<b>83</b>
	<b>Bringing IMS On Line .....</b>	<b>84</b>
	Mapping the Sample IMS DBD and Copybooks.....	84
	Loading the eXadas Catalogs.....	89
	Establishing the Interface to DBCTL/DRA .....	91
	Accessing the IMS Data With SQL (Locally) .....	92
	Accessing the IMS Data With SQL (Remotely).....	94

---

<b>Chapter 10 Bringing Sequential On Line .....</b>	<b>95</b>
<b>Introduction to Bringing Sequential On Line With eXadas .....</b>	<b>95</b>
<b>Bringing Sequential On Line.....</b>	<b>96</b>
Mapping the Sample Sequential Copybook.....	96
Loading the eXadas Catalogs.....	100
Accessing the Sequential Data With SQL (Locally) .....	102
Accessing Sequential Data With SQL (Remotely).....	103
<b>Chapter 11 Bringing VSAM On Line .....</b>	<b>105</b>
<b>Introduction to Bringing VSAM On Line with eXadas .....</b>	<b>105</b>
<b>Bringing VSAM On Line.....</b>	<b>106</b>
Mapping the Sample VSAM Copybook .....	106
Loading the eXadas Catalogs.....	110
Accessing the VSAM Data With SQL (Locally).....	112
Accessing the VSAM Data With SQL (Remotely) .....	113
Accessing VSAM Data Through CICS .....	114
VTAM Resource Definitions.....	114
CICS Resource Definitions .....	115
Creating a CICS VSAM Table .....	116
Accessing the CICS VSAM Data with SQL (Locally).....	117
Accessing the CICS VSAM Data with SQL (Remotely) .....	118

# Preface

## Introduction

This preface provides information on the notations used in this guide.

## Notations Used in This Guide

The following notations are used in this book:

- New terms are shown in **bold** and parameters are shown in ALL CAPS.
- `Computer font` is used to display text to enter, examples, and samples of code.
- When square brackets [ ] enclose a portion of a format, that portion is optional.
- When braces { } enclose a portion of a format, one of the options within the braces must be selected. When only one possibility is shown, the purpose of the braces is to delimit that portion of the format to which a following ellipsis applies.
- An ellipsis ... means that the portion of the format enclosed by the immediately preceding pair of braces or brackets may be repeated.
- A vertical bar | separating two words indicates that either word can be used.
- All other symbols, such as parentheses, are mandatory and must appear as specified.

# **WARNING:** When editing members of SCACSAMP and SCACCONF,

make sure that line numbers are not automatically inserted. (Under ISPF, set

NUM OFF in the edit profile.)



# Overview

## Introduction to eXadas

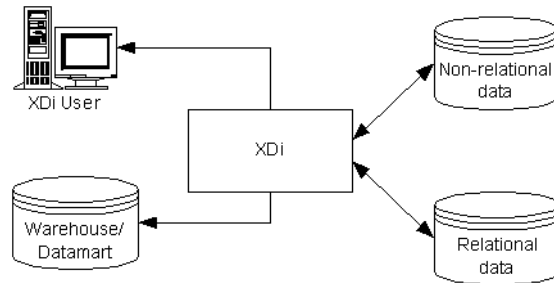
This chapter provides an overview of the eXadas Data Integrator™ and includes the following topics:

- [“Product Overview,”](#) on page 2,
- [“Operational Components,”](#) on page 3,
- [“Application Components,”](#) on page 8, and
- [“Administrative Components,”](#) on page 9.

# Product Overview

eXadas is a powerful, efficient, and easy-to-implement eData solution. End-users and developers can transparently access distributed, mission-critical information using the desktop and Internet tools and applications of their choice. The following figure demonstrates how eXadas accesses relational and non-relational data.

**Figure 1: Accessing Data with eXadas**



eXadas is a complete, high-powered, eData solution that delivers:

- SQL access to relational and legacy data;
- a scalable, high-performance, easy-to-use product;
- a standards-based solution introducing no new application interfaces (APIs); and
- a modular solution that integrates easily with existing environments.

The eData Engine contains the following major components:

- Server,
- Enterprise Server,
- Client Interface Module,
- C Precompiler,
- ODBC and JDBC Connectors, and
- DataMapper.

These components are grouped into three functional areas:

- Operational, which process requests for data and deliver the results to the client tool or application.
- Administrative, which configure components and manage system data.
- Application-enabling, which provide a 3GL hook into the eData Engine.

This chapter provides an overview and summary-level description of these components.

---

# Operational Components

Operational components provide the processing required to connect tools and applications with data. They are responsible for:

- accepting and validating SQL from the application or tool,
- communicating between the end-user and data source platforms,
- accessing the appropriate data source(s), and
- converting results into a consistent relational format.

The Operational components include:

- Server,
- eXadas Enterprise Server,
- Client Interface Module, and
- Client Connectors (ODBC and JDBC).

These modules are discussed in the sections that follow.

## eXadas Server

All data access is performed by platform-specific eXadas Servers. A Server is responsible for the following functions:

- Accepting SQL queries from clients.
- Determining the type of data to be accessed.
- Rewriting the SQL query into the native file or database access language needed. For non-relational data sources, a single SQL access could translate into multiple native data requests. For relational data sources, the SQL provided is translated into the specific SQL of the Relational Database Management System (RDBMS) to be accessed.
- Query optimization based on generic SQL query rewrite and file or database specific optimization.
- Querying multiple data sources for JOINS.
- JOIN optimization based on index statistics held in the eXadas Meta Data Catalog.
- Translating result sets into a consistent relational format. For non-relational data sources this involves restructuring data into columns and rows. For relational data sources, result sets are translated into a single relational format.
- Post Query processing of result sets, such as ORDER BY sorting.
- Issue all client catalog queries to the appropriate RDBMS or eXadas Meta Data Catalog.

A Server accepts connection requests from client applications. Client applications can access a Server using either a CrossAccess-supplied Connector or applications developed using the eXadas C precompiler. Precompiler-developed applications can either reside on the same platform that the Server is executed on or on a remote platform.

There are five types of tasks that can run in the Server:

- Region Controller, which includes an MTO Operator Interface,
- Connection Handlers,
- Query Processors,
- Logger, and
- Initialization Services.

These tasks are described in the sections that follow.

## Region Controller

The Server has multiple tasks running within it. The main task is the Region Controller. The Region Controller is responsible for starting, stopping, and monitoring the other tasks running within the Server. The Region Controller determines which tasks to start based on configuration parameter settings. See [Appendix A, “Configuration Parameters”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information on configuration parameters.

The Region Controller also supplies an OS/390 MTO (Master Terminal Operator) interface that can be used to monitor and control a Server address space.

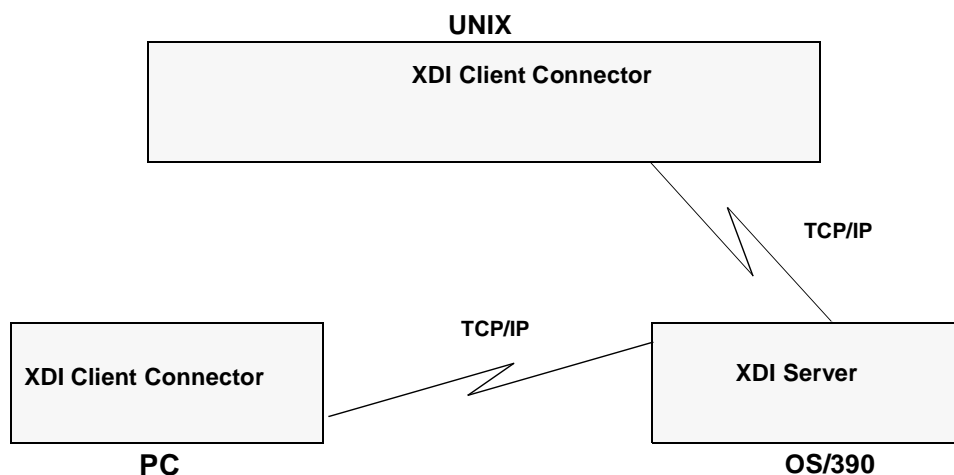
## Connection Handlers

A Connection Handler (CH) task is responsible for listening for connection requests from client applications and routing them to the appropriate Query Processor task.

eXadas contains four typical transport layer modules that can be loaded by the Connection Handler task:

- TCP/IP,
- Cross Memory Services, and
- MQ Series.

A local OS/390 client application can connect to a Server using any of these methods (the recommended approach is to use OS/390 Cross Memory Services). Remote client applications (running under Windows, a UNIX platform, or a different OS/390 image) use TCP/IP to communicate with a remote Server.

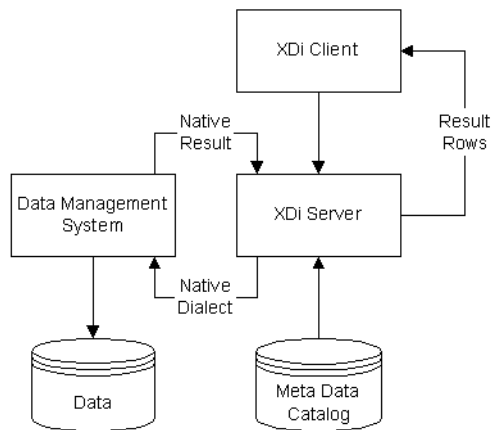
**Figure 2: Sample Communication Implementation**

### Query Processor

The Query Processor is the subcomponent of the Server that is responsible for translating client SQL into database and file-specific data access requests. The Query Processor uses native database and file facilities that maintain both the structural integrity and the performance characteristics of the data source. [Figure 3: “Operational Processing Flow,” on page 6](#), shows the operational processing flow.

The Query Processor treats the different database/file systems as a single data source and is capable of processing SQL statements that access either a single type of database/file system or reference multiple types of databases/file systems. The Query Processor also supports SQL update operations (DELETE, INSERT, and UPDATE).

To process SQL data access requests, data definitions must be mapped to logical tables. The eXadas DataMapper tool is used in conjunction with the eXadas Meta Data Utility to perform this mapping. This information is stored in eXadas-generated Meta Data Catalogs, which emulate DB2 system catalogs. See [“DataMapper,” on page 9](#), for more information.

**Figure 3: Operational Processing Flow**

## Logger

A single Logger task can be running within a Server. The Logger reports on Server activities and is also used in error diagnosis situations.

## Initialization Services

Initialization Services are special tasks used to prepare the Server execution environment to access relational and non-relational data, initialize high level language environments for use by exits, or allow the Server to use the OS/390 Work Load Manager (WLM) services to process queries in WLM goal mode. Currently, the following Initialization Services are supplied:

- IMS BMP/DBB/DLI Initialization Service is used to initialize the IMS Region Controller to access IMS data using an ASMTDLI interface.
- IMS DRA Initialization Service is used to initialize the eXadas DRA interface and to connect to an IMS DBCTL region to access IMS data using the DRA interface.
- Datacom Initialization Service is used to initialize and acquire CA-Datacom/DB task areas.
- CAF Initialization Service is used to connect to a DB2 subsystem to access/update DB2 data using the DB2 Call Attach Facility.
- VSAM Initialization Service is used to improve VSAM performance by sharing open files between users.
- WLM Initialization Service is used to initialize and register with the OS/390 Work Load Manager subsystem (using the WLM System Exit). This allows individual queries to be processed in WLM goal mode.
- The Language environment Initialization Service is used to initialize IBM's Language Environment or COBOL II, which allows exits to be written in a high-level language.

---

## eXadas Enterprise Server

The Enterprise Server can be used to manage a large number of concurrent users across multiple data sources. An Enterprise Server contains the same tasks that a Server uses, with the exception of the Query Processor and the Initialization Services.

Like a Server, the Enterprise Server's Connection Handler is responsible for listening for client connection requests. However, when a connection request is received, the Enterprise Server does not forward the request to a Query Processor task for processing. Instead, the connection request is forwarded to a Data Source Handler (DSH) and then to a Server for processing. The Enterprise Server maintains the end-to-end connection between the client application and the target Server. It is responsible for sending messages to and receiving messages from the client application and the Server.

The Enterprise Server is also used to perform load balancing. Using configuration parameters, the Enterprise Server determines the locations of the Servers that it will be communicating with and whether those Servers are running on the same platform as the Enterprise Server.

The Enterprise Server can automatically start a local Server if there are no instances active. It can also start additional instances of a local Server when the currently active instances have reached the maximum number of concurrent users they can service, or the currently active instances are all busy.

## Client Interface Module

The Client Interface Module is CrossAccess-supplied code that is linked with a user-written client program to establish and maintain connections with Servers and Enterprise Servers. The Client Interface Module performs the following functions:

- Determining and loading the appropriate transport layer module, based on configuration.
- Establishing communications with a Server or Enterprise Server.
- De-referencing host variables in SQL statements.
- Storing and retrieving data in the application storage area(s).
- Presenting error and feedback information to the application.

The Client Interface Module can establish multiple connections to a Server or Enterprise Server(s) on behalf of a single application program.

## Client Connectors

Desktop tools and applications can issue SQL data access requests to an XDi Server through an XDi ODBC or JDBC Connector.

The eXadas ODBC and JDBC clients provide a single interface between end-user tools, applications, and other eXadas operational components. High-speed performance and application integrity are provided by the 32-bit thread safe ODBC and JDBC and Connectors. A single client can access all data sources on all platforms.

The eXadas Connector serves as both an ODBC or JDBC driver and a Connection Handler to other platforms, leveraging the underlying TCP/IP or MQ Series communications backbone.

Five software components interact to enable client data access using eXadas:

- A platform-specific ODBC Driver Manager that loads Connectors on behalf of an application. This component is delivered with the operating system for all Windows platforms (for ODBC only).
- The eXadas ODBC and JDBC Connector that processes function calls, submits SQL requests to a specific data source, and returns results to the application.
- Data source definitions that consist of the name and location of the data the user wants to access. The required data source definitions consist of a data source name and communications parameters (TCP/IP or MQ Series, ). The data source name is used to identify a specific Server or Enterprise Server that will be used to service data access requests.
- The Client Interface Module that is used to bridge from the eXadas ODBC or JDBC Connector to the Query Processor task running in a Server.
- The eXadas Connection Handler that is used to communicate with a Server or Enterprise Server. eXadas supplies a Connection Handler that supports TCP/IP implementations that use a Winsock interface and MQ Series.

For more information on the eXadas Connectors, see the *eXadas Data Integrator OS/390 Connectors Guide*.

## Application Components

Application-enabling components provide developers with a means of using eXadas's data delivery capabilities within 3GL applications. The eXadas C precompiler enables applications using embedded SQL to access heterogeneous data sources that span platforms using a single, standard API.



---

An application is written as if all data being accessed is in a single relational database. The eXadas precompilers convert SQL into calls to eXadas components. Multiple heterogeneous data sources that reside on local or remote platforms can be accessed by the application without regard to location or the type of file or database to be accessed.

## Administrative Components

Administrative Components are tools and utilities used to perform the housekeeping and data administration required to define an installation's environment and to define the data to be accessed by eXadas. The eXadas DataMapper is one of these administrative components. It is discussed in the section that follows.

### DataMapper

The eXadas DataMapper is a Microsoft Windows-based application that automates many of the tasks required to create logical table definitions for non-relational data structures. The objective is to view a single file or portion of a file as one or more relational tables. The mapping must be accomplished while maintaining the structural integrity of the underlying database or file.

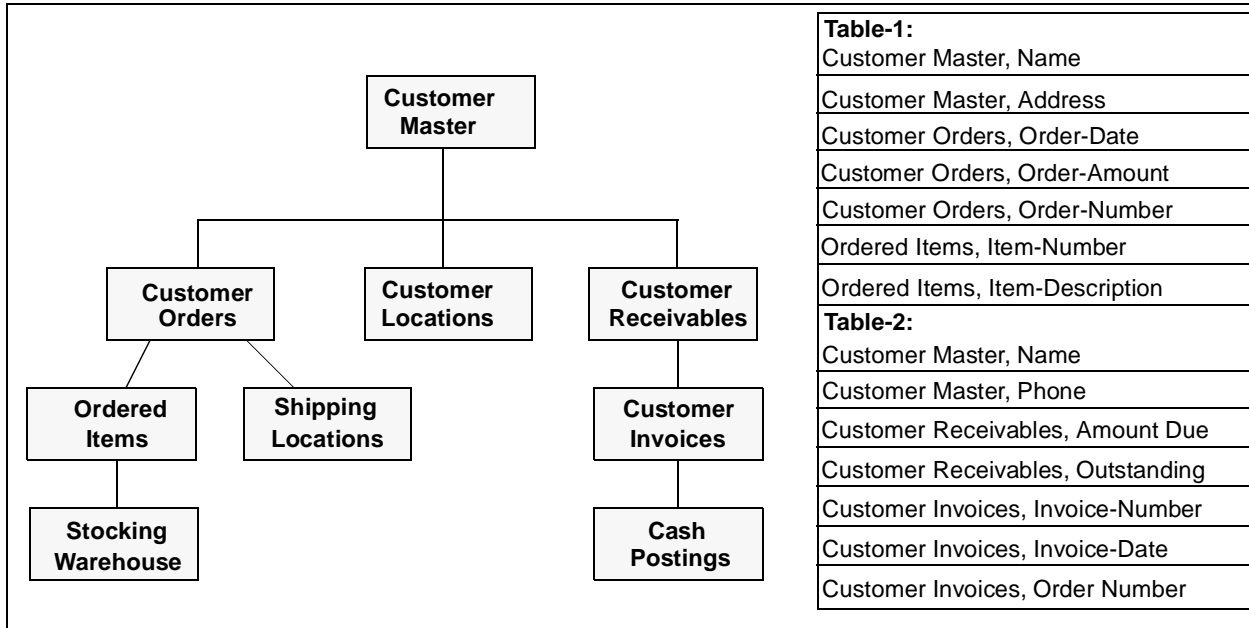
The DataMapper interprets existing physical data definitions that define both the content and the structure of non-relational data. The tool is designed to minimize administrative work, using a definition-by-default approach.

The DataMapper accomplishes the creation of logical table definitions for non-relational data structures by creating Meta Data Grammar from existing non-relational data definitions (COBOL copybooks). The Meta Data Grammar is used as input to the eXadas Meta Data Utility to create a Meta Data Catalog that defines how the non-relational data structure is mapped to an equivalent logical table. The Meta Data Catalogs are used by Query Processor tasks to facilitate both the access and translation of the data from the non-relational data structure into relational result sets.

The DataMapper import utilities create initial logical tables from COBOL copybooks. You refine these initial logical tables in a graphical environment to match site- and user-specific requirements. You can utilize the initial table definitions automatically created by DataMapper, or customize those definitions as needed.

A sample mapping of the Fields within the Segments of a hierarchical IMS database to two logical tables are shown in [Figure 4: "Sample Mapping from Hierarchical IMS DB to Logical Table."](#)

**Figure 4: Sample Mapping from Hierarchical IMS DB to Logical Table**



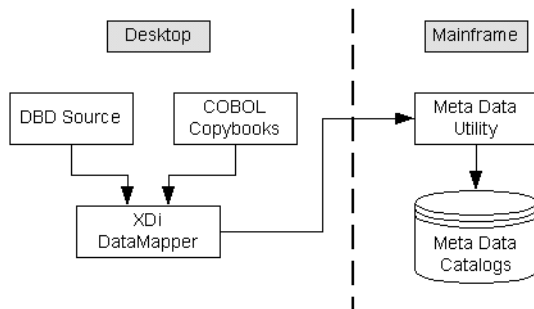
Multiple logical tables can be created that map to a single physical file or database. For example, a site may choose to create multiple table definitions that all map to an employee VSAM file:

- One table is used by department managers who need access to information about the employees in their departments,
- another by HR managers who have access to all employee information,
- another by HR clerks who have access to information that is not considered confidential, and
- another by the employees themselves who can query information about their own benefits structure.

Customizing these table definitions to the needs of the user is not only beneficial to the end-user, but recommended.

Figure 5: “DataMapper Workflow” shows the Data Administration workflow with DataMapper.

