IBM

Managing Oracle 8.1.7 on OS/390

- Experiences with Oracle 8i for OS/390
- Using WebSphere and UNIX System Services functions
- Managing Oracle on OS/390 with Workload Manager

Kathryn Arrell
Marc Connolly
Dennis Dutcavich
Steven Hwung
Tom Russell

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-5972-00

Managing Oracle 8.1.7 on OS/390

January 2002

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 109.

First Edition (January 2002)

This edition applies to Oracle8 and Oracle8i (8.1.7) on OS/390 V2 R6 and up.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Contents

Preface

This IBM Redbook describes some aspects of how to manage Oracle 8.1.7 and above on OS/390, such as:

- Using Workload Manager to manage Oracle workloads
- Setting up an Oracle JDBC driver for a Web client
- Using the Oracle pre-compilers and utilities on OS/390 and OS/390 UNIX System Services
- Using Recovery Manager (RMAN) with Oracle8i on OS/390
- Using partitioned tables to improve performance
- Managing the database with Oracle Enterprise Manager, including:
  - Setting up a repository
  - Connecting to the Oracle database on OS/390
  - Setting up an intelligent agent on UNIX System Services on OS/390

The book is based on real-world experiences gained by team members and others during installation of this product at:

- The IBM/Oracle EMEA Joint Solution Center, Montpellier, France
- The IBM/Oracle International Competency Center, San Mateo, California
- The IBM ITSO S/390 Center in Poughkeepsie, New York
- The Oracle Advanced Technology Center, Reston, Virginia

The use of Workload Manager, the OS/390 UNIX System Services pre-compilers, and utilities are available on Oracle 8.1.7 with OSDI, Oracle8i, and Oracle 9i. Facilities which are available on earlier versions of Oracle8 on OS/390 are mentioned throughout this book, but the primary focus is the newest capabilities of the product. Consult the appropriate Oracle documentation for details about back-level features.

This redbook will be of use to those customers who are using an Oracle8, 8i, or 9i database on OS/390.

Note

Several IBM product name changes were instituted recently. OS/390 is now referred to as z/OS. The project to produce this book was run using the old product name, so the terminology used throughout is OS/390. The information applies equally to the product now known as z/OS.

Furthermore, this redbook was prepared before Oracle 9i was available, so throughout the text we refer to Oracle 8i. References to Oracle 8i apply to the 9i release as well.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center. Material has been contributed by IBM S/390 Specialists and Oracle database specialist who support Oracle8i on OS/390.

Kathryn Arrell is a S/390 support specialist at the IBM/Oracle International Competency Center at IBM San Mateo. Previously she was an ERP specialist at the ITSO in Poughkeepsie, New York.

Marc Connolly is an Oracle Senior Mainframe Sales Consultant, Server Solutions, Oracle Service Industries. Marc provides Oracle on OS/390 technical sales support for customers in Oracle's Service Industries.

Dennis Dutcavich is a software engineer with the IBM S/390 division. He works in the ERP Business Segment and provides technical marketing support for Oracle Applications on OS/390.

Steve Hwung is a Database Specialist with IBM Global Services on assignment to the IBM/Oracle Competency Center in Redwood Shores.

Tom Russell is a S/390 Specialist with IBM Canada. Presently, he is on a special assignment to Oracle in Redwood Shores, California.

Thanks to the following people for their contributions to this project:

Jim Gillespie
ORAMAIN Consulting Services LLC

Richard Conway
Vasilis Karras
International Technical Support Organization, Poughkeepsie Center

Moshe Reder
IBM/Oracle International Competency Center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in "IBM Redbooks review" on page 117 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Using Workload Manager with Oracle 8.1.7 (OSDI)

Key to this release of OSDI is the concept of the *enclave*. In previous versions of Oracle for OS/390, work from batch, TSO, or CICS was managed by the MVS Workload Manager in a service class associated with the batch job, TSO user, or CICS address space. All client server work coming through Net8 was run at the priority of Net8.

The use of MVS enclaves extends the capability of Net8, so that work items from different sources can be distinguished from each other, and run appropriately to the business value of the work.

This chapter discusses MVS enclaves, and shows the panels used to build a Workload Manager policy for an Oracle 8.1.7 OSDI system running Oracle Applications 11.03.

1.1 Enclaves

An enclave is a transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces, and that is reported on and managed as a unit. The enclave is managed separately from the address spaces it runs in. CPU and I/O resources associated with processing the transaction are managed by the transaction's performance goal, accounted to the transaction, and reported to the transaction.

Before MVS/ESA 5.2.0, the only recipient of resource consumption and priorities was at the address space level. The address space can be swapped in or swapped out, it can be given more or less processor storage, and its priority can be adjusted to control the CPU resources given to the associated dispatchable units (that is, SRBs and TCBs). This approach is effective for traditional workloads such as TSO and batch.

However, to face the challenges of a client/server environment, a new set of dispatchable units was designed.

The performance management difficulties that the Oracle client/server workloads posed to OS/390 were that OS/390 MVS did not provide an anchor for a transaction other than an address space. All client requests came through the TNS address space and were managed under the priority for that address space.

To handle this problem, a new kind of dispatchable unit was introduced in MVS/ESA 5.2.0: the preemptible-class SRB. Together with the new SRB, a new OS/390 construct called an enclave was introduced.

Prior to OSDI, which uses these new MVS facilities, Oracle provided no ability to prioritize among requests according to their business value, since the requests are not reported as transactions to MVS. All requests are managed as part of the Net8 address space. This means that the Workload Manager controls for Net8 must be set so that the processing requirements of the requests most valuable to the business are satisfied. Any other requests that flow through the Net8 address space also get this favorable treatment, even if they are not as important to the business.

By adding this support to the OSDI release, Oracle has improved the ability of the MVS system to manage Oracle client/server workloads by enabling Workload Manager to manage the individual users' transactions within the DB server address space. Workload Manager differentiates among dispatchable units associated with different business units of work, providing an additional level of manageability and control solely through the existing Workload Manager view of goals for work.

1.2 Enclave resource accounting

A *dependent* enclave is a logical continuation of the transaction already active in a client's address space. Therefore, CPU and MSO service for a dependent enclave is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the address space's transaction. MSO service for the enclave is calculated based on the frame count of the owning address space, not on frame usage in the address space (or spaces) where the enclave is executing.

For an *independent* enclave, CPU service is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the enclave's service class or performance group period. MSO service is not calculated for an independent enclave.

For both dependent and independent enclaves, IOC service is included in the SMF 30 and 72 records associated with the address space where the enclave work is executing. SRB service for enclaves is always zero.

As of Oracle 8.1.7 (OSDI), Net8 will allocate an independent enclave for each user logging in to the database. Figure 1 on page 3 shows how TNS (Net8) transactions are now managed.

A logon request arrives from the network into the TNS address space.

- Net8 creates an independent enclave.
- Net8 calls the Workload Manager classify service. If there is an active policy with classification rules for OSDI, Workload Manager associates the enclave with a service class.
- Net8 schedules the SRB into the enclave.
- When the MVS dispatcher finds the enclave SRB, it is dispatched at the priority associated with the enclave, independent of the priority of the Net8 address space.
- The SQL code in the application uses a Program Call (PC) instruction to execute the required code in the server address space, while still getting all the CPU accounting charged to the enclave.

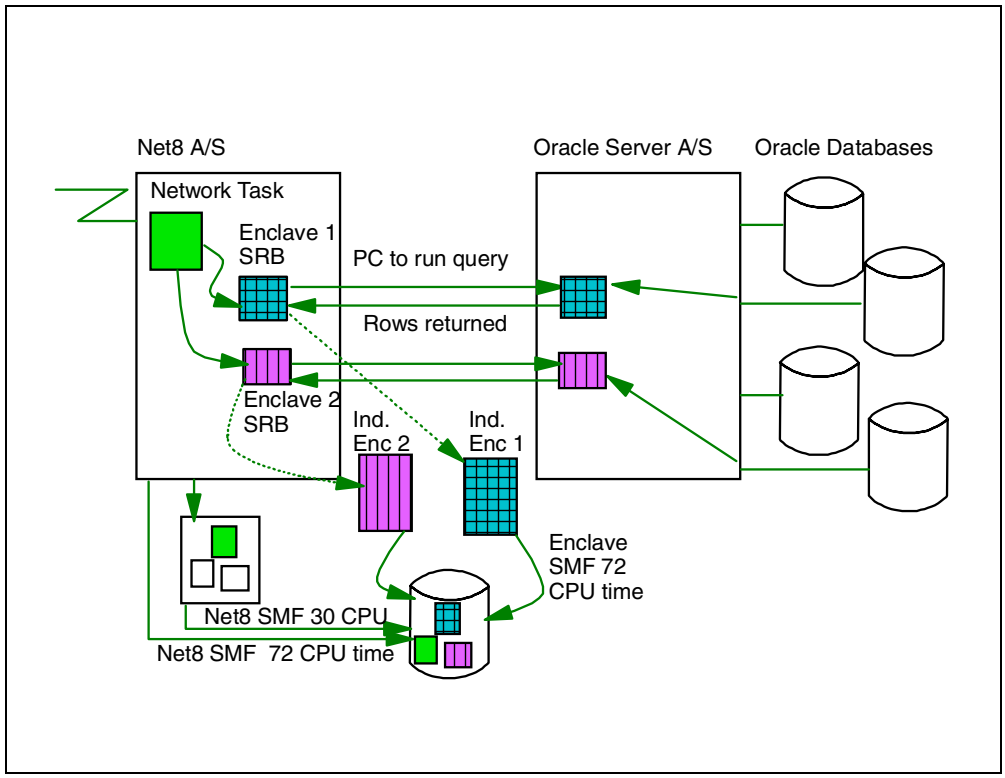

Figure 1. Managing network requests in OSDI

For the 8.1.7 release of OSDI, the classification of the request is done at logon time. This means the enclave is created when the user session is established, and deleted when the user session completes. Because the enclave processing the requests does not report each individual transaction to Workload Manager, only a velocity goal is appropriate for the service class assigned to the enclave.

We expect that later versions of OSDI will classify the individual requests and report their completion to Workload Manager. When this function is available, you will be able to assign response time goals to the individual requests, increasing the granularity of the control available to Workload Manager.

If there is no Workload Manager policy active, then all enclave SRBs will be executed using the dispatching priority of the Net8 address space. If you are running in Workload Manager compatibility mode but have a Workload Manager policy active, it is possible to manage the enclaves in a performance group (PGN) distinct from Net8 by adding the SRVCLASS= parameter to the IEAICSxx member of parmlib. However, since you must still build a Workload Manager policy and activate it in order to get this capability, we recommend that you run in goal mode.

1.3 Implementation of the Workload Manager policy

See *OS/390 Planning: Workload Management* GC28-1761 for information on the rules for defining a Workload Manager policy. This section discusses the creation of a policy to control the workload in an Oracle OSDI system that we tested. The system being measured was a pre-release Oracle OSDI database providing the database services for the Oracle Applications Suite Release 11.0.3.

1.3.0.1 The system under test

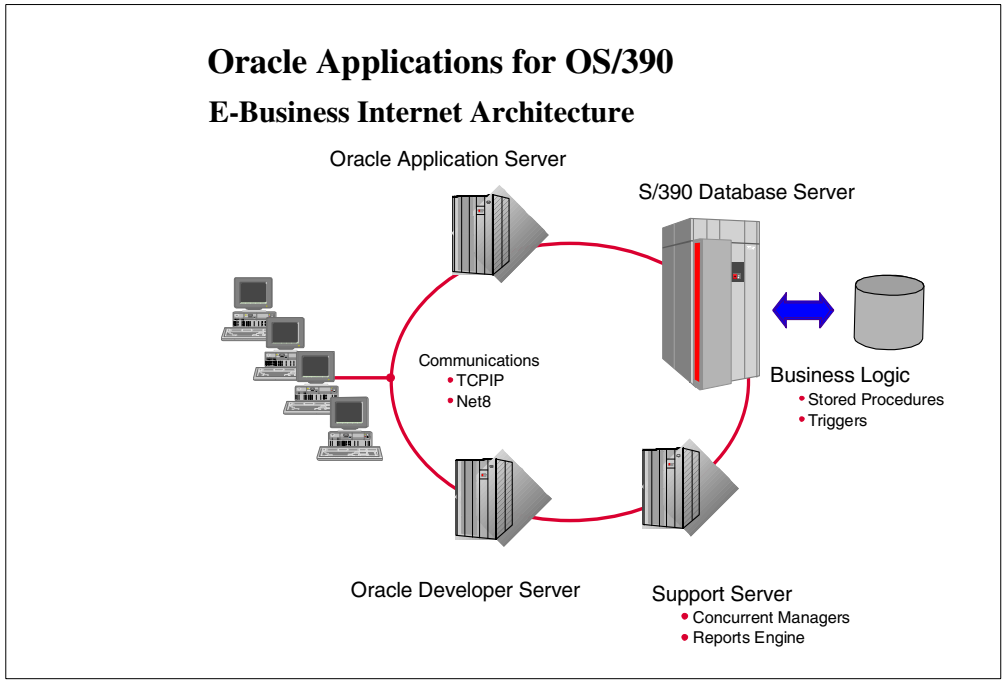

Figure 2. System under test: Logical design

In this exercise we tested the Oracle Applications Suite. In this environment the users at the PC workstations use a Web browser, such as Netscape or Internet Explorer, to access applications running on an Oracle application server. This server provides a Web interface to the browser on each user's client workstation, and accesses the database through the database server running on OS/390. Much of the Oracle Application Suite is implemented as Oracle stored procedures, which execute in the database server.