**Figure 5: DataMapper Workflow**



---

**NOTE:** The DataMapper contains embedded FTP support to facilitate file transfer to and from the mainframe.

The steps in [Figure 5: “DataMapper Workflow”](#) are described as follows:

1. Import existing descriptions of your non-relational data into DataMapper. COBOL copybooks, IMS-DL/I Database Definitions (DBDs), and IDMS schema/subschema can all be imported into the DataMapper.

The DataMapper creates default logical table definitions from the COBOL copybook information. If these default table definitions are suitable for end-users, skip to step 3.

2. Refine or customize the default table definitions as needed by the users. For example, importing the record layout for the VSAM customer master file creates the default `Customer_Table`. Two additional tables can also be created from the original:
  - `Marketing_Customer_Table` containing only those data items required by the marketing department.
  - `Service_Customer_Table` containing only those data items required by support representatives.
3. Export the logical table definitions to the mainframe where the database/file resides. These definitions are then used as input to the Meta Data Utility, which creates the Meta Data Catalogs.

After completing these steps, you are ready to use eXadas operational components with your tools and applications to access your non-relational data.

For step-by-step information on how to use the DataMapper to map non-relational data to logical tables, see the tutorial chapters of the *eXadas DataMapper Guide*.



# 2

## Concepts

### Introduction to Concepts

The previous chapter provided an overview of the major eXadas components and how they interrelate. This chapter focuses on the components that make up the core eXadas system and key concepts as they pertain to these components. Additional information describing how the other platforms interface with the eXadas components is also covered.

The topics covered in this chapter provide an overview of the Server's capabilities and also define many of the concepts and terminology that eXadas employs. All of the topics discussed in this chapter are discussed in detail in the remainder of this guide. You will also find references to other eXadas guides that you may need to review.

This chapter discusses the following topics:

- [“Relational Access to Data,” on page 14](#), provides an overview of how eXadas accesses all data as if the data were contained in a relational database.
- [“Client/Server Architecture,” on page 15](#), provides an overview of how an eXadas Client Interface Module communicates with a Server. Different communication protocols are identified as well as a brief discussion about the use of the Enterprise Server for enterprise deployments of eXadas.

- [“Data Sources,” on page 16](#), provides an overview of the eXadas concept of an ODBC-like data source definition. CrossAccess uses the term **data source** to identify the eXadas Server (and the service within that Server) that responds to your application’s SQL requests.
- [“Non-Relational Data Mapping,” on page 16](#), provides an overview of how eXadas makes non-relational databases/file systems look like a DB2 relational database. Key concepts discussed include how non-relational-to-relational mapping is performed and how common non-relational database/file system features are supported in eXadas.
- [“eXadas Server and Client Components,” on page 18](#), provides an overview of the components comprising a Server and how client applications communicate with that Server.
- [“National Language Support,” on page 23](#), provides an overview of how eXadas supports translation of non-English data. eXadas supports both single-byte and double-byte character set translation.
- [“Configuration Methodology,” on page 24](#), describes how to configure an eXadas system.

## Relational Access to Data

One of the features that makes eXadas such a powerful solution is that, regardless of the database or file system you want to access, your data is accessed consistently just as if it were in a DB2 database. This lets your front-end tools and applications use the power of SQL to retrieve and manipulate data and removes the necessity and complexity of using varied access methods within them.

eXadas has its own set of system catalogs, called Meta Data Catalogs. The DataMapper, a Windows graphical tool, is used to generate the input to the Meta Data Utility. Mapping your data is covered later in this chapter, as well as described in detail in [Chapter 8, “Mapping Data,”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

eXadas supports the DB2 version 4 dialect of SQL. This is also called the SQL-92 standard. eXadas supports a fairly complete SQL implementation (for example, inner joins, outer joins, sub-selects, GROUP BY, HAVING, and scalar functions).

---

# Client/Server Architecture

eXadas uses a client/server architecture. In order to communicate with a Server, your applications need to interface with an eXadas Client Interface Module. This is performed using one of the following methods:

- An eXadas Connector, which contains a Client Interface Module (for UNIX and Windows environments).
- An application developed for OS/390, UNIX, or Windows that uses one of the eXadas precompilers.

The precompiler generates calls to the Client Interface Module, which is bound with your application. In OS/390 the Client Interface Module is link-edited in with your application. On UNIX the Client Interface Module is distributed as a shared library that applications can link to. In Windows environments, the Client Interface Module is supplied as a DLL.

Before running your application you must configure the Client Interface Module. Depending upon the platform, configuration is performed in one of several ways. In a Windows environment, configuration is performed using the ODBC Connector Administrator. In OS/390 or UNIX, configuration is performed using a text file.

Regardless of the configuration method used, you must identify one or more data sources that your application will be accessing. For each data source you must identify the communications protocol (TCP/IP, MQ Series, or LU 6.2) used to communicate with an eXadas Server.

eXadas supports TCP/IP and MQ Series communications protocols for ODBC- or JDBC-based applications or for OS/390, UNIX, and Windows applications. Additionally, for local (OS/390) applications, a Cross Memory Connection Handler Service is also supported that uses Data Spaces.

The following addresses must be supplied during Client Interface Module configuration based on protocol:

- TCP/IP: the IP address and port number of the Server.
  - The IP address can be specified using dot notation or as a host name.
  - The port number can be specified as a number or a service name.
- MQ Series (OS/390 and NT only)
- Cross Memory Services: the name of a Cross Memory data space and queue name must be specified.

These are automatically created by the Server and require no definitions to any other subsystems.

If your application supports a large number of concurrent users (more than can be handled by a single Server), then the Enterprise Server is available to manage these situations. The Enterprise Server sits between your application and the eXadas Server and appears as the Server to the client. The Enterprise Server is

responsible for starting additional Servers as the number of concurrent users increases. For information about the Enterprise Server, see [Chapter 17](#), “Enterprise Server,” in the *eXadas Data Integrator OS/390 Reference Guide*.

## Data Sources

In eXadas a data source is similar to an ODBC data source. CrossAccess uses the term **data source** to identify a Query Processor that runs in an eXadas Server.

The Query Processor is database/file system neutral. You can map a data source to a particular type of database/file system or to multiple databases/file systems. CrossAccess recommends that you define data sources in organizational terms based on the data that your end users need to access, and not based on the underlying database/file system.

For example, if you have a Credit department that require access to IMS and VSAM data, you can create a single data source, CREDIT. In this example, you would define the logical tables that reference the desired IMS and VSAM credit data in a set of Meta Data Catalogs that are accessible to the Server with a service defined for the CREDIT data source. Logical tables are discussed in the next topic and are described in [Chapter 8](#), “Mapping Data,” in the *eXadas Data Integrator OS/390 Reference Guide*.

In the credit example, when your application connects to the CREDIT data source it has access to all the credit related data as if the data were contained in a single DB2 database. This allows your application to perform heterogeneous joins for any of these tables even though the data is physically stored in IMS databases, and VSAM files.

An eXadas Server can support multiple data sources running different types of Query Processors. For example, in addition to your credit application, you could have another application that needs to access accounting information. In this case, you would create the logical tables that contain the desired accounting information in the same Meta Data Catalogs that contain your credit tables. You would then define another data source/service, ACCOUNTING, for the Server.

## Non-Relational Data Mapping

For non-relational data, like that in IMS and VSAM databases, you must map the non-relational data into a relational representation of the data that eXadas can use. This representation is referred to as a **logical table**.



You can map multiple logical tables against a single physical database/file system. A typical situation in which you would perform this operation is when there is a VSAM file containing 10 different types of records. In this case, you would define 10 different logical tables, one for each type using “views,” each with its own record type.

When accessing IMS data, eXadas only accesses a single hierarchical path in the database as a logical table. If your IMS databases contains multiple child segments, in different hierarchical paths, then you must define logical tables for each hierarchical path that you want to access. Once this is done you can use JOINS to retrieve all of the data you need from different hierarchical paths in a single query.

Logical table definitions are stored in an eXadas Meta Data Catalog. The logical tables, their associated column definitions, and index information are defined by Meta Data Grammar. The index information is used to optimize access to the physical database/file system.

The Meta Data Grammar is a text file processed by the Meta Data Utility, which updates a set of Meta Data Catalogs. When the Meta Data Utility is executed, it verifies the syntax and contents of the Meta Data Grammar and verifies that the physical databases/files referenced exist. During this verification process, the Meta Data Utility also collects additional physical information that is used to further optimize access to the physical database/file.

Although the USE Grammar is stored in human-readable format, and can be defined manually, this is a very tedious and error-prone process. CrossAccess supplies the DataMapper to make the definition process easy. The DataMapper is a Windows-based tool that imports COBOL copybooks to build initial logical tables. Using the DataMapper, you can refine these initial definitions to include only the information you need to create multiple logical table definitions from a single physical database/file (as in the VSAM record type example).

The DataMapper generates the proper Meta Data Grammar for each of the logical tables you have defined for a data source. The DataMapper also contains embedded FTP support so that you can easily download the COBOL copybooks for import into the DataMapper. Additionally, you will use the embedded FTP support to move the generated Meta Data Grammar back to OS/390 so it can be run through the Meta Data Utility. A full description of how to perform non-relational mapping is described in [Chapter 8, “Mapping Data”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

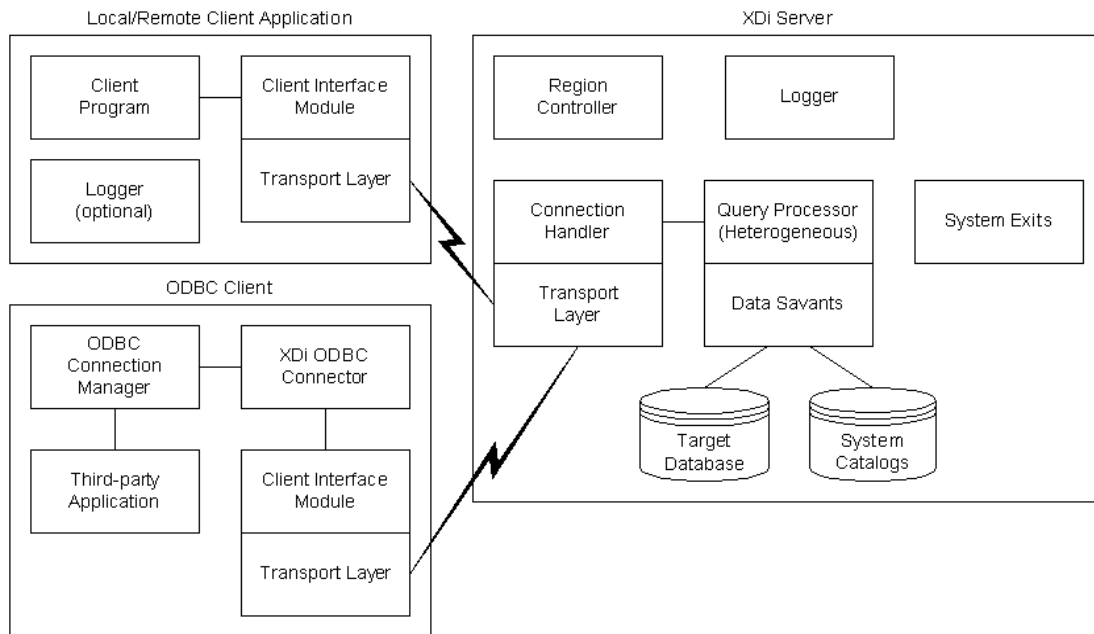
In relational databases, the format and structure of the tables are strictly enforced. For example, a column is defined as one of the data types that is supported by the database. Additionally, relational databases do not support repeating items. Instead you define a separate table that contains multiple rows of data for each repeating data item. These restrictions are typically not enforced by a non-relational database or file system.

eXadas supports the definition of repeating data items in the data mapping process. eXadas logically joins the repeating and non-repeating data and returns a separate row for each instance of the repeating data.

# eXadas Server and Client Components

**Figure 6: “eXadas Architecture”** shows the components that make up a Server and how applications communicate with a Server using eXadas Connectors. It also describes how local and remote client applications and ODBC client applications interface with a Server. Each of the major components and its relationship to other system components is described in more detail in the remainder of this section.

**Figure 6: eXadas Architecture**



## Server Components

The Server consists of several components. All of these are either directly or indirectly activated by the Region Controller based on configuration parameters defined in a master configuration member. These different components are referred to as **services** and are defined using the SERVICE INFO ENTRY parameter.

The **Region Controller** is responsible for starting, stopping, and monitoring the different services that are running within the Server. The services are implemented as individual load modules running as separate OS/390 tasks within the Server address space. Most of the Services can have multiple instances and most can support multiple users.

A description of the different types of services that the Region Controller manages follows.

---

## Initialization Services

Initialization services are special purpose services (tasks) that are used to initialize/terminate different types of interfaces to underlying database management systems or OS/390 system components. For example, an initialization service is provided to activate the DRA interface used by the IMS DRA Data Savant in order to access IMS data. An example of an OS/390 system component initialization service is the Workload Manager (WLM) service.

## Connection Handler Services

The Connection Handler Service task is responsible for accepting connections from your applications. The connection request is routed to the requested Query Processor service for subsequent processing.

## Query Processor Services

The Query Processor is the eXadas relational engine that services user SQL requests.

The Query Processor can service SELECT statements and stored procedure invocations. The Query Processor invokes one or more Data Savants to access the target database/file system referenced in an SQL request. The following Data Savants are supported:

- IMS BMP/DBB interface: allows IMS data to be accessed through an IMS Region Controller. A Region Controller is restricted to a single PSB for the Server, limiting the number of concurrent users the query processor can handle.
- IMS DRA interface: allows IMS data to be accessed using the IMS DRA interface. The DRA interface supports multiple PSBs and is the only way to support a large number of concurrent users. This is the interface CrossAccess recommends.
- Sequential interface: allows access to Sequential files or PDS members.
- Stored procedure interface: allows an OS/390 Assembler, C, COBOL, or PLI application program to be invoked.
- VSAM interface: allows access to VSAM ESDS, KSDS or RRDS files. This interface also supports use of alternate indexes.
- IDMS interface: allows access to IDMS files.
- ADABAS interface: allows access to ADABAS files.
- Datacom interface: allows access to Datacom files.
- DB2 interface: allows access to DB2 files.

## MTO Interface

The MTO interface is an OS/390 Master Terminal Operator interface that allows you to display and control the services and users that are being serviced by a

Server. Using the MTO interface you can also dynamically configure the Server. The MTO interface is contained within the Region Controller service.

## Logger Service

The Logger service is a task that is used for system monitoring and troubleshooting. During normal operations you will not need to be concerned with the Logger Service.

## eXadas Server System Exits

The eXadas Server is a fully multi-threaded implementation designed to service large numbers of concurrent users. A set of system exits for security and accounting purposes is supplied with eXadas.

All system exits are written in Assembler language and are designed to run in a multi-user environment. Source code is provided for all exits so that you can customize the supplied exits to meet your site standards. Complete descriptions about activating the system exits and their APIs can be found in [Chapter 15, “System Exits”](#) in the *eXadas Data Integrator OS/390 Reference Guide*. The following system exits are provided:

- SAF security exit,
- SMF accounting exit,
- CPU Resource Governor exit,
- Workload Manager exit,
- DB2 Thread Management exit, and
- Record Processing exit.

The exits are described in the sections that follow.

### SAF Security Exit

The SAF security exit is provided to perform checks to determine whether a user is authorized to access the Sequential or VSAM data set(s) referenced in a query. For IMS access, the SAF exit determines whether the user is authorized to use the PSB(s) that will be utilized to access the IMS databases referenced in a query. Additionally, the SAF exit is used to verify that a user has authority to execute a stored procedure program. The SAF exit is activated using the SAF EXIT configuration parameter.

### SMF Accounting Exit

The SMF exit is provided to allow you to generate SMF user records that report the CPU time and elapsed time a user was connected to a Query Processor service. The SMF exit is invoked immediately after the user connects to a Query Processor

---

service task instance. Recording ends immediately before final disconnect processing within the Query Processor service task instance. An SMF record for the user is generated when the user disconnects from the Server. The SMF exit is activated using the SMF EXIT configuration parameter

### **CPU Resource Governor Exit**

The CPU Resource Governor exit is used to restrict the amount of CPU time that a user can consume for a unit-of-work. In eXadas, a unit-of-work is considered a single query or series of queries. Typically, the unit-of-work is a single query. However, if your application opens multiple simultaneous cursors for the same data source, then the unit-of-work begins when a PREPARE is issued for the first cursor and ends when all cursors are closed.

When the CPU Resource Governor exit is activated it is passed the available CPU time for that user. Periodically, the CPU Resource Governor exit is called to check how much CPU time has been used. Once the allotted time is exceeded the exit returns a return code that stops the query. The frequency with which the exit is called is controlled by eXadas.

The CPU Resource Governor exit is activated using the CPU GOVERNOR configuration parameter.

In addition to the CPU Resource Governor feature, eXadas also supports resource governing features based on the number of rows fetched and the number of rows staged in the result set. These governors are set using the MAX ROWS RETRIEVED and MAX ROWS EXAMINED configuration parameters. These governors restrict the number of records retrieved by the Data Savants activated to process a query and the number of possible rows that can be returned in a result set. These are somewhat coarse governors and large amounts of CPU can be expended before one of these limits is reached.

### **Work Load Manager Exit**

The Work Load Manger exit allows you to place the queries that a user is running under Work Load Manager (WLM). WLM Goal mode allows the amount of resources consumed by a query to be controlled by the operating system based on site-defined rules. WLM compatibility mode allows you to use RMF monitor reports to examine the usage of resources by a query in comparison to site-defined goals. The Work Load Manager exit is activated using the WLM EXIT configuration parameter.

The Work Load Manager exit uses the same unit-of-work concept that the CPU Resource Governor exit uses. Typically, a unit-of-work is a single query unless your application opens multiple simultaneous cursors, in which case the unit-of-work is from first cursor open to last cursor close.

**NOTE:** The Work Load Manager exit uses enclave TCB support. Enclave TCB support was added to Work Load Manager in OS/390 release 2 version 3 so that you must have this release or higher installed in order to use the Work Load Manager exit.

When active the Work Load Manager exit joins a Work Load Manager enclave when the unit-of-work starts. The enclave is left when the unit-of-work is completed. While in the unit-of-work, the query processor is under Work Load Manager control, if WLM goal mode is active.

### **DB2 Thread Management Exit**

The eXadas DB2 Thread Management/Security Exit modifies the default behavior of connecting to and disconnecting from DB2. In addition, this exit performs SAF calls to validate the user ID of the eXadas client and establishes the correct primary authorization ID for the client in DB2.

### **Record Processing Exit**

The Record Processing Exit, available in the VSAM and sequential data access, is used to modify the characteristics of the record to make it easier for eXadas to process.

## **Client Applications (Precompiler)**

Non-ODBC client programs are run through an eXadas precompiler that embeds code acting as a bridge between the user-written program and the Client Interface Module.

The Client Interface Module is responsible for loading the appropriate transport layer to establish a connection with the target Server(s). The Client Interface Module is capable of supporting more than one connection to more than one data source. When your application connects to a data source, the Connection Handler activates the appropriate Transport Layer Service based on configuration parameters. The Client Interface Module is responsible for shipping all requests to the appropriate Transport Layer Service for the duration of the session (until your application disconnects from a data source).

## **Client Connectors**

A client accesses a Server in a manner similar to the precompiler-based application. The main difference is that the Client Interface Module is embedded in the eXadas Connector. Like a precompiler based application, the Client Interface Module activates the same types of transport layers (based on connection requests) that exist in the Server. The following are the supported eXadas client Connectors:

- ODBC and
- JDBC.

---

# National Language Support

eXadas supports application access to non-English data including Double Byte Character Set (DBCS) data. National Language Support is implemented in both the Client Interface Module and the Query Processor.