When parts of this application want to perform some operations that are not required instantly, they insert requests into a table that acts as a queue of off-line requests. This queue is processed by a server that runs the Oracle Concurrent Manager. This server retrieves queued requests from the database and processes them by executing the proper application logic, calling the database server for any database requests.

One of the major advantages of this release of Oracle is that it provides a way to distinguish database requests from the users running their database requests on the application server from the lower priority requests from the concurrent manager.

1.3.0.2 Classification

Oracle OSDI Net8 connects to Workload Manager at initialization as a new subsystem identified as 'OSDI'. The user logon requests are classified under this subsystem. Table 1 on page 5 lists the information passed to Workload Manager that can be used to classify the work request and assign a service class.

Table 1. Classification attributes for Subsystem OSDI

Attribute	Value
SI	OSDI subsystem name
UI	User ID from the client. For Oracle Applications this is the UID of the user running the application server on the middle-tier processor
NET	if SNA: client Network Name from VTAM if TCP: First eight characters of dotted IP address. (example, 100.024.)
LU	if SNA: The client LU name. if TCP: Last eight characters of dotted IP address. Note that the IP address requires leading zeros to be specified.
CT	Protocol from connect, TCP or LU6.2
SPM	Position 1 to 8. Oracle Service Name for this connection. The service name is defined in the parameters used to initialize the Oracle ODSI subsystem.
SPM	Position 9 to 89. TCP/IP hostname (left justified)

A workload manger policy is defined using an ISPF dialog. The default method of invoking the dialog is to execute the TSO command:

```
EXEC 'SYS1.SBLSCLI0(IWMARIN0)'
```

See *WLM Planning,* GC28-1761, for further details. The following examples show some of the screen dialogs presented by the Workload Manager application as we defined the policy used in our Oracle Application Standard benchmark.

The first screen shows the main selection panel for the dialog.

```
File  Utilities  Notes  Options  Help
------------------------------------------------------------------------
Functionality LEVEL004         Definition Menu         WLM Appl LEVEL008
Command ===> ____________________________________________________________

Definition data set  . . : 'RUSSELL.OLYMPIA.OSDI.POLICY'

Definition name  . . . . . WLMOSDI   (Required)
Description  . . . . . . . WLM Policy for Oracle OSDI

Select one of the
following options. . . . . _2_  1.  Policies
                               2.  Workloads
                               3.  Resource Groups
                               4.  Service Classes
                               5.  Classification Groups
                               6.  Classification Rules
                               7.  Report Classes
                               8.  Service Coefficients/Options
                               9.  Application Environments
                              10.  Scheduling Environments
```

To the Workload Manager, a “workload” is a construct used only for reporting. Resources used by various service classes can be grouped together into Workloads through the Workload Manager application. Entering a 2 on the main menu takes you to the workload definition panel. We defined a workload called

ORACLE, which contains all the service classes used to run the application. You can assign these service classes to workloads in any way you like.

```
Workload  View  Notes  Options  Help
-----------------------------------------------------------------------
                        Workload Selection List        Row 1 to 5 of 5
Command ===> ___________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar
                                                   ----Last Change-----
Action  Name      Description                      User      Date
  __    JES       Batch Jobs                       RUSSELL   2000/06/16
  __    OMVS      USS Workload                     RUSSELL   2000/06/16
  __    ORACLE    ORACLE 8i Workload               RUSSELL   2000/06/16
  __    STC       STC workload                     RUSSELL   2000/06/16
  __    TSO       TSO workload                     RUSSELL   2000/06/16
****************************** Bottom of data **************************
```

Typing a 4 on the main Workload Manager panel takes you to the Service class definition panel.

On this panel we defined four service classes for Oracle work, and assigned them all to the same workload. The service class ORACLE will be assigned by a classification rule to the address spaces that run any Oracle functions. The service classes ORAMT1, ORAMT2 and ORAMT3 will be assigned to the work running in the enclaves built by Net8.

```
Service-Class  View  Notes  Options  Help
-------------------------------------------------------------------------------
                        Service Class Selection List         Row 1 to 15 of 15
Command ===> ___________________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar


Action  Class     Description                      Workload
  __    BATHI     Batch High Priority              JES
  __    BATLO     Batch Low Priority               JES
  __    BATMED    Batch Medium                     JES
  __    ORACLE    Default Oracle                   ORACLE
  3_    ORAMT1    Oracle Mid Tier #1               ORACLE
  __    ORAMT2    Oracle Mid Tier #2               ORACLE
  __    ORAMT3    Oracle Mid Tier #3               ORACLE
  __    STCHI     STC High Priority                STC
  __    STCLO     STC Low Priority                 STC
  __    STCMED    STC Medium                       STC
  __    TSO       TSO NORMAL USERS                 TSO
  __    TSOHI     TSO High Priority                TSO
  __    USSHI     USS High Priority                OMVS
  __    USSLO     USS Low Priority                 OMVS
  __    USSMED    USS Medium                       OMVS
****************************** Bottom of data ********************************
```

```
. Service-Class  Xref  Notes  Options  Help
 ------------------------------------------------------------------------------
                      Modify a Service Class              Row 1 to 2 of 2
 Command ===> ____________________________________________________________

 Service Class Name . . . . . : ORAMT1
 Description  . . . . . . . . . Oracle Mid Tier #1
 Workload Name  . . . . . . . . ORACLE    (name or ?)
 Base Resource Group  . . . . . ________  (name or ?)

 Specify BASE GOAL information.  Action Codes: I=Insert new period,
 E=Edit period, D=Delete period.

         ---Period---  --------------------Goal--------------------
 Action  #  Duration   Imp.  Description

   __
   __    1              1    Execution velocity of 99
 ****************************** Bottom of data ********************************
```

For the release of OSDI that we tested, Net8 only called Workload Manager to classify the workload at logon time. The service class must have a goal that is appropriate for the duration of the whole session, and so we assign a velocity goal. In later releases of OSDI, Net8 will call the Workload Manager to classify each individual request, and to report the completion of each request. This will allow the use of response time goals for the Net8 workload. For the current release, however, we have assigned a velocity goal based on the source of the workload. General interactive traffic on our benchmark came from two different application servers. For this interactive traffic, arriving from Mid-tier number one and Mid-tier number 3, we assigned an Importance 1 (the highest level) and a velocity goal of 99. This ensures that Workload Manager will treat the work from these application servers with as high a priority as possible. The service classes assigned are ORAMT1 and ORAMT3

```
 Service-Class  Xref  Notes  Options  Help
 ------------------------------------------------------------------------------
                      Modify a Service Class              Row 1 to 2 of 2
 Command ===> ____________________________________________________________

 Service Class Name . . . . . : ORAMT2
 Description  . . . . . . . . . Oracle Mid Tier #2
 Workload Name  . . . . . . . . ORACLE    (name or ?)
 Base Resource Group  . . . . . ________  (name or ?)

 Specify BASE GOAL information.  Action Codes: I=Insert new period,
 E=Edit period, D=Delete period.

         ---Period---  --------------------Goal--------------------
 Action  #  Duration   Imp.  Description

   __
   __    1                   Discretionary
 ****************************** Bottom of data ********************************
```

For our benchmark, we configured the driver software so that the concurrent manager requests for data were isolated to a specific mid-tier processor. Within the Oracle Application Suite, this function is lower priority than the interactive work. Interactive users place batch requests, such as requesting that a report be printed, in a concurrent manager queue database. The concurrent manager periodically queries this queue and processes the requests that have been queued there. This is a non-critical, batch function, so we assigned the service

class ORAMT2, which was assigned to all work from Mid-tier processor number 2, a discretionary goal. This ensures that this workload is the lowest priority workload in the MVS system, and never competes for resources with the interactive workload.

```
Subsystem-Type  View  Notes  Options  Help
------------------------------------------------------------------------
                 Subsystem Type Selection List for Rules       Row 1 to 5 of 5
Command ===> ____________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar

                                                      ------Class-------
Action  Type      Description                         Service   Report
  __    JES       Classify BATCH workload             BATMED
  __    OMVS      OMVS - USS                          USSMED
  3_    OSDI      ORACLE Subsytem                     ORACLE
  __    STC       Started Tasks                       STCMED
  __    TSO       Classify TSO workload               TSO
****************************** Bottom of data ********************************
```

Typing a 6 on the main menu takes us to panels associated with the classification rules. The classification rules are organized under the various subsystems that own work and call Workload Manager to classify this work. The JES subsystem rules define the rules for classifying batch jobs. Here we select the OSDI subsystem to add classification rules that will be used when Net8 classifies work arriving from the Net8 client.

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type        Row 1 to 5 of 5
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : OSDI        Fold qualifier names?   Y  (Y or N)
Description  . . . ORACLE Subsytem

Action codes:  A=After    C=Copy         M=Move     I=Insert rule
               B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
                                                        More ===>
         -------Qualifier-------------            -------Class--------
Action    Type       Name     Start                Service     Report
                                             DEFAULTS: ORACLE      ________
 ____  1  SI         ORAC     ___                  ________    ________
 ____  2    NET        010.100. ___                ________    ________
 ____  3      LU         001.080  ___             ORAMT1      ________
 ____  3      LU         002.082  ___             ORAMT2      ________
 ____  3      LU         003.081  ___             ORAMT3      ________
***************************** BOTTOM OF DATA ******************************
```

For our test configuration for Oracle applications, we had three mid-tier processors running the Oracle applications server.

These three servers were connected through distinct TCP/IP addresses. We had configured these application servers so that the Concurrent Manager (batch) requests all came from the TCP/IP address 10.100.1.82. The other two application servers, on 10.100.1.80 and 10.100.3.81, delivered standard application interactive traffic. The rules in the above classification rule panel assign three different service classes to the work arriving from the three mid-tier processors.

Note the nesting of the rules. The first level rule applies to all OSDI workload from the Oracle SID ORAC. Using the Insert Sub-rule (IS) command allows you to enter a level two rule for NET. Referring to Table 1 on page 5, you can see that the value passed by Net8 in the NET attribute is the first eight characters of the TCP/IP address. Note the requirement for leading zeros on the numeric portion of the address. Similarly, third level rules for the LU attribute check the last eight characters of the TCP/IP address.

This set of nested rules will assign the service class ORAMT1 to work arriving in Net8 from the client at IP address 10.100.1.80, ORAMT2 to work from 10.100.2.82, and ORAMT3 for work from 10.100.3.81.

The service class ORACLE is assigned as the default service class to work classified by Net8 that doesn't get assigned a service class by the above classification rules. In our case, there was no work assigned this service class.

If you were running a test Oracle OSDI instance on this MVS system, you could use an additional level one rule to classify work for a different Oracle SID coming from a Net8 instance.

If the test instance had its own Oracle subsystem, you could add a level one rule to check for SI. If the test instance was an additional service of an existing subsystem, the level one check would be for SPM position 1 to 8, which is the service name assigned by the Oracle Subsystem. (See *Oracle8i Enterprise Edition for OS/390 with OSDI System Administration Guide, A85482-1*, for a description of the parameters supplied to the initialization of the Oracle subsystem.) By classifying on the test instance, you can assign a different service class that has a goal with lower importance than the production rules.

In the future, when Net8 supports response time goals, you will be able to use attributes such as transaction name or plan name to classify work from Net8.

The above rules classify work from Net8, that is, work that is classified under subsystem OSDI. The Net8 address space itself, and the Oracle database server address spaces, must also have a service class assigned. These address spaces are classified by MVS under subsys STC.

```
Subsystem-Type  Xref  Notes  Options  Help
-------------------------------------------------------------------------
               Modify Rules for the Subsystem Type        Row 1 to 1 of 1
Command ===> ___________________________________________ SCROLL ===> PAGE

Subsystem Type . : STC         Fold qualifier names?   Y  (Y or N)
Description  . . . Started Tasks