For the Client Interface Module, there is a set of configuration parameters that define both the local (where the client is running) and host (the Server) code pages. Two sets of these code page parameters can be specified. One set identifies the code page conversions to be performed on Single Byte Character Set (SBCS) data while the second set identifies the code page conversions for DBCS data. The Client Interface Module configuration parameters include:

- SBCS LOCAL CODEPAGE
- SBCS HOST CODEPAGE
- DBCS LOCAL CODEPAGE
- DBCS HOST CODEPAGE

**NOTE:** SBCS/DBCS translation is only supported for non-OS/390 Client Interface Modules.

For DBCS data eXadas supports the DB2 GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC data types. These data types are expected to contain pure DBCS data. For these data types the DBCS-related code page conversions are performed.

The DB2 CHAR, VARCHAR, and LONG VARCHAR data types can contain SBCS data or a mixture of SBCS and DBCS data (this is referred to as **mixed-mode data**). eXadas inspects the contents of these data types and performs the appropriate conversions using the code page information supplied in the SBCS-related configuration parameters. When manipulating mixed-mode data the Query Processor expects the DBCS data to be delimited with DBCS shift codes.

For most U.S. and European customers, code page conversion parameters are not required. By default, eXadas performs the appropriate ASCII/EBCDIC conversions for you.

The Query Processor also supports special processing for national language support. The majority of the Server is written in SAS/C. The Query Processor uses SAS/C locale support for mixed-mode and DBCS comparison operations. These special comparison functions are activated using the Server LOCALE configuration parameter.

# Configuration Methodology

eXadas configuration varies based on the type of Client Interface Module used and Server types. For example, when you use the eXadas ODBC Connector with an application, configuration is performed with the ODBC Connector Administrator. When configuring precompiler-based Windows, OS/390, or UNIX Client applications, configuration is performed manually, using a text configuration file.

Client configuration is simple and straightforward. You must define the data sources that your application uses. You can define additional tuning and debugging parameters. However, this configuration is usually only performed once per new application deployment.

The eXadas Server is designed for continuous operation. As your use of eXadas expands, the Servers are designed such that you can add new data sources/services without affecting existing applications. You can also perform tuning and troubleshooting without having to stop a Server.

The Servers' configuration files are text files that contain the various configuration parameters defining services and other operational and tuning parameters. These configuration files are stored as members in a configuration PDS. Servers have three classes of configuration members stored in the SCACCONF data set. These (and the sample members provided) are:

- Server configuration (CACDSCF),
- Query Processor configuration (CACQPCF), and
- Administrator configuration (CACADMIN).

These configuration members are described in the sections that follow.

## Server Configuration (CACDSCF)

There is a single configuration member that defines the services that will be running within the eXadas Server address space. Services are defined using the SERVICE INFO ENTRY configuration parameter. This configuration member also contains other configuration parameters that affect all of the services running within a Server.



---

## Query Processor Configuration (CACQPCF)

Each Query Processor service defined in the Server configuration member can name a service level configuration member containing Query Processor-specific definitions.

## Administrator Configuration (CACADMIN)

The Query Processor also allows configuration parameter values to be overridden at an individual user ID level. User configuration overrides are activated using the USER CONFIG parameter, which must be specified in either the Server or the Query Processor configuration member.

When user configuration overrides are activated, the user connects to the Server and a Query Processor service task is selected to service that user. The configuration PDS is accessed using the user ID for that user as the configuration member name. If a member name exists, then the configuration definitions found in that member override applicable definitions that exist in the Query Processor configuration member.

**NOTE:** Typically, you will only use the user configuration override feature when you are developing an application, for tuning, or for troubleshooting and should be used with caution. For normal production operations the configuration parameters used to control the Query Processor should be defined at the Query Processor configuration member level.

## Methods for Updating Configuration Members

You can update the Server configuration members manually, or dynamically using the OS/390 MTO interface. When you update the Server configuration member manually, you must restart the Server for the updates to take affect.

When updating Query Processor configuration members the associated service must be stopped and then restarted for the updates to take affect. Manual updates to an Administrator configuration member take affect when a user connects to the Server and a Query Processor is activated.

An overview of how to perform dynamic configuration using the MTO Operator interface is discussed in [Chapter 10, “Server Operations”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.



# 3

## System Requirements

### Introduction to System Requirements

This chapter describes the resources required to install and execute the eXadas Server. System requirements include:

- [“Minimum Disk Space Requirements for Installation,”](#) on page 28, describes the disk space requirements needed to install eXadas.
- [“Operational Disk Space Requirements,”](#) on page 29, describes the disk space requirements to operate a Server.
- [“Minimum Release Level Requirements,”](#) on page 30, discusses the different system program product release level requirements.
- [“APF Authorization Requirements,”](#) on page 30, describes the APF authorization requirements.
- [“Run-time Memory Storage Requirements,”](#) on page 31, describes the estimated run-time memory storage requirements.

# Minimum Disk Space Requirements for Installation

In order to install a Server approximately 68 cylinders of disk space are required. During installation the libraries referenced in this section are created regardless of the components being installed. All disk space amounts are for 3390 storage units.

The Server is installed using the IBM SMP/E product. The SMP/E product is used to apply, monitor, and ensure the integrity of service updates to the products installed under its control. SMP/E installs a “target” set of runtime libraries as well as a “distribution” set of backup libraries. The target library DD names as well as the low-level qualifiers of their data set names all begin with the letter “S,” the distribution libraries with the letter “A.” Both sets of libraries and the disk requirements of each are shown in the following table.

**Table 1: Disk Space Requirements**

Component	Contents	Disk Space Requirements
SCACLOAD	APF-authorized load library	15 cylinders
SCACCONF	Configuration parameters	1 cylinder
SCACSAMP	Sample JCL	2 cylinder
SCACSASC	SAS/C transient library	6 cylinders
SCACMAC	Macro library for use with SYSTEM exits and client applications	1 cylinder
SCACMENU	System Messages Catalog	4 tracks
ACACLOAD	SMP/E “Distribution” copy of SCACLOAD	15 cylinders
ACACCONF	SMP/E “Distribution” copy of SCACCONF	1 cylinder
ACACSAMP	SMP/E “Distribution” copy of SCACSAMP	2 cylinder
ACACSASC	SMP/E “Distribution” copy of SCACSASC	6 cylinders
ACACMAC	SMP/E “Distribution” copy of SCACMAC	1 cylinder
ACACMENU	SMP/E “Distribution” copy of SCACMENU	4 tracks

---

# Operational Disk Space Requirements

In addition to the installation libraries, if you are running the Servers as started tasks, you need to place the Server JCL procedure in a procedure library that is accessed when a start command is issued to activate a Server. Because you already have these procedure libraries allocated and the size of the Server procedures are relatively small, no additional disk space should be required for this aspect of implementing eXadas.

The only other set of permanent data sets that you need are for the Meta Data Catalogs. The disk space required for the Meta Data Catalogs depends on the number of non-relational-to-relational mappings that you have defined. Additionally, you will need to decide whether to create new sets of the Meta Data Catalogs for each application that you create or whether all of, or a group of, related applications will share the same Meta Data Catalogs.

Disk space requirements for the Meta Data Catalogs are minor. The Meta Data Catalogs consist of index and data components. Average Meta Data Catalogs contain approximately 100 logical tables with a total of approximately 2,000 columns. The disk space requirements for the index component is one cylinder while the data component is three cylinders. You can safely assume that approximately five cylinders of disk space are required for each set of Meta Data Catalogs that you create.

To summarize, you will need about 62 cylinders of 3390 disk space for the basic operational infrastructure and another 3 to 5 cylinders for each set of Meta Data Catalogs that you implement. For planning purposes assume you will need approximately 71 cylinders of permanent disk space to operate eXadas.

Another disk space requirement to estimate is the temporary disk space required to process user queries. Space is required to store the result set of a query before it is sent to the user. You control the size and location of where the temporary file(s) are written using the LD TEMP SPACE configuration parameter. For performance reasons, it is recommended to use the HIPERSPACE option. A smaller (1M) value should be specified for the initial allocation with a larger EXTEND value specified.

As a rule of thumb, assume that the amount of temporary storage that will be required is twice the size of the total result set. For example, if the result set is 3 megabytes then approximately 6 megabytes of temporary storage is required to stage the result set output.

The total amount of temporary DASD that is required is based on the types of queries and the number of concurrent users. For example, if you have 50 concurrent users all issuing queries that retrieve 3 megabytes of data and the queries contain an ORDER BY clause then 150 megabytes of temporary DASD is required to service all 50 users.

# Minimum Release Level Requirements

The following table identifies minimum release level requirements for some of the different components that make up a Server.

**NOTE:** The release levels listed in the following table are minimum release levels required for eXadas. More recent release levels are also valid.

**Table 2: Minimum Release Level Requirements**

System Component	Minimum Release Level
Base System	OS/390 Release 2.6 DFSMS Release 1.4
VSAM Data Savant	DFP/VSAM version 2.3
IMS DBB/BMP Data Savant	IMS version 3.1
IMS DRA Data Savant	IMS version 5.1
IDMS Data Savant	IDMS version 14.1
DB2 Data Savant	DB2/390 version 4.1
Datacom Data Savant	Datacom version 9.0
ADABAS Data Savant	ADABAS level 5.3, or higher. Version 6.21 requires fix AN21035.
TCP/IP	IBM TCP/IP version 3.2.
VTAM LU6.2	VTAM version 4.5. For CICS Stored Procedures implementation, VTAM version 4.1.
Cross Memory	OS/390 release 2.6
SAS/C transient library	SAS/C version 6.0 with IBM TCP/IP version 3.2.
C Precompiler	SAS/C compiler with IBM TCP/IP 3.2 support service applied to object libraries

## APF Authorization Requirements

Local client applications do not require the eXadas load library to be APF-authorized. However, the Server and Enterprise Servers require APF authorization for this library in most typical installations, as does the configuration used in the Installation Verification Procedure. After the system installation has been verified, APF authorization can be removed if your configuration does not include any use

of the Cross Memory transport layers, Hiperspace defined in the LD TEMP SPACE parameter, or any of the Security, Accounting, or Work Load Manager System Exits. The SAS/C transient library must be APF-authorized.

## Run-time Memory Storage Requirements

The amount of virtual storage required to operate a Server varies based upon:

- the number and types of components configured,
- how each component is configured,
- the number of concurrent users being serviced, and
- the complexity of the queries being run by each user.

### Storage Requirements for Client Applications

The following table shows the typical storage requirements for an OS/390 client application.

**Table 3: Storage Requirements for Client Applications**

Component	Minimum Storage
Client Interface Module	251K
Cross Memory Transport Layer load module	58K*
TCP/IP Transport Layer load module	52K*
Message pool	400K
SAS/C transient modules	200K
TOTAL	909K (XM Transport Layer connected to a single Server)

\* The number of Transport Layer modules actually loaded depends on the number of protocols configured and the number of application/Server connections.

## Storage Requirements for a Server

Additional storage is also required for the user-written program portion of the client application.

[Table 4, “Minimum Storage Requirements for a Fully-Configured Server”](#) identifies the minimum virtual storage requirements for a fully-configured Server using all of the protocols. Virtual storage requirements by a Server largely depends on the number of users being serviced and the complexity of the queries being issued by each user.

**Table 4: Minimum Storage Requirements for a Fully-Configured Server**

Component	Minimum Storage
Region controller load module	267K
Connection handler load module	250K*
Cross Memory Transport Layer load module	58K*
TCP/IP transport layer load module	52K*
Query Processor load module	98K
Datacom Data Savant	291K
DB2 Data Savant	291K
IMS Data Savant load module	291K
IDMS Data Savant load module	800K
Sequential Data Savant load module	229K
VSAM Data Savant load module	253K
ADABAS Data Savant load module	291K
Message pool	4,000K (user configurable)
Logger load module	241K
WLM initialization load module	242K
IMS interface load module (front end)	240K**
IMS interface load module (back end)	241K**
IMS/DRA interface load module	248K**
eXadas Enterprise Server data source handler	250K
TOTAL	7601K (with DBB/BMP Interface and single client connection)



\* The number of Transport Layer modules actually loaded depends on the number of protocols configured.

\*\* Only one IMS Interface type (DBB/DLI, BMP, or DRA) can be loaded by a single Server.

## Temporary Storage Requirements Per User

This section describes the estimated temporary storage requirements for internal control blocks and data buffers.

When a user connects to a Server, additional storage allocations are made on behalf of the user. Additional storage is also temporarily made to process an SQL query. The following table identifies the type and size of the storage requests that would be made to process a simple single table query.

**Table 5: Temporary Storage Requirements for Users**

Component	Minimum Storage
User session control blocks	100K *
Query processor control blocks	200K
Result set control blocks	100K
File buffer cache	100K **
TOTAL	500K

\* The amount of storage allocated for user session control blocks is dependent upon the size of the messages that are sent between the client and Server. This is regulated by a configurable parameter.

\*\* The amount of storage allocated for file buffer caching is a user configurable parameter. The default value is 100K. File caching storage is allocated on demand. For many queries, file buffer caches are not necessary.

## Fifty Concurrent User Storage Requirements

eXadas has many configuration parameters that can be used to control the amount of virtual storage that is allocated to service a user. Some of these parameters explicitly affect the amount of storage required (an example is the configuration parameter used to define the file buffer cache size). Other parameters are applied

to each query. For example, if a query is a candidate for immediate return of data, then file cache buffer space is not needed to store a temporary result set.

[Table 6, “Storage Requirements for 50 Concurrent Users”](#) identifies the storage requirements when the Server is servicing 50 concurrent users issuing simple queries.

**Table 6: Storage Requirements for 50 Concurrent Users**

<b>Component</b>	<b>Minimum Storage</b>
Base Server	7,019K
Fifty users	25,000K
TOTAL	32019K

# 4

## OS/390 Installation

### Introduction to Installation

This chapter describes how to install the Server on your OS/390 system and contains the following topics:

- [“Installing a New Server,”](#) on page 37, and
- [“Installing Additional Components,”](#) on page 42.

Installation is expected to be performed by a systems programmer. During the installation process, the basic libraries required to run eXadas are created and populated. In addition to supplying the load modules required to run eXadas, sample JCL, configuration files, sample application programs, and sample system exits are loaded onto your system.

**NOTE:** All sample JCL, configuration files, and sample application programs referenced in this chapter are in the SCACSAMP and SCACCONF libraries included with your eXadas software shipment.

The sample JCL and configuration files require customization for your site. These modifications should be performed by a systems programmer. This chapter describes how to install the eXadas Server infrastructure. [Chapter 1, “Overview,”](#) [Chapter 2, “Concepts,”](#) and [Chapter 3, “System Requirements”](#) are prerequisites to the installation process.

The customization and configuration processes required to verify this installation using an eXadas sample application are described in the chapters that follow.

**NOTE:** CrossAccess strongly recommends reviewing [Chapter 1, “Overview,”](#) [Chapter 2, “Concepts,”](#) and [Chapter 3, “System Requirements”](#) before installing eXadas.

The Server components are installed using IBM’s SMP/E. This section presumes that the individual installing eXadas is familiar with SMP/E installation procedures and terminology.

Upon successful installation, the components you ordered are populated to the libraries shown in [Table 7, “SMP/E Libraries,”](#) as well as the required elements to verify their proper installation.

**NOTE:** Values included in the following table are low-level qualifiers for the distribution and target libraries.

**Table 7: SMP/E Libraries**

Target	Distribution	Library Contents
SCACLOAD	ACACLOAD	APF-authorized load modules
SCACCONF	ACACCONF	Configuration members
SCACSAMP	ACACSAMP	Sample members
SCACSASC	ACACSASC	SAS/C transient library
SCACMAC	ACACMAC	Macro library for sample System Exits

There are two installation options:

- [“Installing a New Server,”](#) on [page 37](#), describes the steps required to install this version of the Server for the first time.
- [“Installing Additional Components,”](#) on [page 42](#), describes the steps required to install additional components to an existing Server.

# Installing a New Server

This section describes how to complete a new installation of eXadas. Complete these steps only if you are installing the Server for the first time or if you are installing a new version or release of eXadas. If you are adding or replacing components in an existing Server, follow the installation procedure in [“Installing Additional Components,” on page 42](#).

**NOTE:** The following steps install the entire XDi product on a single disk unit and volume. If your environment is configured differently, then you must customize the JCL to be performed starting at [step 4](#).

## To install an OS/390 Server and its associated components:

1. Unload sample jobs needed for product installation.

Use JCL similar to the example at the end of this step, with the following changes:

- Replace **\*\*TVOL\*\*** with the VOLSER (volume serial number) of the distribution tape. *Not applicable for Internet or FTP distribution.*
- Replace **\*\*TUNIT\*\*** with the tape drive unit type, such as 560 or 570. *Not applicable for Internet or FTP distribution.*
- Replace **\*\*DUNIT\*\*** with the device name for DASD storage.
- Replace **\*\*HILV\*\*** with the high-level qualifier(s) for the eXadas data sets.
- Replace **\*\*VRM\*\*** with the version/release/modification level for this eXadas version, release, and modification level. The form for this qualifier is VvRrMmm.
- Insert a valid job card.

Submit the job for batch execution. This step requires the distribution tape.

When this installation step is complete, the members specified on the SELECT MEMBER= statements have been copied from the distribution media to the library. A copy of job JCL CACXCOPY follows.

```
//CACXCOPY JOB .
//CACXCOPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=**SYSOUT**
//* Tape distribution
//BASE DD UNIT=**TUNIT**,VOL=SER=**TVOL**,
// DSN=CAC224B.F1,
// LABEL=(2,SL,EXPDT=98000),DISP=OLD
//* Internet or FTP distribution
//* BASE DD DSN=CAC224B.F1,DISP=OLD
//INSTALL DD UNIT=**DUNIT**,
// SPACE=(TRK,(5,5,10),RLSE),
// DSN=**HILEV**.**VRM**.INSTALL,
// RECFM=FB,LRECL=80,BLKSIZE=27920,
// DISP=(NEW,CATLG)
```

```
//SYSIN      DD *
COPY INDD=BASE , OUTDD=INSTALL
SELECT MEMBER=CACXCOPY
SELECT MEMBER=CACXPREP
SELECT MEMBER=CACXALLO
SELECT MEMBER=CACXREC
SELECT MEMBER=CACXAPP
SELECT MEMBER=CACXACC
SELECT MEMBER=CACXPOST
SELECT MEMBER=CACXEPRM
SELECT MEMBER=CACXFPRM
SELECT MEMBER=CACXPARM
SELECT MEMBER=CACXREJ
SELECT MEMBER=CACXREST
SELECT MEMBER=CACXDEL
/*
//
```

2. Edit the CACXPARM member in the INSTALL library.

CACXPARM contains installation-specific information required by the JCL tailoring EXEC.

Information in CACXPARM is case-sensitive and must be upper-case. Enter your changes to the right of the equal sign (=). The CACXPARM variables include:

- JOBCARD . 1 is a JOB statement.
- JOBCARD . 2 is a JOB statement.
- JOBCARD . 3 is a JOB statement.
- JOBCARD . 4 is a JOB statement.
- JOBCARD . 5 is a JOB statement.
- JOBCARD . 6 is a JOB statement.
- JOBCARD . 7 is a JOB statement.
- JOBCARD . 8 is a JOB statement.
- JOBNAMES= specifies whether to replace the job name entered in the JOBCARD . 1 field or not. The options are:
  - N: do not replace the job name with the JCL member name. The system uses the job name defined in JOBCARD.1 for all jobs.
  - Y: replace the job name with the JCL member name.
- SYSOUT is the SYSOUT class for all installation job streams.
- DISKVOLU is the disk volume VOLSER where the SMP/E and eXadas data sets are to be allocated.
- DISKUNIT is the disk unit related to DISKVOLU.
- TEMPUNIT is the disk unit to be used for all SMP/E temporary data sets.

- TAPEVOLU is the distribution tape VOLSER. The VOLSER can be found on the packing list and tape label sent with your eXadas shipment. *No change required for internet or FTP distribution.*
  - TAPEUNIT is the generic unit related to TAPEVOLU. *No change required for internet or FTP distribution.*
  - SMPEHILV is the SMP/E data set high-level qualifier(s).
  - CACFHILV is the eXadas data set high-level qualifier(s).
  - CACFVVRM is the eXadas version, release, and modification level, in the form VvRrMmm.
  - TARGET is the eXadas SMP/E target zone name.
  - DISTRIB is the eXadas SMP/E distribution zone name.
3. Run the JCL-tailoring REXX EXEC.