Action codes:  A=After    C=Copy         M=Move     I=Insert rule
               B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
                                                          More ===>
         -------Qualifier-------------             -------Class--------
Action    Type       Name     Start                Service     Report
                                          DEFAULTS: STCHI      ________
  ___  1  TN         ORAC*    ___                  SYSMED      ________
****************************** BOTTOM OF DATA ******************************
```

The above classification rule for subsys=STC assigns a default service class of STCHI to any started tasks that do not match a rule, or do not have a system-delivered default. The started tasks required for MVS operation are

delivered with attributes that cause them to be assigned the service class SYSTEM or SYSSTC automatically. The Oracle Server address spaces have little workload of their own—all the application work is managed under a service class assigned by TSO, JES, OMVS, or OSDI (Net8). The processing these DB server address spaces do that is not under the users service class consists of such things as database logging and other instance-related functions. Measurements done by Oracle show that the performance of the system is better if the database server region is lower priority than the clients, and so we recommend you assign a service class with a lower importance goal than the requests through Net8. In this case, we assigned STCMED

Assigning a priority to the server address space that is lower than the client requestor's is not the normal way that most shops configure their server address spaces; however, our measurements have shown a throughput increase by specifying the priorities in this way. As well, for Oracle 8i you should be aware that any parallel processing done by specifying a degree of parallelism (DOP) greater than one for tables that occupy more than one extent on the disk, is done by server tasks in the server address space, and run at the priority of the Oracle database address space, not the client. If you run the Oracle server address space at a high priority, then a clever user who turns on parallel processing for his or her query can significantly affect other users of the Oracle instance. See 6.2, "Parallel query" on page 71 for a discussion of Oracle parallel processing (formally known as the Parallel Query Option).

1.4 CPU Resource Accounting with OSDI and Enclaves

The implementation of OSDI will change the way the CPU time is recorded for applications using Oracle. Customers using OSDI may require changes in their resource accounting, capacity planning and operations departments.

One of these changes is the result of the use of multiple address spaces for the OSDI database server. In previous versions of Oracle, there was a single address space for the Oracle database server. CPU time consumed by work arriving from the Network was charged to Net8, and the time used by server functions was charged to the server address space. With OSDI, you can have multiple address spaces for the database server. Some CPU time is charged to each of these server address spaces, although all work that cannot be done in cross-memory mode will be performed in the primary server address space, and will be charged to that address space.

The following screen display of the SDSF monitor shows there were 25 server address spaces in the configuration being tested. Note that the CPU time for the ORAxxxx address spaces is much higher in ORA0001 than in the others. All processing that cannot be done in cross memory mode is done in the primary address space ORA0001.

CPU time consumed by the independant enclaves created for the Net8 work is reported in the ECPU column of the SDSF display. This column is normally far to the right in the default SDSF screen format, but we used the Arrange option under the View Command pull-down to move the ECPU and ECPU% columns onto the default display. Please note that the processing time for address space ORANET has a large amount of ECPU time, which is the CPU time charged to the enclaves created for the Net8 work. This CPU time is recorded as enclave

time in the SMF type 30 record for Net8, and in the type 72 SMF records for the service classes that the enclaves were assigned.

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------
SDSF DA OU02  OU02     PAG    0 SIO     2 CPU   1      LINE 1-26 (26)
NP  JOBNAME  STEPNAME SRVCLASS DP CPU%  ECPU%  CPU-TIME  ECPU-TIME REAL
    ORACNET  ORACNET  SYSSTC   FE 0.00   0.00   2001.54   45261.24 7218
    ORAC015  ORAC015  SYSSTC   FE 0.00   0.00      3.06       3.06 4381
    ORAC025  ORAC025  SYSSTC   FE 0.00   0.00      3.33       3.33 4402
    ORAC024  ORAC024  SYSSTC   FE 0.00   0.00      3.83       3.83 4246
    ORAC019  ORAC019  SYSSTC   FE 0.00   0.00      5.26       5.26 4022
    ....
    ORAC007  ORAC007  SYSSTC   FE 0.00   0.00      3.34       3.34 3978
    ORAC006  ORAC006  SYSSTC   FE 0.00   0.00      3.09       3.09 3984
    ORAC005  ORAC005  SYSSTC   FE 0.00   0.00      3.76       3.76 3888
    ORAC004  ORAC004  SYSSTC   FE 0.00   0.00     11.07      11.07 4058
    ORAC003  ORAC003  SYSSTC   FE 0.00   0.00      3.09       3.09 4072
    ORAC002  ORAC002  SYSSTC   FE 0.00   0.00      3.46       3.46 4048
    ORAC001  ORAC001  SYSSTC   FE 0.00   0.00    547.68     547.68 3802
```

Chapter 2. Web clients

The prerequisite to using the JDBC drivers with an Oracle database on S/390 is to have an IBM HTTP Server for OS/390 installed. The information for planning, installing and using the IBM WebServer on OS/390 is in the book *IBM HTTP Server for OS/390, Planning, Installing and Using,* SC31-8690. A summary of these steps is described in Appendix A, “Cloning/customizing an IBM Web server on OS/390” on page 99.

After the installation, the following OMVS files were in our /web/ora directory

ORACLE1 @ SC04:/web/ora>ls -la

```
drwxrwxrwx   7 AAAAAAA  SYS1        8192 Feb 24 15:37 .
drwxr-xr-x   4 AAAAAAA  OMVSGRP     8192 Feb 24 10:26 ..
-rw-r--r--   1 PUBLIC   SYS1     1039564 Feb 24 17:52 OracleServletJDBC.log
-rw-r--r--   1 PUBLIC   SYS1        9991 Feb 24 15:21 SQLinkServletJDBC.log
-rw-r--r--   1 AAAAAAA  SYS1           9 Feb 24 16:00 httpd-pid
-rwxrwxrwx   1 AAAAAAA  SYS1      128160 Feb 23 15:43 httpd.conf
-rwxrwxrwx   1 AAAAAAA  SYS1         502 Feb 21 00:13 httpd.envvars
drwxrwxrwx   2 AAAAAAA  SYS1        8192 Feb 25 00:00 logs
drwxrwxrwx   2 AAAAAAA  SYS1        8192 Jun  3  1999 our-cgi
drwxrwxrwx   3 AAAAAAA  SYS1        8192 Feb 21 00:15 pub
drwxrwxrwx   2 AAAAAAA  SYS1        8192 Feb 25 00:00 reports
drwxrwxrwx   2 AAAAAAA  SYS1        8192 Feb 24 17:43 servlets
lrwxrwxrwx   1 AAAAAAA  SYS1          15 Feb 24 15:37 waslogs -> /was/ora/a/logs
```

This chapter describes how to enable the IBM WebSphere Application Server to access Oracle data using JDBC.

2.1 Obtain Oracle JDBC drivers

In order for the Web servers and their Java servlets to have access to Oracle, you need the Oracle JDBC type 4 (*thin*, Java Sockets) driver class files.

The Oracle drivers are comprised of two class files: one is for National Language Support (NLS) and the other, base JDBC driver requirements.

You can obtain copies of these files directly from Oracle Corporation, at the following URL:

```
http://www.oracle.com
```

Follow the links to the *Oracle Technology Network*. There you need to register in order to obtain the Oracle JDBC Drivers.

When downloading JDBC driver files, be sure to obtain the *JDBC-Thin, 100% Java* and *NLS-zip, 100% Java* versions of the class files, and make sure you select them for the appropriate Oracle version you are using.

We tested the Oracle8i 8.1.6 JDBC “thin” Driver files specifically developed for version 1.1.x of the JDK.

Please note that at the time of this writing there are two sets of Oracle JDBC drivers. One is for use with JDK 1.1.x and the other is for the J2EE (JDK 1.2.x)

releases. Make sure you download the 1.1.x version of appropriate Oracle JDBC type 4 drivers since WAS 1.2 only supports JDK 1.1.x.

We downloaded and used the drivers in the following directory:

/usr/lpp/oracle/jdbc/classes

2.2 Configure and install Oracle for OS/390

You will also need to have Oracle for OS/390 version 8.04 or later (8.1.5) installed and configured for access via TCP/IP. Consult the Oracle for OS/390 Installation, Administration and User's guides for details.

2.3 IBM WebSphere Application Server settings for JDBC

Add the Oracle JDBC drivers to the ncf.jvm.classpath in the IBM WebSphere Application Server configuration file was.conf.

These are two zip files:

/usr/lpp/oracle/jdbc/classes/classes12.zip

/usr/lpp/oracle/jdbc/classes/nls_charset12.zip

Run update properties to activate the changes to was.conf.

This is all the setup required. To test your configuration, you can use the sample servlet for JDBC provided by Oracle. Obtain the sample from the additional material repository of the redbooks Web server at:

`ftp://www.redbooks.ibm.com/redbooks/sg245604`

It is called OracleTestServlet.java. It must be compiled and moved into the appropriate directory where you store servlets, like /web/apple/servlets, and then invoked from your browser.