From the command line in ISPF, enter the following command, replacing CACFHILV and CACFVVRM with the values entered in the JCL in [step 1](#).

```
TSO EX `cacfhilv.cacfvvrm.INSTALL(CACXEPRM)'
```

CACXEPRM updates the following members in INSTALL data set with information contained in CACXPARM as entered in [step 2](#):

- CACXPREP,
- CACXALLO,
- CACXREC,
- CACXAPP,
- CACXACC,
- CACXPOST,
- CACXREJ,
- CACXREST, and
- CACXDEL.

The message:

```
***Customization of installation JCL complete***
appears when CACXEPRM completes the update.
```

**WARNING:** CACXEPRM must only be executed once. If an error occurs, delete all data sets and restart the installation at Step 1.

4. Submit member CACXPREP.  
This job allocates and prepares SMP/E control data sets.
5. Submit member CACXALLO.  
This job allocates eXadas data sets.
6. Submit member CACXREC.

This job RECEIVES the eXadas component(s) using SMP/E.

7. Submit member CACXAPP.

This job APPLYS the eXadas component(s) into the target libraries using SMP/E.

8. Submit member CACXACC.

This job ACCEPTs the eXadas component(s) into the distribution libraries using SMP/E.

9. Submit member CACXPOST.

This job allocates VSAM data cluster and Sequential files for deployment of VSAM and Sequential data sources and also populates the error message catalog, SCACMENU. For additional information about deploying VSAM or Sequential data sources, see [Chapter 10, “Bringing Sequential On Line”](#) or [Chapter 11, “Bringing VSAM On Line.”](#)

**NOTE:** All of the batch jobs submitted in steps 4 through 9 must end in a Condition Code 0 for installation to be completed successfully. Any job that does not end in Condition Code 0 requires investigation and resolution before proceeding to the next step. (Bypassing ++HOLD statements, if necessary, during step 7 and step 8 results in a Condition Code 4 for these steps. This is an acceptable code in these situations).

eXadas also supports access and update to IAM files. If you have IAM installed at your site and would prefer to run the VSAM IVP process using IAM instead, then you can modify CACXPOST and alter the DEFINE CLUSTER to define the IVP VSAM cluster as an IAM cluster. Use one of the following methods to convert the VSAM file to an IAM file:

- Add \$IAM to the name of the file.
- Supply an OWNER(\$IAM) clause.
- If you are using SMS managed storage for IAM files, supply a DATACLAS or STORCLAS parameter that contains \$IAM in the name.

10. APF-authorize the SCACLOAD and SCACSASC libraries, if needed.

11. Grant authority to the eXadas data sets.

Grant execute authority to SCACLOAD and SCACSASC, update authority to SCACCONF, SCACSAMP, and SCACMAC, and read authority to all other eXadas data sets.

Minimally, you need to grant this authority to all developers who will be using, installing, and/or maintaining eXadas. You may want to grant universal access of read to these data sets (execute access for SCACLOAD and SCACSASC).



## IBM MQ Series

### To configure the Server to support MQ Series communications from a client:

1. Modify the Server JCL.

You may have to modify the supplied sample JCL for the Server. Any OS/390 component using MQ Series must have access to the following MQ Series libraries:

- *thlqual.SCSQANLx* and
- *thlqual.SCSQAUTH*

where *thlqual* is replaced with the high-level qualifier for your MQ Series installation and *x* is replaced with the language letter to be used (probably E). If these libraries are globally available (in LPA), then no JCL modifications are required. Otherwise the eXadas Server must have the previous two libraries concatenated into the STEPLIB DD statement for the first step in the Server's JCL (EXEC PGM=CACCNTL).

2. Obtain the name of the Local Queue name that the Server will use.

Contact your MQ Series administrator and obtain the names set up for use by eXadas. A different Local Queue name is required for each MQ Series Connection Handler Service that you configure in the Server.

3. Edit the Server master configuration member (SCACCONF member CACDSCF).

The sample master configuration member supplied contains default parameter values for the key configuration parameters that are required to execute a Server. The sample master configuration member also contains a set of SERVICE INFO ENTRIES that are commented out. These SERVICE INFO ENTRIES apply to the eXadas sample applications. Once fully activated, you will be able to access eXadas database/files both locally and remotely.

4. Activate the IBM MQ Series Connection Handler Service

Uncomment the SERVICE INFO ENTRY for the IBM MQ Series Connection Handler Service. This SERVICE INFO ENTRY can be identified by the comments in the master configuration member. The last field in the SERVICE INFO ENTRY (the task data field) parameter specifies the name of the local queue used to accept connections from local and remote client applications. This name, in addition to a model queue name, must also be specified in the local or remote client's configuration file on the DATASOURCE parameter.

For more information on the SERVICE INFO ENTRY parameter, see [Appendix A, "Configuration Parameters,"](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

# Installing Additional Components

This section describes how to install additional components to an existing Server. Follow these steps only if you are upgrading an existing system. If you are installing eXadas for the first time, or installing a new version or release of eXadas, follow the steps in [“Installing a New Server,” on page 37](#).

## To install additional components:

1. Edit members CACXREC, CACXAPP, and CACXACC in the INSTALL library.
2. Increment the SOURCEID specified in each member. For example, if the current SOURCEID is SOURCEID (CAC224B), increment it to SOURCEID(CAC223C).

**WARNING:** When editing SCACSAMP and SCACCONF, members, make sure that line numbers are not automatically inserted by the editor. (Under ISPF, set NUM OFF in the edit profile.)

3. Submit member CACXREC.

This job RECEIVES eXadas components from the distribution tape using SMP/E.

4. Submit member CACXAPP.

This job APPLYS the eXadas components into the target libraries using SMP/E.

5. Submit member CACXACC.

This job ACCEPTs the eXadas components into the distribution libraries using SMP/E.

**NOTE:** All of the batch jobs submitted in steps 2 through 4 must end in Condition Code 0 for installation to be completed successfully. Any job that does not end in Condition Code 0 requires investigation and resolution. (Bypassing ++HOLD statements during step 3 and step 4 results in a Condition Code 4 for these steps. This is an acceptable code in these situations).

# Starting the eXadas Server

This section describes the steps needed to start the Server as a started task.

The process is as follows:

- Copying the Server JCL into your PROCLIB.
- Reviewing the eXadas Server configuration members.

- Configuring the Server for communications.
- Starting the Server.

The process is described in detail in the sections that follow.

## Copying the eXadas Server JCL

The SCACSAMP data set contains sample JCL to start the Server as a started task.

### To copy the eXadas server JCL:

1. Copy the CACDS member from the SCACSAMP data set to your PROCLIB.
2. Customize to run in your environment.

## eXadas Server Configuration Members

The configuration members are stored in the SCACCONF data set and contains Service Info Entries (SIEs) that define the various services. Within each SIE are various fields that define the service, the number of tasks started at eXadas Server start up, the minimum and maximum number of tasks allowed, timeout values, and trace options.

CACDSCF is the eXadas Server configuration file. It contains parameters that affect the entire process. All eXadas service definitions must be included in this member. Each eXadas Server is required to have its own configuration member.

CACQPCF is the Query Processor (QP) service configuration member and is defined by field 10 (task data) of a Query Processor Service Info Entry contained in the CACDSCF member. This member contains configuration parameters that affect a particular QP and its access of data. The intent is to be able to tune different QP services separately. Each QP SIE designates a data source name that is unique within a given eXadas Server. This can be used to direct client requests to different Query Processors, each tuned to satisfy a different class of query.

For more information about the SERVICE INFO ENTRY parameters, see [Appendix A, “Configuration Parameters”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

CrossAccess recommends that after you complete the data source access verification, and before you start to develop your own applications, you review the sample configuration members in order to become familiar with all of the common parameters and their categories (eXadas Server or QP). The syntax and detailed descriptions of all parameters are contained in [Appendix A, “Configuration Parameters”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

There is also a third type of configuration member, the administration override configuration member. This member is intended for use by eXadas administrators during application development and problem diagnosis and is not for use by general end-users. See [Chapter 2, “Deploying Applications”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for additional information.

## Configure the eXadas Server for Communications

This section describes how to set up the Server for TCP/IP communications.

### TCP/IP

**To configure the Server to support TCP/IP communications from a remote client application:**

1. Obtain the name/IP address and the port number or socket number the Server will use.

Contact your network administrator and obtain the IP address of the OS/390 mainframe the Server is running. Also obtain the port number that has been assigned for use by the Server.

2. Edit the Server configuration.

The eXadas configuration member, CACDSCF, in the SCACCONF data set, contains a SERVICE INFO ENTRY for TCP/IP communications. Uncomment the SERVICE INFO ENTRY for the TCP/IP Connection Handler Service. The last two fields in the SERVICE INFO ENTRY parameter need to be modified to specify the IP address of the OS/390 host machine and socket number that is used to accept connections from client applications.

**NOTE:** The port number must not be in use by any other application, and it is recommended to be greater than 5000.

If your OS/390 TCP/IP system is using off-load gateways, ensure that the IP address that is specified reflects the IP address of the OS/390 TCP/IP stack, not the address of an off-load gateway device’s IP stack.

If you are using Interlink TCP/IP then the hostname must be specified as an IP address in dot-decimal notation and must be all zeros (000.000.000.000). See your Interlink documentation for information on configuring a server application.

For more information on the SERVICE INFO ENTRY parameter see [Appendix A, “Configuration Parameters”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

---

## Starting the eXadas Server

Now that we have configured communications, we are ready to start the Server.

### To start the eXadas server:

1. Issue an operator command to start the Server.

From the OS/390 console, issue a standard start command:

```
S CACDS
```

2. Review the initialization messages to confirm successful initialization with your communications environment and that the Server is **READY**. Once you have confirmed a successful start of the Server, you can stop with a standard stop command:

```
P CACDS
```

Proceed to the appropriate chapter for the database you will be bringing online with eXadas:

- [Chapter 5, “Bringing Adabas On Line”](#)
- [Chapter 6, “Bringing DB2 On Line”](#)
- [Chapter 7, “Bringing Datacom On Line”](#)
- [Chapter 8, “Bringing IDMS On Line”](#)
- [Chapter 9, “Bringing IMS On Line”](#)
- [Chapter 10, “Bringing Sequential On Line”](#)
- [Chapter 11, “Bringing VSAM On Line”](#)

**NOTE:** If you are installing more than one database type, you must install each database individually. After configuring the Server for the first database type, bring the Server up. Then, to configure the same Server for another database type, bring the Server down, uncomment the appropriate libraries, and start the Server again.



# 5

## Bringing Adabas On Line

### Introduction to Bringing Adabas On Line With eXadas

This chapter describes the steps necessary to bring Adabas data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the Adabas files are brought on line, you are ready to access the data using standard SQL.

This section describes the process using the sample Adabas file Employees, that is part of a sample database usually created during the installation process of the Adabas product. If this sample Adabas file is not available in your site, you can use one of yours. The steps outlined in this chapter are the same steps used to bring your own Adabas tables online with the eXadas Server.

The process is as follows:

- Run the eXadas utility called the USE Grammar Generator (USG) to extract file definition information about the Adabas file and create logical tables.
- Load the eXadas Catalogs with the logical tables.
- Access the Adabas data with SQL.

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe.

For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

The process is described in detail in the section that follows.

## Bringing Adabas On Line

During the installation of the Adabas product, a sample Adabas file called Employees is usually created under a sample database. It contains 1107 records of employee information. We will bring this file on line with eXadas. If you do not have this file installed on your system, the same process described below can be used to bring your own Adabas files on line with eXadas.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate for your site. This includes concatenating Adabas specific libraries provided by Software AG. Templates of these libraries are commented out in the JCLs. You will need to uncomment them, and provide the appropriate high-level qualifiers.

### Set Up Adabas-Specific Environment

Before the installation can proceed, a module has to be linked into the Adabas module ADAUSER and a new version of ADALNK must be created.

Edit and submit the CACADAL job. this will create and populate the SCACLOAD data set with the CACADABS and ADALNK modules, which is needed for Adabas access.

### Access File Definition Information And Create Logical Tables

The SCACSAMP data set on the mainframe contains a job that extracts information on the layout of the Employees file from Predict and generates the



USE grammar describing the eXadas logical table CAC.EMPLADA. The member name is CACADAUG.

**To create logical tables:**

- Customize the JCL in member CACADAUG to run in your environment and submit.

The value of N (after the keyword SYSDIC) must be the file number of the Predict file on your Adabas system. If the Employees file is defined under a name different from EMPLOYEES-FILE, then you need to change it appropriately.

This will create a member called CACIVGAD.

If you are using the sample employees file installed from Adabas version 7.1 or higher, the Predict view name should be EMPLOYEES. This view is available with Predict 4.11. If you have the new sample employees file, but the EMPLOYEES view is not available, use the EMPLOYEES-FILE view and make the following changes to the output of the CACADAUG job:

- a. The BIRTH field should replace the USE AS DECIMAL(6,0), with USE AS DATE “MMDDYYYY”,
- b. The LEAVE\_START and LEAVE\_END fields should replace the DECIMAL(6,0), with DECIMAL(8,0),

## Loading the eXadas Catalogs

**To load the eXadas catalogs with the table you created in the previous section:**

1. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, this step can be skipped.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the eXadas Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the eXadas Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

2. Edit the member CACADADD to ensure that the Adabas environment information is correct.
3. Load the catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

4. Customize JCL to run in your environment and submit.
  - a. Make sure the symbolic, GRAMMAR=, is pointing to the appropriate USE GRAMMAR member (for example, GRAMMAR=CACIVGAD).
  - b. Uncomment and set the Adabas symbolic, the steplib DDs that point to your Adabas libraries, and the DDCARD DD.
  - c. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs are loaded with the logical tables.

A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

5. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTS required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTS for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the Adabas Data With SQL (Locally)

### To access Adabas data with SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the eXadas Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this parameter. It defines the data source name that is used by the client to connect to the eXadas Server. The name CACSAMP is the data source name that will be defined as an ODBC Data Source. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

```
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second is ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Start eXadas Server and access data from local client.

Before starting the Server you must ensure the Adabas symbolic and DDs have been uncommented and modified for your environment. These include the STEPLIB DD and the DDCARD DD.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for Adabas.

In the SCACCONF data set you will also find a member called CACSQL. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD which we'll talk about next.

**NOTE:** If you are testing for your own Adabas tables, you want to create your own SQL in the same member (CACSQL) or create another member for your own SQL statements.

- b. Configure the client.

The client configuration file is used to communicate to the eXadas Server using the communication protocol defined in the eXadas Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server, described in [“Starting the eXadas Server,” on page 45](#).

- c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the eXadas Server using CACSQL as the input SQL.

- d. Customize the JCL to run in your environment and submit.

- e. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the eXadas Server, which is important if ever you need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the Adabas Data With SQL (Remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, MSAccess, MSExcel, Crystal Reports, IIS, WebSphere, and so on.



# 6

## Bringing DB2 On Line

### Introduction to Bringing DB2 On Line With eXadas

This chapter describes the steps necessary to bring DB2 data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the DB2 files have been brought on line, you are ready to access the data using standard SQL.

This chapter describes the process using the sample DB2 file Employees, which is part of a sample database usually created during the installation process of the DB2 product. If this sample DB2 file is not available at your site, you can use one of your own DB2 files. The following steps are the same steps used to bring your own DB2 tables online with the eXadas Server.

The process is as follows:

- Setting up the interface with DB2.
- Loading the eXadas Catalogs with the logical tables.
- Accessing the DB2 data with SQL

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe.

For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

The process is described in detail in the sections that follows.

## Bringing DB2 On Line

During the installation of the DB2 product, a sample DB2 table called EMP is usually created under a sample database. It contains 42 records of employee information. We will be bringing this table online with eXadas. If you do not have this table installed on your system, the same process described in the following sections can be used to bring your own DB2 tables online with eXadas.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate for your site. This includes concatenating DB2 specific libraries provided by IBM. Templates of these libraries are commented out in the JCLs. You will need to uncomment them, and provide the appropriate high-level qualifiers.

### Setup Interface With DB2

Before the installation can proceed, you must bind an application plan for use by the DB2 Call Attach Facility (CAF) service. You will need BINDADD authority to run the BIND job.

#### To bind an application plan:

1. Edit bind control card.

Edit the CACBCNTL member in the SCACSAMP data set and change the term *DSN* to the appropriate DB2 subsystem ID for your site. If required by your site standards, you can also change the plan name from CAC22PLN to a name that fits those standards.

2. Run bind job.

Edit the bind JCL member CACBIND to conform to your environment. This job uses the provided DBRM (MSLP2EC), to create a plan CAC22PLN (or the name provided by you in CACBCNTL).

3. Submit the CACBIND job.
4. Provide EXECUTE authority for the plan.



For example, you can use the following statement if you want to give execute authority to all:

```
GRANT EXECUTE ON PLAN CAC22PLN TO PUBLIC;
```

## Import DB2 Table Definitions And Create Logical Tables

The SCACSAMP data set on the mainframe contains a member with an example of the syntax used to extract information on the layout of the EMP table from DB2 catalog and create the eXadas logical table CAC.EMPLDB2. The member name is CACDB2UG.

### To import DB2 table definitions:

- Customize the statements in member CACDB2UG for your environment.
  - a. Change the value of *DSN* to be the appropriate DB2 subsystem ID for your site.
  - b. If you bound the plan with some other name, change the name of the plan CAC22PLN accordingly.
  - c. Provide the fully qualified two-part DB2 table name for the sample table EMP.

## Loading the eXadas Catalogs

### To load the eXadas catalogs with the table you created in the previous section:

1. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, you can skip this step.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the eXadas Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the eXadas Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

2. Load the catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Make sure the symbolic, GRAMMAR =, is pointing to the appropriate USE GRAMMAR member (GRAMMAR=CACDB2UG).
- b. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has run successfully, the catalogs are loaded with the logical tables.

A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

### 3. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.
- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the DB2 Data With SQL (Locally)

### To access the DB2 data with SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the eXadas Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ....
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP 2 1 4 5 4 5M 5M CACQPCF
SERVICE INFO ENTRY = CACQP ACCOUNTING . . .
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Define a CAF Service for DB2 Access.

DB2 database connections are created and managed by a separate DB2 CAF service. The following example shows the configuration of this service.

```
SERVICE INFO ENTRY = CACCAF DSN 2 1 5 1 4 5M 5M CAC22PLN
```

The service name DSN in the CAF service identifies the DB2 subsystem the service connects to for user queries.

This service has a minimum task count of 1, a maximum task count of 5, and a maximum connections count of 1. Unlike a Query Processor task, the CAF task requires the maximum connections to be 1 as only one connection to DB2 can be created by each task. Based on the definition in the example, a maximum of 5 concurrent users can access the DB2 subsystem DSN at a time. Based on the preceding SERVICE INFO ENTRY example there is a possibility that some users will not be able to connect to DB2 as the Query Processor Service supports 20 concurrent users and the DB2 CAF service only supports 5.

3. Start eXadas Server and access data from local client.

Before starting the Server you must ensure the DB2 symbolic and STEPLIB DD have been uncommented and modified for your environment.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

a. Uncomment the SQL statement(s) for DB2.

In the SCACSAMP data set there is a member called CACSQL. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD which we'll talk about next.

**NOTE:** If you are testing for your own DB2 tables, you will want to create your own SQL in the same member (CACSQL) or create another member for your own SQL statements.

b. Configure the client.

The client configuration file is used to communicate to the eXadas Server using the communication protocol defined in the eXadas Server. In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,” on page 45 in Chapter 4, “OS/390 Installation.”](#)

c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the eXadas Server using CACSQL as the input SQL. Customize the JCL to run in your environment and submit.

d. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the eXadas Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the DB2 Data With SQL (Remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

It is assumed you are providing the client application(s) that utilize one of the eXadas Connector, for example, MSAccess, MSEXcel, Crystal Reports, IIS, WebSphere, and so on.



# Bringing Datacom On Line

## Introduction to Bringing Datacom On Line With eXadas

This chapter describes the steps necessary to bring Datacom data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the Datacom files have been brought on line, you are ready to access the data using standard SQL.

This chapter describes the process using the sample Datacom table CUST, which is part of a sample database usually created during the installation process of the Datacom DB product. If this sample Datacom table is not available in your shop, you can use one of yours. The steps outlined here are the same steps used to bring your own Datacom tables online with the eXadas Server.

The process includes:

- Running the standard Datacom utility called DDUTILTY to punch out the file definition of the Datacom table in a COBOL copybook format.
- Mapping the Datacom copybook using the eXadas Data Mapper to create logical tables.
- Loading the eXadas Catalogs with the logical tables.
- Accessing the Datacom data with SQL.

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe and the DataMapper is installed on a PC.

For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

The process is described in detail in the section that follows.

## Bringing Datacom On Line

During the installation of the Datacom DB product, a sample Datacom table called CUST is usually created under a sample database. It contains 116 records of customer information. We will be bringing this table on line with eXadas. If you do not have this table installed on your system, the same process described below can be used to bring your own Datacom tables online with eXadas.

**WARNING:** In all of the jobs that follow, you will need to customize the JCL as appropriate for your site. This includes concatenating Datacom specific libraries provided by Computer Associates. Templates for these libraries are commented out in the JCL. You will need to uncomment them as well as provide the appropriate high-level qualifier.

### Punch out file definition in COBOL format

The SCACSAMP data set on the mainframe contains a job to punch out the layout of the CUST table in COBOL copybook format. The member name is CACDCSLG. Customize the JCL to run in your environment and submit. This creates a copybook called caccus.fd in a location specified by you. The steps involved to customize this job are:

- change DCOM symbolic to Datacom high-level qualifier,
- change DCOU symbolic to copybook output HLQ,
- supply authorized user name in '-USR' statement,



- supply user access code (password) in ‘-USR’ statement,
- supply CA-DATACOM/DB table occurrence entity-name in the ‘-utl copy,table’ statement (CUST, if you are using the sample Customer table)
- supply CA-DATACOM/DB table occurrence status/version in the ‘-utl copy,table’ statement.
- supply a member name in the ‘-utl copy,table’ statement (for example, caccusfd)

After the job completes, with a return code of 0, edit the generated member caccus.fd and remove the first line (this line is non-standard COBOL and will cause syntax errors if not removed).

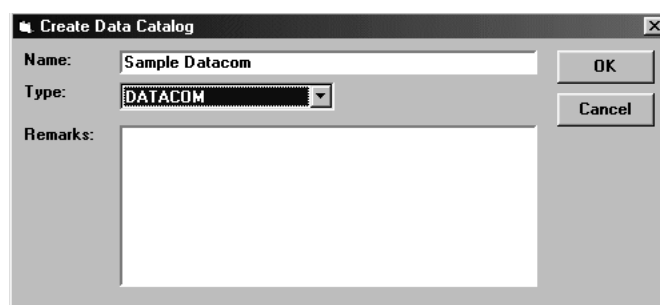
## Mapping the Sample Datacom Copybook

The COBOL copybook generated by the above step, CACCUSFD, is the basic input to the mapping process described below.

For more detailed information on data mapping, see the *eXadas DataMapper Guide*.

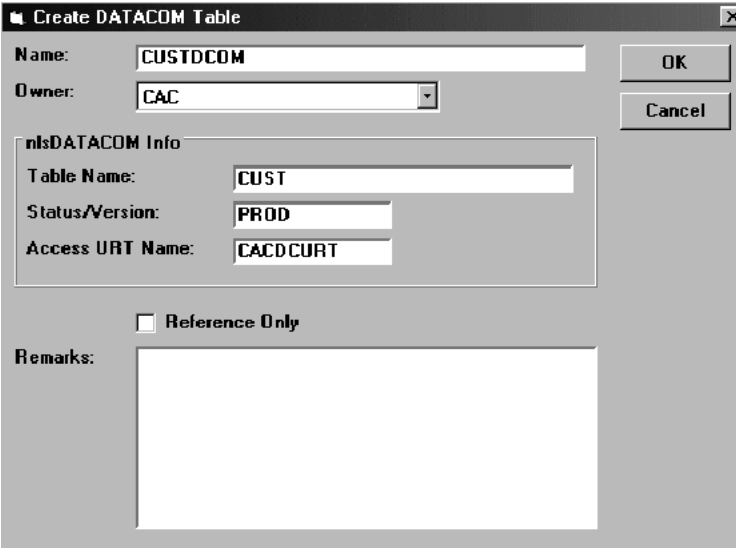
### To map the sample Datacom copybook:

1. FTP the CACCUSFD member to a directory of your choice on the PC where the Data Mapper is installed as CACCUS.FD.
2. From the Windows Start menu, select the eXadas DataMapper.
3. From the File menu, select Open Repository and select the “Sample.mdb” repository in the xadata directory.
4. From the Edit menu, select Create a new Data Catalog. The following screen appears.



5. Enter the following information in the dialog box:
  - Name: Sample Datacom
  - Type: DATACOM
6. Click OK.

7. From the Window menu, select List Tables. Since this is a new Data Catalog, the list of tables will be empty.
8. From the Edit menu, select Create a new Table.



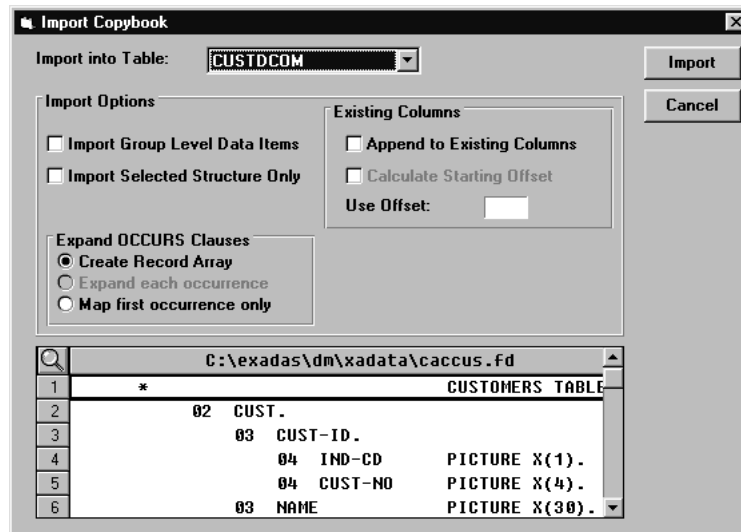
The screenshot shows a dialog box titled "Create DATACOM Table". It has a close button in the top right corner. The dialog contains the following fields and controls:

- Name:** A text input field containing "CUSTDCOM".
- Owner:** A dropdown menu showing "CAC".
- Table Name:** A text input field containing "CUST".
- Status/Version:** A text input field containing "PROD".
- Access URT Name:** A text input field containing "CACDCURT".
- Buttons:** "OK" and "Cancel" buttons are located on the right side.
- nlsDATACOM Info:** A section containing the "Table Name", "Status/Version", and "Access URT Name" fields.
- Reference Only:** A checkbox that is currently unchecked.
- Remarks:** A large empty text area for notes.

9. Enter the following information in the Create DATACOM Table, dialog box:
    - Name: CUSTDCOM
    - Owner: CAC
    - The Datacom table: CUST
    - A value for Status/Version: PROD
    - The name of the URT module that will be used to access Datacom: CACDCURT

CUST will be the Datacom table opened when the eXadas logical table CAC.CUSTDCOM is accessed with an SQL statement.
  10. Click OK.
- You are now ready to import the definitions from the caccus.fid copybook you FTPed from SCACSAMP data set.
11. From the File menu, select Import External File and select the caccus.fid copybook that you stored on the PC.

Once the Import Copybook dialog box appears, the caccus.fd copybook is loaded and ready to Import.



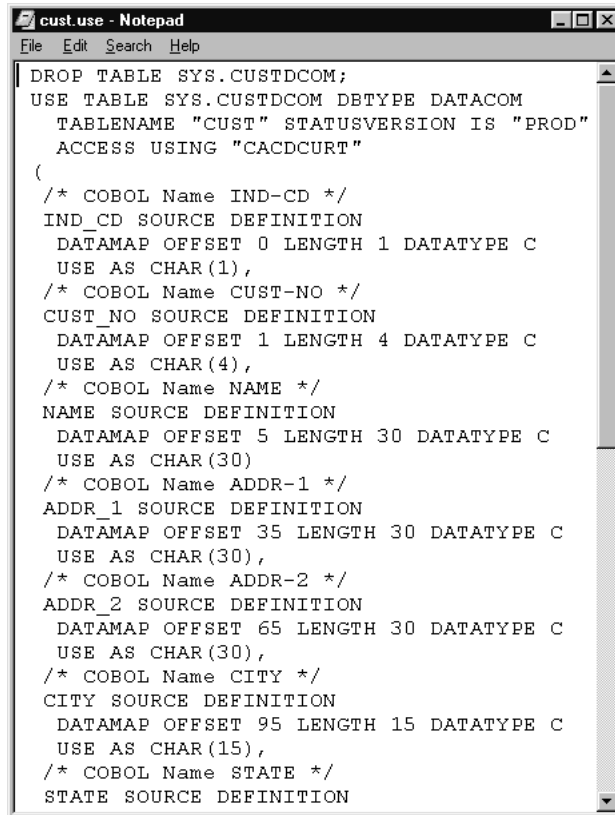
- Click Import. This imports the COBOL definitions from the caccus.fd copybook into the table CAC.CUSTDCOM.

III	Column Name	Offset	Lth	SQL Datatype	
1	IND_CD	0	1	CHAR(1)	COBOL Name IND-CD
2	CUST_NO	1	4	CHAR(4)	COBOL Name CUST-NO
3	NAME	5	30	CHAR(30)	COBOL Name NAME
4	ADDR_1	35	30	CHAR(30)	COBOL Name ADDR-1
5	ADDR_2	65	30	CHAR(30)	COBOL Name ADDR-2
6	CITY	95	15	CHAR(15)	COBOL Name CITY
7	STATE	110	2	CHAR(2)	COBOL Name STATE
8	ZIP	112	9	CHAR(9)	COBOL Name ZIP
9	CRED_IND	121	1	CHAR(1)	COBOL Name CRED-IND

- Modify the view of the table as desired. Columns can be deleted and column names can be changed.
- Close the Columns for DATACOM Table CUSTDCOM dialog box.
- Close the DATACOM Tables for Data Catalog Sample Datacom dialog box.  
At this point, you should be back to the list of Data Catalogs dialog box entitled Sample.mdb.
- Ensure that Data Catalog Sample Datacom - DATACOM is highlighted and select Generate USE Statements from the File menu.
- Select a file name for the generated statements to be stored on the PC, for example: cust.use.

Once generation is complete you can view the USE GRAMMAR from the Windows notepad or click Yes when the Data Catalog USE Generation Results

dialog box appears. The following is an example of what your completed USE GRAMMAR might look like:



```

cust.use - Notepad
File Edit Search Help
DROP TABLE SYS.CUSTDCOM;
USE TABLE SYS.CUSTDCOM DBTYPE DATACOM
  TABLENAME "CUST" STATUSVERSION IS "PROD"
  ACCESS USING "CACDCURT"
(
  /* COBOL Name IND-CD */
  IND_CD SOURCE DEFINITION
  DATAMAP OFFSET 0 LENGTH 1 DATATYPE C
  USE AS CHAR(1),
  /* COBOL Name CUST-NO */
  CUST_NO SOURCE DEFINITION
  DATAMAP OFFSET 1 LENGTH 4 DATATYPE C
  USE AS CHAR(4),
  /* COBOL Name NAME */
  NAME SOURCE DEFINITION
  DATAMAP OFFSET 5 LENGTH 30 DATATYPE C
  USE AS CHAR(30)
  /* COBOL Name ADDR-1 */
  ADDR_1 SOURCE DEFINITION
  DATAMAP OFFSET 35 LENGTH 30 DATATYPE C
  USE AS CHAR(30),
  /* COBOL Name ADDR-2 */
  ADDR_2 SOURCE DEFINITION
  DATAMAP OFFSET 65 LENGTH 30 DATATYPE C
  USE AS CHAR(30),
  /* COBOL Name CITY */
  CITY SOURCE DEFINITION
  DATAMAP OFFSET 95 LENGTH 15 DATATYPE C
  USE AS CHAR(15),
  /* COBOL Name STATE */
  STATE SOURCE DEFINITION

```

## Loading the eXadas Catalogs

To load the catalogs with the table you created in the previous section:

1. FTP the generated USE GRAMMAR `cust.use` to the SCACSAMP data set on the mainframe.
2. Assemble User Requirements Table (URT).
  - a. Edit the member CACDCURT in the SCACSAMP data set and ensure that the entry for the CUST table is accurate. If you are mapping some other Datacom table, make an entry for that table.
  - b. Edit the member CACDCURA and customize it for your environment and submit. This JCL assembles and links the URT and stores the module in the SCACLOAD data set.
3. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, you can skip this step.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the eXadas Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the eXadas Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

4. Edit the member CACDCID to ensure that the Datacom security information is correct.
5. Load the catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Make sure the symbolic, GRAMMAR=, is pointing to the appropriate USE GRAMMAR member (GRAMMAR=CUSTUSE).
- b. Uncomment and set the Datacom symbolic, the STEPLIB DDs that point to your Datacom libraries, and the DDIDENT DD.
- c. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs are loaded with the logical tables created in the DataMapper.

**NOTE:** A return code of four is expected. The DROPTABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

6. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.
- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

**NOTE:** If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the Datacom Data With SQL (Locally)

To access Datacom data locally, using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...  
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Uncomment and review SERVICE INFO ENTRY (SIE) for DCOM.

```
SERVICE INFO ENTRY = CACDCI DCOM 2 1 1 50 4 5M 5M 4
```

The last field controls the number of connections to be opened with a Datacom MUF. A value of 4 is usually safe and sufficient. However, you can change this value if required by your site standards.

3. Start eXadas Server and access data from local client.

**NOTE:** Before starting the Server you must ensure the Datacom symbolic and STEPLIB DDs have been uncommented and modified for your environment.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for Datacom.

In the SCACSAMP data set you will also find a member called CACSQ. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD which we'll talk about next.

**NOTE:** If you are testing for your own Datacom tables, you want to create your own SQL in the same member (CACSQ) or create another member for your own SQL statements.

- b. Configure the client.

The client configuration file is used to communicate to the Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,”](#) on page 45 in [Chapter 4, “OS/390 Installation.”](#)

- c. Execute local client

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the eXadas Server using CACSQ as the input SQL. Customize the JCL to run in your environment and submit.

- d. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

**NOTE:** This step is designed to issue SQL statements locally to the eXadas Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the Datacom data with SQL (remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.



# Bringing IDMS On Line

## Introduction to Bringing IDMS On Line With eXadas

This chapter describes the steps necessary to bring IDMS data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the IDMS files are brought on line, you are ready to access the data using standard SQL.

The following pages describe this process using a sample IDMS database called Employee Demo Database. This database is part of the IDMS installation. The steps outlined in this chapter are the same steps used to bring your own IDMS databases online with eXadas.

The process involves:

- Punching schema and subschema.
- Mapping IDMS schema/subschema using the eXadas Data Mapper to create logical tables.

- Loading the eXadas Catalogs with the logical tables.
- Accessing the IDMS data with SQL.

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe and the Data Mapper is installed on a PC.

The process is described in detail in the section that follows.

**NOTE:** For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

## Bringing IDMS On Line

Your IDMS installation should have a sample database called Employee Demo Database. The Schema named EMPSCHEM and Subschema named EMPSS01 identify this demo database. This is the database we will bring online with eXadas. The same process described in this section can be used to bring your own IDMS databases online with eXadas.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate for your site. This includes concatenating IDMS-specific libraries provided by Computer Associates. Templates for these libraries are included in the JCLs. You will need to uncomment them and provide the appropriate high-level qualifiers.

## Punching the Schema and Subschema

The SCACSAMP data set on the mainframe contains a member called CACIDPCH. This member contains sample JCL that can be used to punch the schema and subschema.

### To punch the schema and subschema:

- Customize JCL to run in your environment and submit.

By default, this job creates two members in the SCACSAMP data set called CACIDSCH and CACIDSUB. These newly created members contain the schema and subschema for the Employee Demo Database.

## Mapping the IDMS Schema and Subschema

The SCACSAMP data set on the mainframe now contains the schema and subschema created when you customized the JCL. This schema and subschema describe the Employee Demo Database you will map to this database. The files must be placed in your editor or in library members to be transferred to your PC.

For more detailed information on data mapping, see the *eXadas DataMapper Guide*.

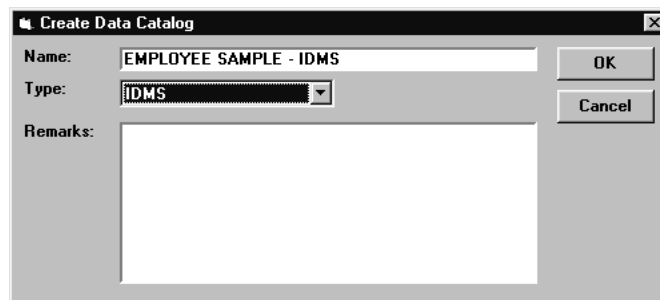
### To map the IDMS schema and subschema:

1. FTP the CACIDSCH and CACIDSUB members from the SCACSAMP data set to a directory of your choice on the PC where the Data Mapper is installed.

As you FTP these members, you may rename them with the following extensions:

- cacidsch.sch
- cacidsub.sub

2. From the Windows Start menu, select eXadas DataMapper.
3. From the File menu, select Open Repository and select the Sample.mdb repository in the xadata directory.
4. From the Edit menu, select Create a new Data Catalog. The following screen appears:



5. Enter the following information in the dialog box:
  - Name: Employee Sample - IDMS
  - Type: IDMS
6. Click OK.
7. From the File menu, select Load IDMS Schema for Reference.
8. From the Open dialog box, select the schema that you FTPed from the mainframe (cacidsch.sch). You will also be prompted to load the sub-schema as well (cacidsub.sub).

9. Highlight the IDMS Data Catalog and select List Tables from the Window menu. Since this is a new Data Catalog, the list of tables will be empty.
10. From the Edit menu, select Create a new Table.

**Create IDMS Table**

Name:

Owner:

Schema Name:  Version:

Subschema Name:

Database Name:

Access Module Name:   Reference Only

**Path Information**

Record Name:

Alias Name:

Add'l Recs	Set Name	Target Record
1	DEPT-EMPLOYEE	DEPARTMENT

Remarks:

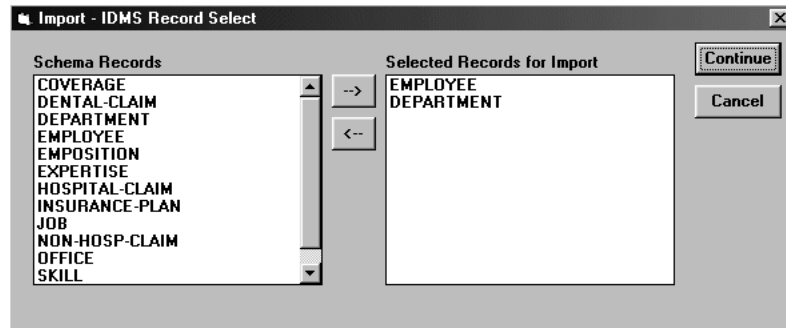
The following steps create a logical table that includes the IDMS records EMPLOYEE and DEPARTMENT as defined by the Set DEPT-EMPLOYEE.

- a. Change the name in the Name field from EMPLOYEE to EMPLIDMS.
- b. Select CAC from the Owner drop down list.
- c. Enter the Database Name of sample employee database (EMPDEMO) or the database of the file you are mapping.
- d. Select EMPLOYEE from the Record Name drop down list.
- e. Click the blue left arrow in the Add'l Recs section.
- f. Select DEPARTMENT from the Target Record drop down list.
- g. Click OK.

You are now ready to import the field definitions for the records EMPLOYEE and DEPARTMENT from the currently loaded schema.

11. From the File menu, select Import External File. You will be asked if you would like to import from the existing schema. Click Yes.

12. Click Continue from the Import - IDMS Record Select dialog box.



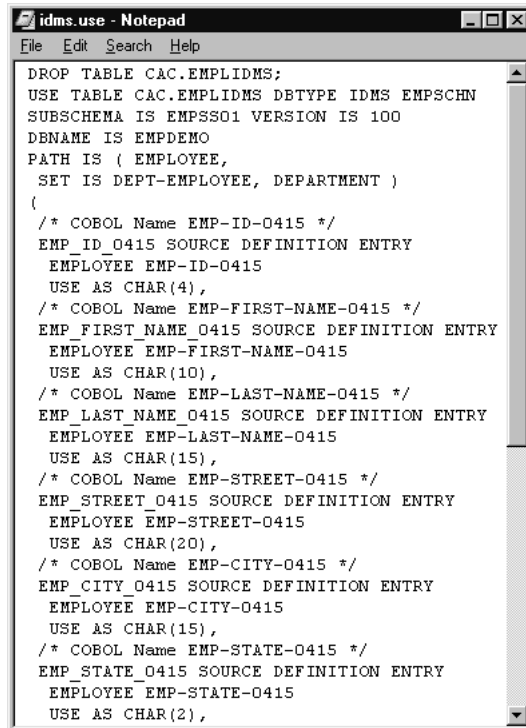
13. Click Import on the Import Copybook dialog box. This imports the COBOL definitions from the loaded schema into the table CAC.EMPLIDMS, as shown in the following example.

III	Column Name	Offset	Lth	SQL Datatype	
1	EMP_ID_0415	0	4	CHAR(4)	COBOL Name EMP-ID-0
2	EMP_FIRST_NAME_0415	4	10	CHAR(10)	COBOL Name EMP-FIRS
3	EMP_LAST_NAME_0415	14	15	CHAR(15)	COBOL Name EMP-LAS
4	EMP_STREET_0415	29	20	CHAR(20)	COBOL Name EMP-STR
5	EMP_CITY_0415	49	15	CHAR(15)	COBOL Name EMP-CITY
6	EMP_STATE_0415	64	2	CHAR(2)	COBOL Name EMP-STA
7	EMP_ZIP_FIRST_FIVE_041	66	5	CHAR(5)	COBOL Name EMP-ZIP-
8	EMP_ZIP_LAST_FOUR_0417	71	4	CHAR(4)	COBOL Name EMP-ZIP-
9	EMP_PHONE_0415	75	10	CHAR(10)	COBOL Name EMP-PHO

14. Close the Columns for IDMS Table EMPLIDMS dialog box.
15. Close the IDMS Tables for Data Catalog Employee Sample... dialog box.
- At this point, you should be back to the list of Data Catalogs dialog box entitled Sample.mdb.
16. Ensure the data catalog Employee Sample - IDMS is highlighted and select Generate USE Statements from the File menu.
17. Select a file name for the generated statements to be stored on the PC, for example: `idms.use` and click OK.

Once generation is complete you can view the USE GRAMMAR from the Windows notepad or click Yes when the Data Catalog USE Generation Results

dialog box appears. The following is an example of what your completed USE GRAMMAR might look like.