`http://your_server/servlets/OracleTestServlet`

Chapter 3. Using the Oracle C/C++ pre-compilers

The Oracle C/C++ pre-compilers can be used to develop Oracle applications on OS/390 using OS/390 UNIX System Services. This is a new capability in Oracle 8.1.7 with OSDI.

This chapter describes how to do the following operations:

- Set the environment variables
- Precompile Oracle applications
  - Sample program before compile
  - Sample program after compile

You can find information about installation of OS/390 UNIX System Services facilities in the redbook *Experiences Installating Oracle8i Release 3 (8.1.7) with OSDI,* SG24-5973.

3.1 Setting the environment variables

There is a default profile, also called profile, in the /etc directory. We copied this file to an Oracle user as .profile and set the following environment variables:

```
export ORACLE_HOME=/orauss
export TWO_TASK=ORAC
export TNS_ADMIN=/orauss
export LIBPATH=$LIBPATH:/orauss/lib
export PATH=$PATH:/orauss/bin
```

3.2 Precompiling Oracle applications

To run the Pro*C/C++ precomplier in OS/390 UNIX System Services, you issue either of the following commands:

- `proc  INAME=source-file` (and/or other options separated by one or more spaces)

  or

- `proc source-file` (and/or other options separated by one or more spaces)

The second form can be used if the INAME option is entered as the first one. For the list of precompiler options and their default values, you can find them in the *Oracle Programmer's Guide to the Pro*C/C++ Precompiler*, A76942.

During precompilation, Pro*C/C++ generates C or C++ code that replaces the SQL statements embedded in the source program. The generated code contains data structures that indicate the data type, length, and address of host variables, as well as information needed by the Oracle runtime library, SQLLIB. The generated code also contains the calls to SQLLIB routines that perform the embedded SQL operations.

In order to show the effect of precompiling, we listed two files in the "Sample file before pre-compiling" on page 17 and "Sample file after pre-compiling" on page 20. The first is the source file containing the SQL statements, and the second file is generated by the precompiler.

Note: If you just enter the `proc` command, it shows the Oracle pre-compiler options that are available.

3.3 Defining users to OS/390 UNIX System Services

To allow a TSO user to access OS/390 UNIX System Services, you need to define an OMVS segment in this user's RACF profile by a systems programmer (super user). Below are the steps for giving OS/390 UNIX System Services access to a TSO user:

- Choose ISPF option R for RACF Application, and press Enter.
- Choose option 4 for user profiles and your own password, and press Enter.
- Choose option 2 and enter the userid for changing a user profile and user name, and press Enter
- Skip the first menu by pressing Enter
- Tab the cursor to the field like below and type in 'yes'

  TO ADD OR CHANGE OPTIONAL INFORMATION, ENTER YES ===>
- On 'RACF – CHANGE USER userid' screen, choose the 'OMVS parameters' option.
- On the 'OMVS Parameters' screen, enter a unique numeric user ID in the UID field, and 'yes' in both HOME and PROGRAM fields. These fields are displayed like the following.
- Specify new User Identifier (UID) ===>
- Change Initial Path Name (HOME) ===>
- Change Program Path Name (PROGRAM) ===>
- Enter Initial Directory Path Name (HOME) for this user. (For example, /u/ibmu01)
- Enter Program Path Name (PROGRAM) for this user. (For example, /bin/sh)

Note: The UID must be unique among all the users accessing OS/390 UNIX System Services, so we created a file called '$SYS.OE.USERS' to keep track of the number being assigned to the users so far. The range of the numbers is between 1 and 2147483647

3.4 Using C++

To port C++ code to OS/390 UNIX System Services, the Oracle pre-compiler sysparms just need to include the following:

```
CODE=CPP
CPP_SUFFIX=C          (or whatever your suffix name is)
```

3.5 Options to use with proc command

If you issue the **proc** command without any options, it will list your choices are shown:

```
Pro*C/C++: Release 8.1.7.0.50 - Production on Fri Dec 22 08:25:54 2000
```

```
(c) Copyright 2000 Oracle Corporation.  All rights reserved.
System default option values taken from: /orauss/precomp/admin/pcscfg.cfg
Option Name    Current Value  Description
--------------------------------------------------------------------------------
auto_connect   no             Allow automatic connection to ops$ account
char_map       charz          Mapping of character arrays and strings
close_on_commitno             Close all cursors on COMMIT
code           ansi           The type of code to be generated
comp_charset   multi_byte     The character set type the C compiler supports
config         default        Override system configuration file with another
cpp_suffix     *none*         Override the default C++ filename suffix
dbms           native         v6/v7/v8 compatibility mode
def_sqlcode    no             Generate '#define SQLCODE sqlca.sqlcode' macro
define         *none*         Define a preprocessor symbol
duration       transaction    Set pin duration for objects in the cache
dynamic        oracle         Specify Oracle or ANSI Dynamic SQL Semantics
errors         yes            Whether error messages are sent to the terminal
errtype        *none*         Name of the list file for intype file errors
fips           none           FIPS flagging of ANSI noncompliant usage
header         *none*         Specify file extension for Precompiled Headers
hold_cursor    no             Control holding of cursors in the cursor cache
iname          *none*         The name of the input file
include        *none*         Directory paths for included files
intype         *none*         The name of the input file for type information
lines          no             Add #line directives to the generated code
lname          *none*         Override default list file name
ltype          none           The amount of data generated in the list file
maxliteral     1024           Maximum length of a generated string literal
maxopencursors 10             Maximum number of cached open cursors
mode           oracle         Code conformance to Oracle or ANSI rules
nls_char       *none*         Specify National Language character variables
nls_local      no             Control how NLS character semantics are done
objects        yes            Support object types
oname          *none*         The name of the output file
oraca          no             Control the use of the ORACA
pagelen        80             The page length of the list file
parse          full           Control which non-SQL code is parsed
prefetch       1              Number of rows pre-fetched at cursor OPEN time
release_cursor no             Control release of cursors from cursor cache
select_error   yes            Control flagging of select errors
sqlcheck       syntax         Amount of compile-time SQL checking
sys_include    *none*         Directory where system header files are found
threads        no             Indicates a multi-threaded application
type_code      oracle         Use Oracle or ANSI type codes for Dynamic SQL
unsafe_null    no             Allow a NULL fetch without indicator variable
userid         *none*         A username/password Ý@dbname¨ connect string
varchar        no             Allow the use of implicit varchar structures
version        recent         Which version of an object is to be returned
PCC-F-02135, CMD-LINE:  User asked for help
```