```

DROP TABLE CAC.EMPLIDMS;
USE TABLE CAC.EMPLIDMS DBTYPE IDMS EMPSCHN
SUBSCHEMA IS EMPSS01 VERSION IS 100
DBNAME IS EMPDEMO
PATH IS ( EMPLOYEE,
SET IS DEPT-EMPLOYEE, DEPARTMENT )
(
/* COBOL Name EMP-ID-0415 */
EMP_ID_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-ID-0415
USE AS CHAR(4),
/* COBOL Name EMP-FIRST-NAME-0415 */
EMP_FIRST_NAME_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-FIRST-NAME-0415
USE AS CHAR(10),
/* COBOL Name EMP-LAST-NAME-0415 */
EMP_LAST_NAME_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-LAST-NAME-0415
USE AS CHAR(15),
/* COBOL Name EMP-STREET-0415 */
EMP_STREET_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-STREET-0415
USE AS CHAR(20),
/* COBOL Name EMP-CITY-0415 */
EMP_CITY_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-CITY-0415
USE AS CHAR(15),
/* COBOL Name EMP-STATE-0415 */
EMP_STATE_0415 SOURCE DEFINITION ENTRY
EMPLOYEE EMP-STATE-0415
USE AS CHAR(2),

```

## Loading the eXadas Catalogs

To load the catalogs with the tables you created in the previous section:

1. FTP the generated USE GRAMMAR (`idms.use`) to the SCACSAMP data set on the mainframe.
2. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, you can skip this step.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

### 3. Load the catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Ensure the symbolic, GRAMMAR =, is pointing to the appropriate USE GRAMMAR member (GRAMMAR=IDMSUSE).
- b. Uncomment, and set the IDMS symbolic, the STEPLIB DDs that point to your IDMS libraries and the DDLPUNCH DD.
- c. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs are loaded with the logical tables created in the DataMapper.

A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

### 4. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the IDMS data with SQL (locally)

### To access IDMS data locally, using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the Server configuration file called CACDSCF. Within the CACDSCF search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. The name CACSAMP is the data source name defined as an ODBC Data Source. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

```
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Start eXadas Server and access data from local client.

Before starting the Server you must ensure the IDMS symbolics and DDs have been uncommented and modified for your environment. These include the DDs for the STEPLIB concatenation, the SYSCTL DD, and the SYSIDMS DD.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.



To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for IDMS.

In the SCACSAMP data set there is a member called CACSQ. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD.

**NOTE:** If you are testing for your own IDMS database you want to create your own SQL in the same member (CACSQ) or create another member for your own SQL statements.

- b. Configure the client.

The client configuration file is used to communicate to the Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,” on page 45.](#)

- c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the Server using CACSQ as the input SQL.

- d. Customize the JCL to run in your environment and submit.

- e. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel, or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the IDMS Data With SQL (remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.

# Bringing IMS On Line

## Introduction to Bringing IMS On Line With eXadas

This chapter describes the steps necessary to bring IMS data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the IMS database is brought on line, you are ready to access the data using standard SQL.

The following pages describe this process using the sample DI21PART database provided by IBM. If this sample database does not exist in your base IMS system, the steps outlined in this chapter are the same steps used to bring your own IMS database on line with the eXadas Server.

The DBD and sample COBOL copybooks describing the DI21PART database are provided in the SCACSAMP data set.

The process includes:

- Mapping the IMS database using the eXadas Data Mapper to create logical tables.
- Loading the eXadas Catalogs with the logical tables.
- Establishing the Interface to DBCTL/DRA.
- Accessing the IMS data with SQL.

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe and the Data Mapper is installed on a PC.

For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

The process is described in detail in the section that follows.

## Bringing IMS On Line

Your IMS system should have the DI21PART database installed as part of the base IMS system. Provided in the SCACSAMP data set is the DBD and two COBOL copybooks, describing the IMS database and the segments you will bring on line with eXadas. You can also use this process to bring your own IMS database online.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate for your site. This includes concatenating IMS-specific libraries provided by IBM. Templates for these libraries are included in the JCLs. You will need to uncomment them and provide the appropriate high-level qualifiers.

### Mapping the Sample IMS DBD and Copybooks

The SCACSAMP data set on the mainframe contains the DBD and COBOL copybooks describing the DI21PART database. The DBD is contained in a member called CACIMPAR and the two COBOL copybooks are in members CACIMROT (PARTROOT segment) and CACIMSTO (STOKSTAT segment). You will need these files to complete the following steps.

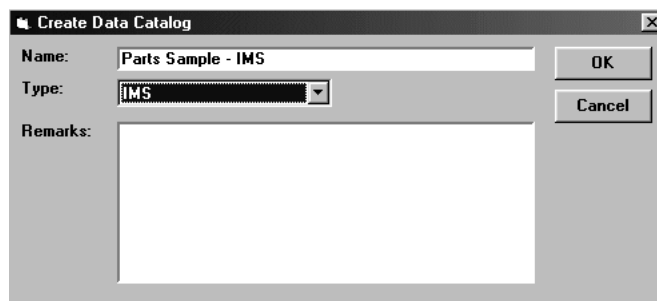
For more detailed information on data mapping, see the *eXadas DataMapper Guide*.

1. FTP the CACIMPAR, CACIMROT, and CACIMSTO members from the SCACSAMP data set to a directory of your choice on the PC where the

DataMapper is installed. As you FTP these members, you may rename them with the following extensions:

- cacimpar.dbd
- cacimrot.fd
- cacimsto.fd

2. From the Windows Start menu, select eXadas DataMapper.
3. From the File menu, select Open Repository and select the `Sample.mdb` repository under the `xadata` directory.
4. From the Edit menu, select Create a new Data Catalog. The following screen appears:



5. Enter the following information in the dialog box:
  - Name: Parts Sample—IMS
  - Type: IMS
6. Click OK.
7. From the File menu, select Load DL/I DBD for Reference. From the Open dialog box, select the DBD you FTPed from the mainframe (cacimpar.dbd).
8. From the Window menu, select List Tables. Since this is a new Data Catalog, the list of tables will be empty.

9. From the Edit menu, select Create a new Table.

The screenshot shows a dialog box titled "Create IMS Table". The fields are filled with the following values: Name: STOKSTAT, Owner: CAC, DBD Name: DI21PART, Index Root: PARTROOT, Leaf Seg: STOKSTAT, PSB Name: DFSSAM03, JOIN PSB Name: (empty), PCB Prefix: (empty), and Remarks: (empty). There are "OK" and "Cancel" buttons on the right side. A checkbox labeled "Reference Only" is located below the "PCB Prefix" field.

The following information creates a logical table that includes the IMS root segment PARTROOT and the segment STOKSTAT as defined by the DBD.

You do not need to fill in the Name field, as it is automatically populated from the Leaf Seg field.

- a. Select CAC from the Owner drop down list.
- b. Select PARTROOT from the Index Root drop down list.
- c. Select STOKSTAT from the Leaf Seg drop down list.

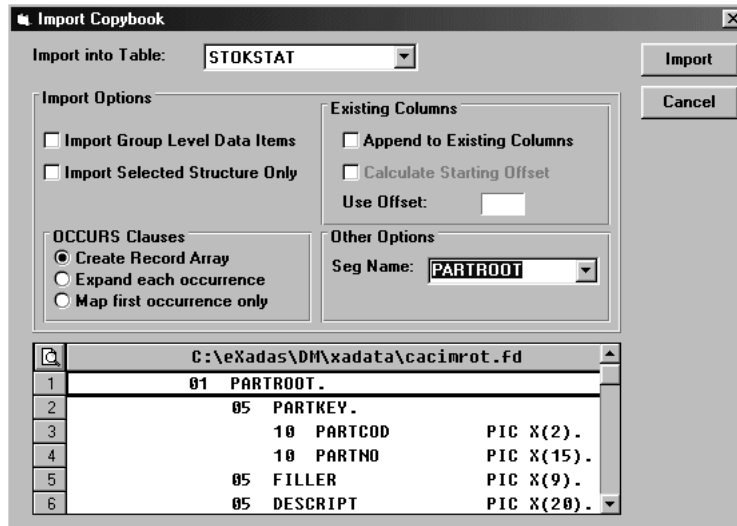
STOKSTAT is referred to as the leaf segment because it acts as the leaf segment as defined by this logical table.

- d. Enter the PSB name that will be scheduled at run-time by DBCTL. The PSB named DFSSAM03 can be used for the DI21PART database.
- e. Click OK.

You are now ready to import the definitions from the CACIMROT and CACIMSTO copybooks you FTPed from SCACSAMP data set.

10. From the File menu, select Import External File and select the CACIMROT copybook that you stored on the PC.

Once the Import Copybook dialog box appears, the CACIMROT copybook is loaded and ready to Import.

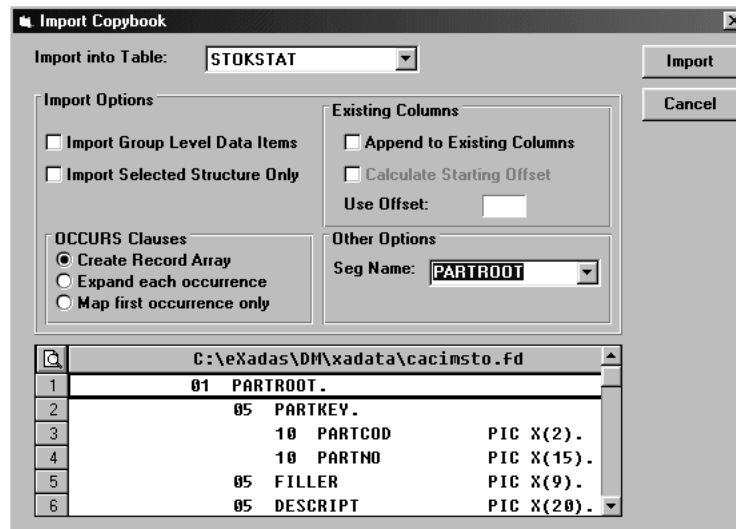


Make sure the correct segment in the Seg. Name drop down is selected. Make sure the PARTROOT segment is selected as it is the segment for which we are loading the copybook.

11. Click Import. This imports the COBOL definitions from the CACIMROT copybook into the table CAC.STOKSTAT. The columns for the table are created:

Column	Column Name	Offset	Lth	SQL Datatype	COBOL Name
1	PARTCOD	0	2	CHAR(2)	PARTCOD
2	PARTNO	2	15	CHAR(15)	PARTNO
3	DESCRIPT	26	20	CHAR(20)	DESCRIPT

12. From the File menu, select Import External File and select the next segment CACIMSTO copybook that you stored on the PC. Once the Import Copybook dialog box appears, the CACIMSTO copybook is loaded and ready to Import.



Make sure the correct segment in the seg. name drop down is selected. Make sure the STOKSTAT segment is selected as it is the segment for which we are loading the copybook. Also, make sure the Append to Existing Columns check box is checked.

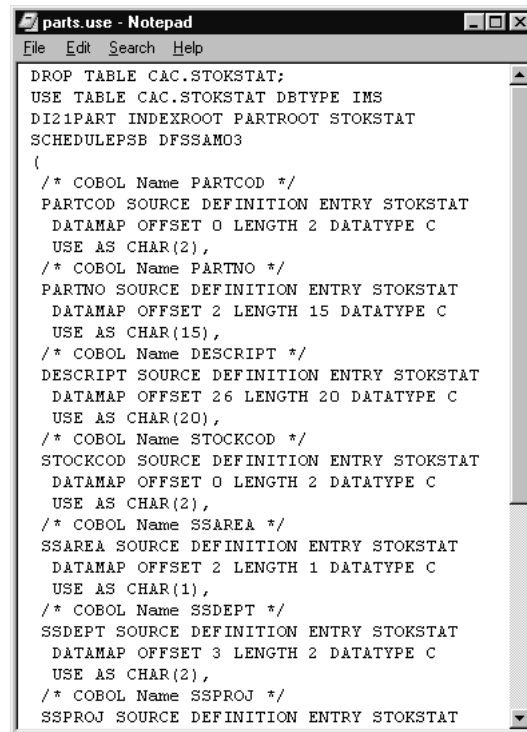
13. Click Import. This concatenates the COBOL definitions from the CACIMSTO copybook into the table CAC.STOKSTAT after the CACIMROT definitions. We have now defined a logical table which includes a root and child segment.

III	Column Name	Offset	Lth	SQL Datatype	
1	PARTCOD	0	2	CHAR(2)	COBOL Name PARTCOD
2	PARTNO	2	15	CHAR(15)	COBOL Name PARTNO
3	DESCRIPT	26	20	CHAR(20)	COBOL Name DESCRIPT
4	STOCKCOD	0	2	CHAR(2)	COBOL Name STOCKCOD
5	SSAREA	2	1	CHAR(1)	COBOL Name SSAREA
6	SSDEPT	3	2	CHAR(2)	COBOL Name SSDEPT
7	SSPROJ	5	3	CHAR(3)	COBOL Name SSPROJ
8	SSDIV	8	2	CHAR(2)	COBOL Name SSDIV
9	SSUNITPR	20	9	CHAR(9)	COBOL Name SSUNITPR

14. Close the Columns for IMS Table STOKSTAT dialog box.
15. Close the IMS Tables for Data Catalog Parts Sample IMS dialog box.  
At this point, you should be back to the list of Data Catalogs dialog box entitled Sample.mdb.
16. Ensure the Data Catalog Parts Sample - IMS is highlighted and select Generate USE Statements from the File menu.
17. Select a file name for the generated statements to be stored on the PC, for example: parts.use and click OK.



Once generation is complete you can view the USE GRAMMAR from the Windows notepad or click Yes when the Data Catalog USE Generation Results dialog box appears. The following is an example of what your completed USE GRAMMAR might look like.



```

parts.use - Notepad
File Edit Search Help
DROPTABLE CAC.STOKSTAT;
USE TABLE CAC.STOKSTAT DETYPE IMS
DI21PART INDEXROOT PARTROOT STOKSTAT
SCHEDULEPSB DFSSAM03
(
/* COBOL Name PARTCOD */
PARTCOD SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 0 LENGTH 2 DATATYPE C
USE AS CHAR(2),
/* COBOL Name PARTNO */
PARTNO SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 2 LENGTH 15 DATATYPE C
USE AS CHAR(15),
/* COBOL Name DESCRIPT */
DESCRIPT SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 26 LENGTH 20 DATATYPE C
USE AS CHAR(20),
/* COBOL Name STOCKCOD */
STOCKCOD SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 0 LENGTH 2 DATATYPE C
USE AS CHAR(2),
/* COBOL Name SSAREA */
SSAREA SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 2 LENGTH 1 DATATYPE C
USE AS CHAR(1),
/* COBOL Name SSDEPT */
SSDEPT SOURCE DEFINITION ENTRY STOKSTAT
DATAMAP OFFSET 3 LENGTH 2 DATATYPE C
USE AS CHAR(2),
/* COBOL Name SSPROJ */
SSPROJ SOURCE DEFINITION ENTRY STOKSTAT

```

## Loading the eXadas Catalogs

To load the catalogs with the table you created in the previous section:

1. FTP the generated USE GRAMMAR (parts.use), to the SCACSAMP data set on the mainframe.
2. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, you can skip this step.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the eXadas Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the eXadas Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

3. Load the catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Make sure the symbolic, GRAMMAR =, is pointing to the appropriate USE GRAMMAR member (GRAMMAR=PARTSUSE).
- b. Uncomment and set the IMS symbolic and the DBDLIB DD that points to your IMS DBD library.
- c. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs are loaded with the logical tables created in the Data Mapper.

A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

4. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

---

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.
- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Establishing the Interface to DBCTL/DRA

Now that an IMS logical table is mapped, we need to establish the interface to IMS using DBCTL. DBCTL is an IMS database manager subsystem that runs in its own address space. DBCTL must be configured and running to use the eXadas IMS DRA interface. The eXadas Server looks just like any CICS region talking to IMS. The DRA is the interface between the eXadas Server and DBCTL. The DRA start-up/router program is supplied with the IMS product and executes within the eXadas Server Address Space. For more information on DBCTL, see the IBM IMS documentation.

### To enable the DRA for use by eXadas:

1. Make the DRA start-up/router (load module DFSPRRC0) accessible to the eXadas Server by either:
  - a. copying DFSPRRC0 from the IMS.RESLIB library (built by the IMS generation process) into the SCACLOAD load library, or
  - b. concatenating the IMS.RESLIB library to the SCACLOAD STEPLIB in the SCACSAMP data set member CACDS.
2. Make the DRA start-up table (load module DFSPZPxx) accessible to the server including the IMS.RESLIB library in the server JCL.
3. Activate the IMS DRA Initialization Service.
  - a. Uncomment the SERVICE INFO ENTRY for the IMS DRA Initialization Service Task found in the SCACCONF data set member CACDSCF. This

SERVICE INFO ENTRY is identified by the comments in the configuration member.