3.6 Sample file before pre-compiling

```
/*
 *  sample1.pc
 *
 *  Prompts the user for an employee number,
 *  then queries the emp table for the employee's
```

```
 *  name, salary and commission.  Uses indicator
 *  variables (in an indicator struct) to determine
 *  if the commission is NULL.
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>

/* Define constants for VARCHAR lengths. */
#define     UNAME_LEN      20
#define     PWD_LEN        40

/* Declare variables.  No declare section is
   needed if MODE=ORACLE. */
VARCHAR     username[UNAME_LEN];  /* VARCHAR is an Oracle-supplied struct */
varchar     password[PWD_LEN];    /* varchar can be in lower case also. */

/* Define a host structure for the output values of
   a SELECT statement.  */
struct
{
    VARCHAR   emp_name[UNAME_LEN];
    float     salary;
    float     commission;
} emprec;

/* Define an indicator struct to correspond
   to the host output struct. */
struct
{
    short     emp_name_ind;
    short     sal_ind;
    short     comm_ind;
} emprec_ind;

/*  Input host variable. */
int         emp_number;

int         total_queried;

/* Include the SQL Communications Area.
   You can use #include or EXEC SQL INCLUDE. */
#include <sqlca.h>

/* Declare error handling function. */
void sql_error(msg)
    char *msg;
{
    char err_msg[128];
    size_t buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);
```

```
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK RELEASE;
    exit(EXIT_FAILURE);
}

void main()
{
    char temp_char[32];

/* Connect to ORACLE--
 * Copy the username into the VARCHAR.
 */
    strncpy((char *) username.arr, "SCOTT", UNAME_LEN);

/* Set the length component of the VARCHAR. */
    username.len =
      (unsigned short) strlen((char *) username.arr);

/* Copy the password. */
    strncpy((char *) password.arr, "TIGER", PWD_LEN);
    password.len =
      (unsigned short) strlen((char *) password.arr);

/* Register sql_error() as the error handler. */
    EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--\n");

/* Connect to ORACLE.  Program will call sql_error()
 * if an error occurs when connecting to the default database.
 */
    EXEC SQL CONNECT :username IDENTIFIED BY :password;

    printf("\nConnected to ORACLE as user: %s\n", username.arr);

/* Loop, selecting individual employee's results */

    total_queried = 0;

    for (;;)
    {
        emp_number = 0;
        printf("\nEnter employee number (0 to quit): ");
        gets(temp_char);
        emp_number = atoi(temp_char);
        if (emp_number == 0)
            break;

/* Branch to the notfound label when the
 * 1403 ("No data found") condition occurs.
 */
        EXEC SQL WHENEVER NOT FOUND GOTO notfound;

        EXEC SQL SELECT ename, sal, comm
            INTO :emprec INDICATOR :emprec_ind
            FROM EMP
            WHERE EMPNO = :emp_number;
```

```
/* Print data. */
        printf("\n\nEmployee   Salary    Commission\n");
        printf("--------   -------   ----------\n");

/* Null-terminate the output string data. */
        emprec.emp_name.arr[emprec.emp_name.len] = '\0';
        printf("%s      %7.2f      ",
            emprec.emp_name.arr, emprec.salary);

        if (emprec_ind.comm_ind == -1)
            printf("NULL\n");
        else
            printf("%7.2f\n", emprec.commission);

        total_queried++;
        continue;

notfound:
        printf("\nNot a valid employee number - try again.\n");

    } /* end for(;;) */

    printf("\n\nTotal rows returned was %d.\n", total_queried);
    printf("\nG'day.\n\n\n");

/* Disconnect from ORACLE. */
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(EXIT_SUCCESS);
```