- b. In the Task Data field at the end of the SERVICE INFO ENTRY parameter, specify additional information that is used to initialize the DRA interface. The following information must be supplied:
  - DRA Start-up Table Suffix: Modify 00 to specify the last two characters of the load module name created in DRA Setup. If you are using the default DRA start-up table load module, leave it as 00.
  - DRA user ID: Modify DRAUSER to specify the default DRA user ID that is used to connect to and register with DBCTL. The DRAUSER is the name by which the eXadas Server is known to DBCTL.

If the DRA user ID is six characters in length and the user ID is in use, eXadas will append a two-digit suffix to the user ID, starting with 00. eXadas will then try to connect with the new user ID or increment by one until it finds an available user ID. A similar process is performed with a seven character user ID and a single digit suffix.

- Default PSB Name: Modify DEFPSB to specify the name of a PSB to be used when an IMS table is referenced whose Meta Data Grammar contains no PSB name. For more information about PSB scheduling when using a DRA interface, see [Chapter 8, “Mapping Data”](#) and [Chapter 9, “Optimization”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

The default load module, DFSPZP00, is in the IMS.RESLIB library. For an example of DFSPZP00, see the *IBM IMS/ESA Installation Volume 2: System Definition and Tailoring*.

## Accessing the IMS Data With SQL (Locally)

To access IMS data locally, using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.
  - In the SCACCONF data set, see the eXadas Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...  
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Start eXadas Server and access data from local client.

Before starting the Server you must ensure the IMS symbolic and STEPLIB DDs have been uncommented and modified for your environment.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for IMS.

In the SCACSAMP data set you will also find a member called CACSQ. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD which we'll talk about next.

**NOTE:** If you are testing for your own IMS database you want to create your own SQL in the same member (CACSQ) or create another member for your own SQL statements.

- b. Configure the client.

The client configuration file is used to communicate to the eXadas Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,” on page 45](#).

- c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the eXadas Server using CACSQ as the input SQL. Customize the JCL to run in your environment and submit.

- d. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the IMS Data With SQL (Remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.

# 10

## Bringing Sequential On Line

### Introduction to Bringing Sequential On Line With eXadas

This chapter describes the steps necessary to bring Sequential data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the Sequential have been brought on line, you are ready to access the data using standard SQL.

The following pages describe this process using a sample Sequential file that was created during the installation process of the Server. A sample COBOL copybook describing this file is also provided in the SCACSAMP data set. The steps outlined in this chapter are the same steps used to bring your own Sequential files on line with the Server.

The process includes:

- Mapping the Sequential copybook using the eXadas DataMapper to create logical tables
- Loading the eXadas Catalogs with the logical tables
- Accessing the Sequential data with SQL

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe and the DataMapper is installed on a PC.

**WARNING:** Since Sequential files do not have any indexes, any SQL statement issued to this data source will result in a full scan of the Sequential file.

**NOTE:** For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

The process is described in detail in the section that follows.

## Bringing Sequential On Line

During the installation of eXadas, a sample Sequential file was created. This Sequential file contains 34 records of employee information. This is the file you will bring on line with eXadas. The same process described below can be used to bring your own Sequential files on line with eXadas.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate to your site.

### Mapping the Sample Sequential Copybook

The SCACSAMP data set on the mainframe contains a sample COBOL copybook describing the employee Sequential file created during the installation process. The member name is CACEMPFD. You need this sample copybook to complete the following steps.

For more detailed information on data mapping, see the *eXadas DataMapper Guide*.



**To map the sample Sequential copybook:**

1. FTP the CACEMPFD member from the SCACSAMP data set to a directory of your choice on the PC where the Data Mapper is installed. The filename on the PC must be cacemp.fd.
2. From the Windows Start menu, select eXadas DataMapper.
3. From the File menu, select Open Repository and select the Sample.mdb repository under the xadata directory.
4. From the Edit menu, select Create a new Data Catalog. The following screen appears:

The screenshot shows a dialog box titled "Create Data Catalog". It has three main sections: "Name" with a text input field containing "EMPLOYEE SAMPLE - SEQUENTIAL", "Type" with a dropdown menu set to "SEQUENTIAL", and "Remarks" with a large empty text area. On the right side, there are two buttons: "OK" and "Cancel".

5. Enter the following information in the dialog box:
  - Name: Employee Sample - Sequential
  - Type: Sequential
6. Click OK.
7. From the Window menu, select List Tables. Since this is a new Data Catalog, the list of tables will be empty.
8. From the Edit menu, select Create a new Table.

The screenshot shows a dialog box titled "Create SEQUENTIAL Table". It has several sections: "Name" with a text input field containing "EMPLSEQ", "Owner" with a dropdown menu set to "CAC", and "Dataset" with two radio buttons, "DS" being selected. Below the radio buttons is a text input field containing "CAC.V2R2M03.SEQ.EMPLOYEE". There is a "Record Exit" section with two text input fields, "Name" and "Max Lth", both of which are empty. Below that is a checkbox labeled "Reference Only" which is unchecked. At the bottom is a "Remarks" section with a large empty text area. On the right side, there are two buttons: "OK" and "Cancel".

9. In the Create Sequential Table dialog box:
  - a. Enter EMPLSEQ in the Name field.
  - b. Enter CAC in the Owner field.
  - c. Click the radio button labeled DS and enter the name of the Sequential data set (where VnRnMnn is the version and release level of eXadas).
  - d. Enter CAC.VnRnMnn.SEQ.EMPLOYEE in the Name field.

“Name” is name of the Sequential file created during the installation process of the Server. This file is the Sequential file opened when the table named CAC.EMPLSEQ is accessed with an SQL statement.

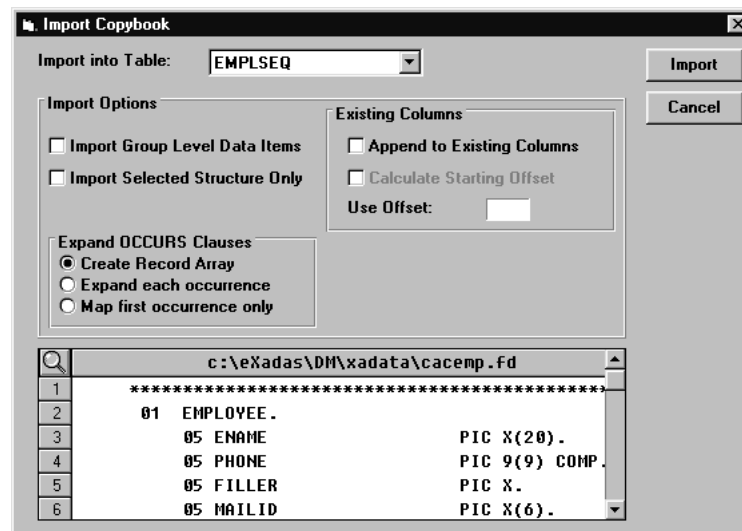
**NOTE:** When bringing your Sequential files on line with eXadas you need to enter the name of your Sequential file in this field.

- e. Click OK.

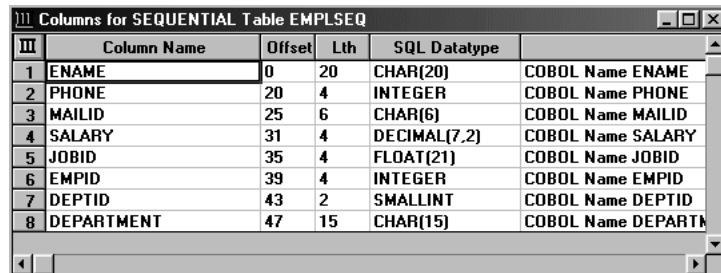
You are now ready to import the definitions from the cacemp.fd copybook you FTPed from SCACSAMP data set.

10. From the File menu, select Import External File and select the cacemp.fd copybook that you stored on the PC and click OK.

Once the Import Copybook dialog box appears, the cacemp.fd copybook is loaded and ready to Import.



11. Click **Import**. This imports the COBOL definitions from the cacemp.fd copybook into the table CAC.EMPLSEQ converting them into SQL data types.

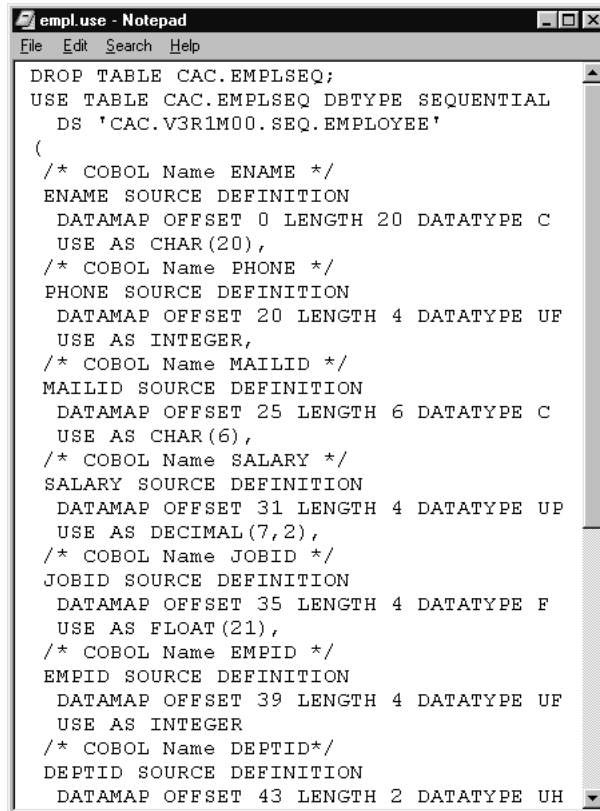


	Column Name	Offset	Lth	SQL Datatype	COBOL Name
1	ENAME	0	20	CHAR(20)	ENAME
2	PHONE	20	4	INTEGER	PHONE
3	MAILID	25	6	CHAR(6)	MAILID
4	SALARY	31	4	DECIMAL(7,2)	SALARY
5	JOBID	35	4	FLOAT(21)	JOBID
6	EMPID	39	4	INTEGER	EMPID
7	DEPTID	43	2	SMALLINT	DEPTID
8	DEPARTMENT	47	15	CHAR(15)	DEPARTM

12. Close the Columns for Sequential Table Employee dialog box.
13. Close the Sequential Tables for Data Catalog Employee Sample dialog box.  
At this point, you should be back to the list of Data Catalogs dialog box entitled Sample.mdb.
14. Ensure the Data Catalog Employee Sample - Sequential is highlighted and select Generate USE Statements from the File menu.
15. Select a file name for the generated statements to be stored on the PC, for example: emp1.use and click OK.

Once generation is complete you can view the USE GRAMMAR from the Windows notepad or click Yes when the Data Catalog USE Generation Results

dialog box appears. The following is an example of what your completed USE GRAMMAR might look like.



```

empl.use - Notepad
File Edit Search Help
DROP TABLE CAC.EMPLSEQ;
USE TABLE CAC.EMPLSEQ DBTYPE SEQUENTIAL
DS 'CAC.V3R1M00.SEQ.EMPLOYEE'
(
/* COBOL Name ENAME */
ENAME SOURCE DEFINITION
DATAMAP OFFSET 0 LENGTH 20 DATATYPE C
USE AS CHAR(20),
/* COBOL Name PHONE */
PHONE SOURCE DEFINITION
DATAMAP OFFSET 20 LENGTH 4 DATATYPE UF
USE AS INTEGER,
/* COBOL Name MAILID */
MAILID SOURCE DEFINITION
DATAMAP OFFSET 25 LENGTH 6 DATATYPE C
USE AS CHAR(6),
/* COBOL Name SALARY */
SALARY SOURCE DEFINITION
DATAMAP OFFSET 31 LENGTH 4 DATATYPE UP
USE AS DECIMAL(7,2),
/* COBOL Name JOBID */
JOBID SOURCE DEFINITION
DATAMAP OFFSET 35 LENGTH 4 DATATYPE F
USE AS FLOAT(21),
/* COBOL Name EMPID */
EMPID SOURCE DEFINITION
DATAMAP OFFSET 39 LENGTH 4 DATATYPE UF
USE AS INTEGER
/* COBOL Name DEPTID*/
DEPTID SOURCE DEFINITION
DATAMAP OFFSET 43 LENGTH 2 DATATYPE UH

```

## Loading the eXadas Catalogs

To load the catalogs with the table you created in the previous section:

1. FTP the generated USE GRAMMAR (`empl.use`) to the SCACSAMP data set on the mainframe.
2. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, this step can be skipped.

- a. In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the Server.
- b. Customize the JCL to run in your environment and submit.

Once this job completes, ensure that the Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

3. Load the Catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Make sure the symbolic, GRAMMAR =, is pointing to the appropriate USE GRAMMAR member (GRAMMAR=EMPLUSE).
- b. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs are loaded with the logical tables created in the DataMapper.

**NOTE:** A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

4. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. See [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

**NOTE:** If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the Sequential Data With SQL (Locally)

To access Sequential data locally, using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

```
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Start the eXadas Server and access data from local client.

Start the Server with the operator command **s cacds**. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for Sequential.

In the SCACSAMP data set you will also find a member called CACSQL. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD which we'll talk about next.

**NOTE:** If you are testing for your own Sequential files you want to create your own SQL in the same member (CACSQL) or create another member for your own SQL statements.

b. Configure the client

The client configuration file is used to communicate to the Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the appropriate DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,” on page 45.](#)

c. Execute local client

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the Server using CACSQL as the input SQL. Customize the JCL to run in your environment and submit.

d. View the output.

The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the resolution process as all the diagnostics are occurring on the mainframe.

## Accessing Sequential Data With SQL (Remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.





# Bringing VSAM On Line

## Introduction to Bringing VSAM On Line with eXadas

This chapter describes the steps necessary to bring VSAM data on line with the eXadas Server. The primary objective of this chapter is to provide a good foundation for the steps to take when developing and deploying your own applications.

Once the VSAM files have been brought on line, you are ready to access the data using standard SQL.

The following pages describe this process using a sample VSAM file which was created during the installation process of the eXadas Server. A sample COBOL copybook describing this file is also provided in the SCACSAMP data set. The steps outlined in this chapter are the same steps used to bring your own VSAM files on line with the eXadas Server.

**NOTE:** All references to VSAM files throughout the eXadas documentation also apply to IAM files. IAM (Innovation Access Method) is supplied by Innovation Data Processing and is a reliable, high-performance disk file manager that can be used in place of VSAM KSDS or ESDS data sets. Innovation Data Processing also

provides an optional Alternate Index feature that is fully supported by eXadas. The only exceptions are references to VSAM RRDS files, which currently are not supported by IAM.

If you would prefer to perform the VSAM IVP against an IAM file, follow the instructions in [step 9](#) (on [page 40](#)) of “[Installing a New Server](#),” to convert the sample VSAM file into an IAM file.

The process includes:

- Mapping the VSAM copybook using the eXadas DataMapper to create logical tables
- Loading the eXadas Catalogs with the logical tables.
- Accessing the VSAM data with SQL.
- Accessing VSAM data through CICS.

**NOTE:** It is assumed the eXadas Server has been installed on the mainframe and the DataMapper is installed on a PC.

The process is described in detail in the section that follows.

For additional information about developing and deploying applications with eXadas, see the *eXadas Data Integrator OS/390 Reference Guide*.

## Bringing VSAM On Line

During the installation of the eXadas Server, a sample VSAM cluster was created. This VSAM cluster contains 34 records of employee information. This is the file we will be bringing online with eXadas. The same process described below can be used to bring your own VSAM files online with eXadas.

**NOTE:** In all the jobs that follow, you will need to customize the JCLs as appropriate to your site.

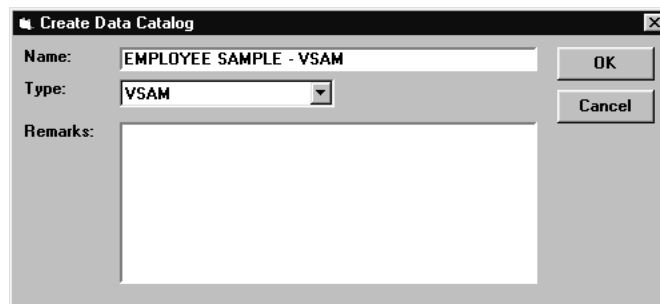
### Mapping the Sample VSAM Copybook

The SCACSAMP data set on the mainframe contains a sample COBOL copybook describing the employee VSAM cluster created during the installation process. The member name is CACEMPFD. You need this sample copybook to complete the following steps.

For more detailed information on data mapping, see the *eXadas DataMapper Guide*.

**To map the sample VSAM copybook:**

1. FTP the CACEMPF.D member from the SCACSAMP data set to a directory of your choice on the PC where the Data Mapper is installed. The filename on the PC must be `cacemp.f.d`.
2. From the Windows Start menu, select eXadas DataMapper.
3. From the File menu, select Open Repository and select the Sample.mdb repository under the xadata directory
4. From the Edit menu, select Create a new Data Catalog. The following screen appears:

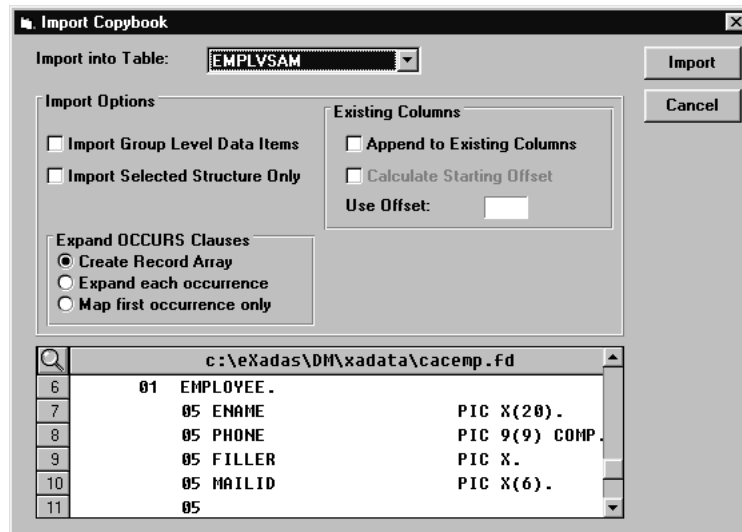


5. Enter the following information in the dialog box:
  - Name: Employee Sample – VSAM
  - Type: VSAM
6. Click **OK**.
7. From the Window menu, select List Tables. Since this is a new Data Catalog, the list of tables will be empty.

8. From the Edit menu, select Create a new Table.

9. In the Create VSAM Table dialog box:
    - a. Enter EMPLVSAM in the Name field.
    - b. Enter CAC in the Owner field.
    - c. Click the radio button labeled DS and enter the name of the VSAM data set (where VnRnMnn is the version and release level of eXadas).  
Name: CAC.VnRnMnn.VSAM.EMPLOYEE.  
“Name” is name of the VSAM cluster created during the installation process of the eXadas Server. This file is the VSAM file opened when the table named CAC.EMPLVSAM is accessed with an SQL statement.  
When bringing your VSAM files online with eXadas you need to enter the name of your VSAM cluster in this field.
  10. Click OK.
- You are now ready to import the definitions from the cacemp.fid copybook you FTPed from SCACSAMP data set.
11. From the File menu, select Import External File and select the cacemp.fid copybook that you stored on the PC and click OK.

Once the Import Copybook dialog box appears, the cacemp.fd copybook is loaded and ready to Import.



12. Click **Import**. This imports the COBOL definitions from the cacemp.fd copybook into the table CAC.EMPLVSAM converting them into SQL data types.

III	Column Name	Offset	Lth	SQL Datatype	
1	ENAME	0	20	CHAR(20)	COBOL Name ENAME
2	PHONE	20	4	INTEGER	COBOL Name PHONE
3	MAILID	25	6	CHAR(6)	COBOL Name MAILID
4	SALARY	31	4	DECIMAL(7,2)	COBOL Name SALARY
5	JOBID	35	4	FLOAT(21)	COBOL Name JOBID
6	EMPID	39	4	INTEGER	COBOL Name EMPID
7	DEPTID	43	2	SMALLINT	COBOL Name DEPTID
8	DEPARTMENT	47	15	CHAR(15)	COBOL Name DEPARTM

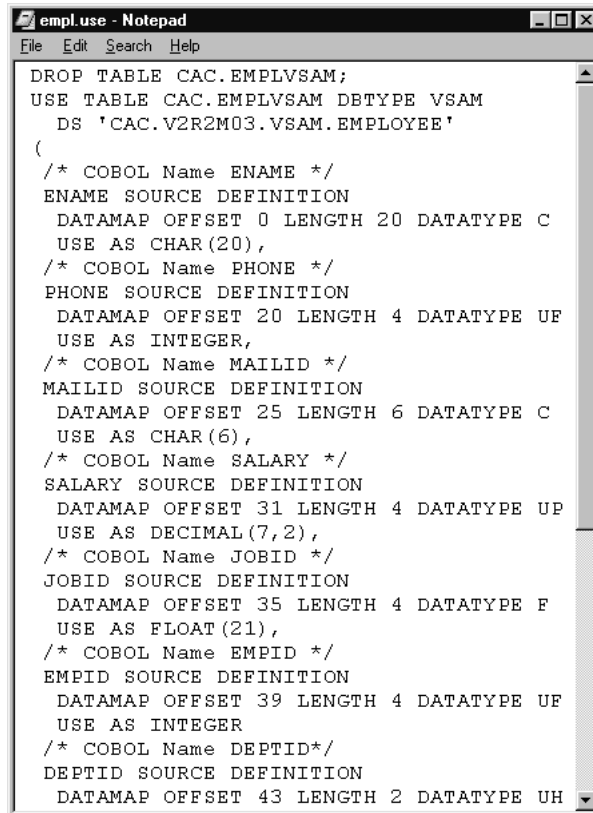
13. Close the Columns for VSAM Table Employee dialog box.
14. Close the VSAM Tables for Data Catalog Employee Sample-VSAM dialog box.

At this point, you should be back to the list of Data Catalogs dialog box entitled Sample.mdb.

15. Ensure the Data Catalog Employee Sample – VSAM is highlighted and select Generate USE Statements from the File menu.
16. Select a file name for the generated statements to be stored on the PC, such as empl.use, and click OK.

Once generation is complete you can view the USE GRAMMAR from the Windows notepad or click Yes when the Data Catalog USE Generation Results

dialog box appears. The following is an example of what your completed USE GRAMMAR might look like.



```

empl.use - Notepad
File Edit Search Help
DROP TABLE CAC.EMPLVSAM;
USE TABLE CAC.EMPLVSAM DBTYPE VSAM
DS 'CAC.V2R2M03.VSAM.EMPLOYEE'
(
/* COBOL Name ENAME */
ENAME SOURCE DEFINITION
DATAMAP OFFSET 0 LENGTH 20 DATATYPE C
USE AS CHAR(20),
/* COBOL Name PHONE */
PHONE SOURCE DEFINITION
DATAMAP OFFSET 20 LENGTH 4 DATATYPE UF
USE AS INTEGER,
/* COBOL Name MAILID */
MAILID SOURCE DEFINITION
DATAMAP OFFSET 25 LENGTH 6 DATATYPE C
USE AS CHAR(6),
/* COBOL Name SALARY */
SALARY SOURCE DEFINITION
DATAMAP OFFSET 31 LENGTH 4 DATATYPE UP
USE AS DECIMAL(7,2),
/* COBOL Name JOBID */
JOBID SOURCE DEFINITION
DATAMAP OFFSET 35 LENGTH 4 DATATYPE F
USE AS FLOAT(21),
/* COBOL Name EMPID */
EMPID SOURCE DEFINITION
DATAMAP OFFSET 39 LENGTH 4 DATATYPE UF
USE AS INTEGER
/* COBOL Name DEPTID*/
DEPTID SOURCE DEFINITION
DATAMAP OFFSET 43 LENGTH 2 DATATYPE UH

```

## Loading the eXadas Catalogs

To load the catalogs with the table you created in the previous section:

1. FTP the generated USE GRAMMAR `empl.use` to the SCACSAMP data set on the mainframe.
2. Run CACCATLG to allocate catalogs.

**NOTE:** If the catalogs have already been allocated previously, you can skip this step.

In the SCACSAMP data set, there is a member called CACCATLG. This member contains JCL to allocate the eXadas catalogs that are used by the eXadas Server.

- a. Customize the JCL to run in your environment and submit.
- b. Once this job completes, ensure that the Server Procedure in the PROCLIB points to the newly created catalogs using the CACCAT and CACINDX DD statements.

**NOTE:** Ensure that the CACCAT and CACINDX DD statements are uncommented in the JCL.

3. Load the Catalogs.

In the SCACSAMP data set, there is a member called CACMETAU. This member contains JCL to load the eXadas catalogs using the USE GRAMMAR as input.

Customize JCL to run in your environment and submit.

- a. Make sure the symbolic GRAMMAR = is pointing to the appropriate USE GRAMMAR member (GRAMMAR=EMPLUSE).
- b. Ensure the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

Once this job has been run successfully, the catalogs have been loaded with the logical tables created in the DataMapper.

A return code of 4 is expected. The DROP TABLE fails since the table does not exist yet.

**WARNING:** Once the catalogs are initialized, security has been set up with System Administration authority (SYSADM) granted to the user ID who installed eXadas and ran the Meta Data Utility. That user ID is the only user who can access the system or the catalogs. To turn off security, the new System Administrator must either grant SYSADM authorization to PUBLIC, allowing all users access and thus negating security, or grant table access authority to individual user IDs. For additional information about eXadas security, see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide*.

4. Grant catalog access rights.

Once the catalogs are loaded with the new tables, you will need to grant the appropriate access rights. In the SCACSAMP data set, there is a member called CACGRANT. This member contains JCL to load the catalogs with the appropriate access rights to the tables.

This job will read in its input from the CACGRIVP member. This member contains the GRANTs required to access the samples tables. If you are bringing your own tables on line with eXadas, you must add the appropriate GRANTs for the new tables. see [Chapter 7, “SQL Security”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for more information.

To customize the CACGRANT JCL to run in your environment, perform the following steps:

- a. Ensure the symbolic GRAMMAR = is pointing to the appropriate member containing the desired security commands.
- b. Ensure that the CACCAT and CACINDX DDs refer to the catalogs created using the CACCATLG JCL.

- c. Review CACGRIVP and uncomment the appropriate GRANT for your database.
- d. Submit.

Once this job completes, the catalogs have been loaded with the desired security.

If you plan to use tools that require access to the catalog information, you must run CACGRANT using CACGRSYS as the input.

## Accessing the VSAM Data With SQL (Locally)

### To access VSAM data locally, using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the eXadas Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...  
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Uncomment the SIE for the VSAM Service (VSAMSRV).
3. Start eXadas Server and access data from local client.

Start the Server with the operator command **s cacds**. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for VSAM.

In the SCACSAMP data set you will also find a member called CACSQ. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD, which is discussed in the next step.



**NOTE:** If you are testing for your own VSAM files you want to create your own SQL in the same member (CACSQL) or create another member for your own SQL statements.

b. Configure the client.

The client configuration file is used to communicate to the Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,” on page 45](#).

c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the Server using CACSQL as the input SQL. Customize the JCL to run in your environment and submit.

d. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the problem resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the VSAM Data With SQL (Remotely)

Connecting remotely to an eXadas Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator OS/390 Connector Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.

## Accessing VSAM Data Through CICS

eXadas provides an interface to CICS to allow VSAM data to be read, updated, inserted, or deleted through CICS.

To access VSAM data through CICS, the eXadas server will establish a VTAM LU 6.2 connection to CICS to initiate a transaction when the query begins and uses this transaction to communicate with CICS during the query. To establish this environment, VTAM and CICS definitions will be required. An additional eXadas table mapping is also required to define a table to use CICS to access VSAM data.

### VTAM Resource Definitions

A VTAM APPL definition will be required to communicate with CICS as well as creating a VTAM mode table.

Sample member CACCAPPL in the SCACSAMP data set contains sample VTAM APPL definitions. It contains two APPL definitions. One is for the eXadas Server and one is for the Meta Data Utilities. If you will be running more than one Meta Data Utility at once or want to balance the load and assign different files to different APPLs, you can add more APPL definitions by duplicating one of the existing ones and changing the name. The following is the sample member:

```
*
* SAMPLE APPL ID DEFINITIONS FOR CICS INTERFACE
*
CACCAPPL VBUILD TYPE=APPL
CACCICS1 APPL  ACBNAME=CACCICS1 ,
               APPC=YES ,
               AUTOSES=1 ,
               MODETAB=CACCMODE ,
               DLOGMOD=MTLU62 ,
               AUTH=(ACQ) ,
               EAS=100 , PARSESS=YES ,
               SONSCIP=YES ,
               DMINWNL=0 ,
               DMINWNR=1 ,
               DSESLIM=100
CACCICS2 APPL  ACBNAME=CACCICS2 ,
               APPC=YES ,
               AUTOSES=1 ,
               MODETAB=CACCMODE ,
               DLOGMOD=MTLU62 ,
               AUTH=(ACQ) ,
               EAS=1 , PARSESS=YES ,
               SONSCIP=YES ,
               DMINWNL=0 ,
               DMINWNR=1 ,
               DSESLIM=1
```

**NOTE:** The sample is set to allow 100 concurrent users. If you will be having additional users, the count on EAS and DSESLIM will need to be adjusted.

A Logon Mode Table entry will need to be created. Member CACCMODE in SCACSAMP data set contains the macro definitions to define it. This member



6. Update the **MAXIMUM** parameter in the **DEFINE SESSION** entries to increase the number of concurrent users. This should be the same as specified on the **DSESLIM** and **EAS** values in the **APPL** definition.

**NOTE:** When adding your own files to CICS, file operation **BROWSE** must be specified to allow **SELECT** queries, **ESDS UPDATE** queries, or **UPDATE**, **INSERT**, or **DELETE** queries to an **RRDS** file to process. **READ** must be specified to allow **UPDATE**, **INSERT**, or **DELETE** queries to process. **UPDATE** must be specified to allow **UPDATE** queries to process. **ADD** must be specified to **INSERT** queries to process. **DELETE** must be specified to allow **DELETE** queries to process.

After successful completion of the job, the new definitions must be installed. This is accomplished with the following CICS transaction:

```
CEDA INSTALL GROUP(CACVSAM)
```

The **CACVSAM** group should then be added to your start-up group. This is accomplished with the following CICS transaction:

```
CEDA ADD GR(CACVSAM) LIST(xxxxxxxx)
```

where *xxxxxxxx* is the name of the start-up group from your **SIT** table.

## Creating a CICS VSAM Table

Open the **EMPLOYEE SAMPLE VSAM Data Catalog** created above.

The steps to create a CICS VSAM table are the same as those for creating a VSAM table (see Steps 9-12 in the “Mapping the Sample VSAM Copybook” section) with the following changes:

1. The table name should be **CICSEMP**.
2. The file name **DD** box should be selected and the dataset name should be **CACEMP**.
3. CICS Access information should be as follows:
  - a. Enter **CACCICS1** as the **Applid**
  - b. Enter **MTLU62** for the **Logmode**
  - c. Enter **EXV1** for the **Transaction ID**
  - d. Enter a **Remote Network Name**, if required at your site.

A **CONNECT TO CICS** statement is generated using the information entered, except the **APPLID** and **Transaction ID** entered, replacing the **1** on the end of the statement with a **2**.

You must then load the catalog, following the steps in “Loading the eXadas Catalogs” section as well as the steps in “Accessing the VSAM Data With SQL (Locally)” before proceeding to the next set of steps.

## Accessing the CICS VSAM Data with SQL (Locally)

### To access the CICS VSAM data using SQL:

1. Review SERVICE INFO ENTRY (SIE) for CACSAMP.

In the SCACCONF data set, see the eXadas Server configuration file called CACDSCF. Within the CACDSCF, search for the SERVICE INFO ENTRY (SIE) containing the word CACSAMP, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
```

Uncomment this configuration parameter. It defines the data source name that is used by the client to connect to the eXadas Server. To create additional data source names, simply replicate this configuration parameter and give it a different data source name, for example:

```
SERVICE INFO ENTRY = CACQP CACSAMP ...
SERVICE INFO ENTRY = CACQP ACCOUNTING...
```

In this case, there are two valid data source names that can be configured on the client side. The first is the default CACSAMP and the second being ACCOUNTING. You can create as many of these SIEs as you need. This is a good way to separate different users based on their business needs. In this case we will be using the default data source CACSAMP.

2. Uncomment the SIE for the VSAM Service (VSAMSRV).
3. Start eXadas Server and access data from local client.

Start the Server with the operator command `s cacds`. See [Appendix C, “MTO Command Reference”](#) in the *eXadas Data Integrator OS/390 Reference Guide* for detailed information about operator commands.

To access data from a local client on the mainframe:

- a. Uncomment the SQL statement(s) for CICS VSAM.

In the SCACSAMP data set you will also find a member called CACSQL. This member contains sample SQL for the various databases supported by eXadas. This member is the input to the CACCLNT JCL pointed to by its SYSIN DD, which is discussed in the next step.

**NOTE:** If you are testing for your own CICS VSAM files you want to create your own SQL in the same member (CACSQL) or create another member for your own SQL statements.

- b. Configure the client.

The client configuration file is used to communicate to the Server using the communication protocol defined in the Server.

In the SCACCONF data set is a member called CACUSCF. Configure the DATASOURCE parameter based on the communications protocol set up in the eXadas Server described in [“Starting the eXadas Server,”](#) on page 45.

- c. Execute local client.

In the SCACSAMP data set there is a member called CACCLNT. This job executes the client batch job to issue SQL to the Server using CACSQ as the input SQL. Customize the JCL to run in your environment and submit.

- d. View the output. The output should contain the SQL statement being issued and the corresponding result set(s).

This step is designed to issue SQL statements locally to the Server, which is important if you ever need to diagnose a problem. By running the SQL locally, you eliminate potential problem areas such as the network and/or the client machine (for example, Wintel or UNIX). It also speeds up the problem resolution process as all the diagnostics are occurring on the mainframe.

## Accessing the CICS VSAM Data with SQL (Remotely)

Connecting remotely to a Server requires the use of one of the eXadas Connectors and an application that is compliant in one of the following standards:

- ODBC or
- JDBC.

See the *eXadas Data Integrator Connectors Guide* for detailed information about installing and configuring the various eXadas Connectors.

**NOTE:** It is assumed you are providing the client application(s) that utilize one of the eXadas Connectors, for example, Microsoft Access, Microsoft Excel, Crystal Reports, IIS, WebSphere, and so on.

## A

Adabas 47–53  
  accessing data 51, 52  
  creating logical table s48  
  Data Savant, minimum release level 30  
  interface 19  
  sample database 47  
  setting up the environment nt48  
APF authorization 30, 40  
Applications, binding 56

## B

Binding applications 56

## C

C precompiler 8  
  minimum release level 30  
CA-DATACOM/DB 63–72  
  accessing data with SQL 70, 72  
  loading catalogs 68  
  mapping sample copybook 65  
  punching out file definition 64  
  sample table 63, 64  
CA-DATACOM/DB Data Savant, minimum release level 30  
CA-DATACOM/DB interface 19  
Catalog access rights, granting 58, 69  
Catalogs, loading  
  Adabas 49  
  CA-DATACOM/DB 68  
  DB2 57  
  IDMS 78  
  IMS 89  
  Sequential 100  
  VSAM 110  
CHAR data type 23  
Child segments, mapping 17  
Client connectors 8, 22  
Client Interface Module 7  
Client/server, communications 3  
COBOL precompiler 8  
Communications  
  between server and application 18  
  configuring 44  
  establishing 7  
  supported protocol s15  
  transport layers 22  
Configuration members, updating 25  
Connection Handler Services 19  
Connection handlers 4  
CPU Resource Governor Exit 21  
Cross Memory 15  
  minimum release level 30

## D

Data mapping  
  *See* Mapping data  
Data Savants 19  
Data sources 16  
Data Spaces 15  
Datacom Data Savant, minimum release level 30  
Datacom. *See* CA-DATACOM/DB  
DataMapper 5, 9, 17  
DB2 55–61  
  accessing data 59, 61  
  creating logical table s57  
  interface 19  
  sample database 56  
  Thread Management Exit 22  
DB2 Data Savant, minimum release level 30  
DB2 GRAPHIC data type 23  
DBCTL database manager subsystem 91  
DELETE command 5  
Disk space requirements 28  
Double Byte Character Set 23

## E

Enterprise Server 7

## F

FTP support, DataMapper 11

## I

IBM MQ Serie s15, 41  
IDMS 73–82  
  accessing data with SQL 80, 81  
  interface 19  
  mapping schema/subschema 75  
  sample database 74  
IDMS Data Savant, minimum release level 30  
IMS 83–94  
  accessing data with SQL 92, 94  
  BMP/DBB interface 19  
  DBB/BMP Data Savant, minimum release level 30  
  DRA Data Savant, minimum release level 30  
  DRA interface 19  
  establishing DBCTL/DRA 91  
  mapping child segments 17  
  mapping data s84–89  
Initialization Services 19  
INSERT 5  
Installation 35–45  
  CXAXPARM member, editing 38  
  disk space requirements 28  
  granting authorities 40  
  IBM MQ Series 41  
  new server 37–40

**J**

- JOINS 3
  - retrieving child segments from IMS databases 17

**L**

- Load balancin g7
- Logger service 20
- LONG VARCHAR data ty e23
- LONG VARGRAPHIC data ty p 23

**M**

- Mapping dat a9–11, 16–17
  - Adabas 48
  - CA-DATACOM/DB 65
  - data sources 16
  - DB2 57
  - IDMS 75
  - IMS 84
  - Sequential 96
  - VSAM 106
- Memory requirements 31
- Meta Data Grammar 9
- Minimum disk space required 28
- Mixed-mode data 23
- MQ Series
  - See IBM MQ Series
- MTO interface 19

**N**

- National Language Support 23
- Non-relational data
  - accessing with SQL 14
  - mapping 16–17
  - translating to relational data 9

**O**

- ODBC client 8
- Operational components 3
- OS/390 minimum release level 30

**P**

- Precompiler 22

**Q**

- Queries, optimizing 3
- Query Processor configuration 25
- Query Processor Services 19
- Query Processors 5

**R**

- Record Processing Exit 22
- Region Controller 4, 18
- Result sets, converting to consistent relational form 3

**S**

- SAF Security Exit 20
- SAS/C transient library, minimum release level 30
- Sequential 95–103
  - accessing data with SQ L102, 103
  - mapping data 96
- Sequential interfa c 19
- Servers 3–6
  - automatically starting 7
  - configuring 24

- starting 45
  - storage requirements 32
- Services, controlling 18
  - SMF Accounting Exit 20
  - SMP/E installation 36
  - SNA LU 6.2 15

**SQL**

- accessing Adabas data 51, 52
- accessing CA-DATACOM/DB data 70, 72
- accessing DB2 dat a59, 61
- accessing IDMS data 80, 81
- accessing IMS data 92, 94
- accessing Sequential data 102, 103
- accessing VSAM data 112, 113
  - rewriting 3
  - validating 3
- SQL-92 14
- Storage requirements 32
- Stored procedure interface 19
- Stored procedures, invoking 19
- System exits 20

**T**

- TCP/IP 15
  - configuring 44
  - minimum release level 30
- Transport layer module, loading 7

**U**

- UPDATE statement 5
- User configuration overrides 25

**V**

- VARCHAR data type 23
- VARGRAPHIC data type 23
- VSAM 105–113
  - accessing data with SQ L112, 113
  - interface 19
  - loading catalogs 110
  - mapping data 106
- VSAM Data Savant, minimum release level 30
- VTAM LU6.2, minimum release level 30

**W**

- Work Load Manager Exit 21