3.7 Sample file after pre-compiling

```
/* Result Sets Interface */
#ifndef SQL_CRSR
#  define SQL_CRSR
  struct sql_cursor
  {
    unsigned int curocn;
    void *ptr1;
    void *ptr2;
    unsigned long magic;
  };
  typedef struct sql_cursor sql_cursor;
  typedef struct sql_cursor SQL_CURSOR;
#endif /* SQL_CRSR */

/* Thread Safety */
typedef void * sql_context;
typedef void * SQL_CONTEXT;

/* Object support */
struct sqltvn
{
  unsigned char *tvnvsn;
  unsigned short tvnvsnl;
```

```
  unsigned char *tvnnm;
  unsigned short tvnnml;
  unsigned char *tvnsnm;
  unsigned short tvnsnml;
};
typedef struct sqltvn sqltvn;

struct sqladts
{
  unsigned int adtvsn;
  unsigned short adtmode;
  unsigned short adtnum;
  sqltvn adttvn[1];
};
typedef struct sqladts sqladts;

static struct sqladts sqladt = {
  1,1,0,
};

/* Binding to PL/SQL Records */
struct sqltdss
{
  unsigned int tdsvsn;
  unsigned short tdsnum;
  unsigned char *tdsval[1];
};
typedef struct sqltdss sqltdss;
static struct sqltdss sqltds =
{
  1,
  0,
};

/* File name & Package Name */
struct sqlcxp
{
  unsigned short fillen;
           char  filnam[11];
};
static const struct sqlcxp sqlfpn =
{
    10,
    "sample1.pc"
};


static unsigned long sqlctx = 218773;


static struct sqlexd {
   unsigned int   sqlvsn;
   unsigned int   arrsiz;
   unsigned int   iters;
   unsigned int   offset;
   unsigned short selerr;
   unsigned short sqlety;
   unsigned int   occurs;
```

```
      const short *cud;
   unsigned char  *sqlest;
      const char  *stmt;
   sqladts *sqladtp;
   sqltdss *sqltdsp;
            void  **sqphsv;
   unsigned int   *sqphsl;
            int   *sqphss;
            void  **sqpind;
            int   *sqpins;
   unsigned int   *sqparm;
   unsigned int   **sqparc;
   unsigned short  *sqpadto;
   unsigned short  *sqptdso;
            void  *sqhstv[4];
   unsigned int   sqhstl[4];
            int   sqhsts[4];
            void  *sqindv[4];
            int   sqinds[4];
   unsigned int   sqharm[4];
   unsigned int   *sqharc[4];
   unsigned short  sqadto[4];
   unsigned short  sqtdso[4];
} sqlstm = {10,4};

/* SQLLIB Prototypes */
extern void sqlcxt (void **, unsigned long *,
                    struct sqlexd *, const struct sqlcxp *);
extern void sqlcx2t(void **, unsigned long *,
                    struct sqlexd *, const struct sqlcxp *);
extern void sqlbuft(void **, char *);
extern void sqlgs2t(void **, char *);
extern void sqlorat(void **, unsigned long *, void *);

/* Forms Interface */
static const int IAPSUCC = 0;
static const int IAPFAIL = 1403;
static const int IAPFTL  = 535;
extern void sqliem(char *, int *);

typedef struct { unsigned short len; unsigned char arr[1]; } VARCHAR;
typedef struct { unsigned short len; unsigned char arr[1]; } varchar;

/* cud (compilation unit data) array */
static const short sqlcud0[] =
{10,4130,0,0,0,
5,0,0,1,0,0,32,69,0,0,0,0,0,1,0,
20,0,0,2,0,0,27,97,0,0,4,4,0,1,0,1,9,0,0,1,9,0,0,1,10,0,0,1,10,0,0,
51,0,0,3,78,0,4,119,0,0,4,1,0,1,0,2,9,0,0,2,4,0,0,2,4,0,0,1,3,0,0,
82,0,0,4,0,0,32,150,0,0,0,0,0,1,0,
};



/*
 *  sample1.pc
 *
 *  Prompts the user for an employee number,
```

```
 *  then queries the emp table for the employee's
 *  name, salary and commission.  Uses indicator
 *  variables (in an indicator struct) to determine
 *  if the commission is NULL.
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>

/* Define constants for VARCHAR lengths. */
#define     UNAME_LEN      20
#define     PWD_LEN        40

/* Declare variables.  No declare section is
   needed if MODE=ORACLE. */
/* VARCHAR     username[UNAME_LEN]; */
struct { unsigned short len; unsigned char arr[20]; } username;
  /* VARCHAR is an Oracle-supplied struct */
/* varchar     password[PWD_LEN]; */
struct { unsigned short len; unsigned char arr[40]; } password;
    /* varchar can be in lower case also. */

/* Define a host structure for the output values of
   a SELECT statement.  */
struct
{
    /* VARCHAR   emp_name[UNAME_LEN]; */
struct { unsigned short len; unsigned char arr[20]; } emp_name;

    float     salary;
    float     commission;
} emprec;

/* Define an indicator struct to correspond
   to the host output struct. */
struct
{
    short     emp_name_ind;
    short     sal_ind;
    short     comm_ind;
} emprec_ind;

/*  Input host variable. */
int         emp_number;

int         total_queried;

/* Include the SQL Communications Area.
   You can use #include or EXEC SQL INCLUDE. */
#include <sqlca.h>

/* Declare error handling function. */
void sql_error(msg)
    char *msg;
```

```
{
    char err_msg[128];
    size_t buf_len, msg_len;

    /* EXEC SQL WHENEVER SQLERROR CONTINUE; */


    printf("\n%s\n", msg);
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    /* EXEC SQL ROLLBACK RELEASE; */

{
    struct sqlexd sqlstm;
    sqlstm.sqlvsn = 10;
    sqlstm.arrsiz = 0;
    sqlstm.sqladtp = &sqladt;
    sqlstm.sqltdsp = &sqltds;
    sqlstm.iters = (unsigned int  )1;
    sqlstm.offset = (unsigned int  )5;
    sqlstm.cud = sqlcud0;
    sqlstm.sqlest = (unsigned char  *)&sqlca;
    sqlstm.sqlety = (unsigned short)256;
    sqlstm.occurs = (unsigned int  )0;
    sqlcxt((void **)0, &sqlctx, &sqlstm, &sqlfpn);
}


    exit(EXIT_FAILURE);
}

void main()
{
    char temp_char[32];

/* Connect to ORACLE--
 * Copy the username into the VARCHAR.
 */
    strncpy((char *) username.arr, "SCOTT", UNAME_LEN);

/* Set the length component of the VARCHAR. */
    username.len =
      (unsigned short) strlen((char *) username.arr);

/* Copy the password. */
    strncpy((char *) password.arr, "TIGER", PWD_LEN);
    password.len =
      (unsigned short) strlen((char *) password.arr);

/* Register sql_error() as the error handler. */
    /* EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--\n"); */


/* Connect to ORACLE.  Program will call sql_error()
 * if an error occurs when connecting to the default database.
 */
```

```
    /* EXEC SQL CONNECT :username IDENTIFIED BY :password; */

{
    struct sqlexd sqlstm;
    sqlstm.sqlvsn = 10;
    sqlstm.arrsiz = 4;
    sqlstm.sqladtp = &sqladt;
    sqlstm.sqltdsp = &sqltds;
    sqlstm.iters = (unsigned int  )10;
    sqlstm.offset = (unsigned int  )20;
    sqlstm.cud = sqlcud0;
    sqlstm.sqlest = (unsigned char  *)&sqlca;
    sqlstm.sqlety = (unsigned short)256;
    sqlstm.occurs = (unsigned int  )0;
    sqlstm.sqhstv[0] = (         void  *)&username;
    sqlstm.sqhstl[0] = (unsigned int  )22;
    sqlstm.sqhsts[0] = (         int  )22;
    sqlstm.sqindv[0] = (         void  *)0;
    sqlstm.sqinds[0] = (         int  )0;
    sqlstm.sqharm[0] = (unsigned int  )0;
    sqlstm.sqadto[0] = (unsigned short )0;
    sqlstm.sqtdso[0] = (unsigned short )0;
    sqlstm.sqhstv[1] = (         void  *)&password;
    sqlstm.sqhstl[1] = (unsigned int  )42;
    sqlstm.sqhsts[1] = (         int  )42;
    sqlstm.sqindv[1] = (         void  *)0;
    sqlstm.sqinds[1] = (         int  )0;
    sqlstm.sqharm[1] = (unsigned int  )0;
    sqlstm.sqadto[1] = (unsigned short )0;
    sqlstm.sqtdso[1] = (unsigned short )0;
    sqlstm.sqphsv = sqlstm.sqhstv;
    sqlstm.sqphsl = sqlstm.sqhstl;
    sqlstm.sqphss = sqlstm.sqhsts;
    sqlstm.sqpind = sqlstm.sqindv;
    sqlstm.sqpins = sqlstm.sqinds;
    sqlstm.sqparm = sqlstm.sqharm;
    sqlstm.sqparc = sqlstm.sqharc;
    sqlstm.sqpadto = sqlstm.sqadto;
    sqlstm.sqptdso = sqlstm.sqtdso;
    sqlcxt((void **)0, &sqlctx, &sqlstm, &sqlfpn);
    if (sqlca.sqlcode < 0) sql_error("ORACLE error--\n");
}



    printf("\nConnected to ORACLE as user: %s\n", username.arr);

/* Loop, selecting individual employee's results */

    total_queried = 0;

    for (;;)
    {
        emp_number = 0;
        printf("\nEnter employee number (0 to quit): ");
        gets(temp_char);
        emp_number = atoi(temp_char);
        if (emp_number == 0)
```

```
            break;

/* Branch to the notfound label when the
 * 1403 ("No data found") condition occurs.
 */
        /* EXEC SQL WHENEVER NOT FOUND GOTO notfound; */


        /* EXEC SQL SELECT ename, sal, comm
            INTO :emprec INDICATOR :emprec_ind
            FROM EMP
            WHERE EMPNO = :emp_number; */

{
        struct sqlexd sqlstm;
        sqlstm.sqlvsn = 10;
        sqlstm.arrsiz = 4;
        sqlstm.sqladtp = &sqladt;
        sqlstm.sqltdsp = &sqltds;
        sqlstm.stmt = "select ename ,sal ,comm into :s1:s2 ,:s3:s4 ,:s5:s6  \
 from EMP where EMPNO=:b2";
        sqlstm.iters = (unsigned int  )1;
        sqlstm.offset = (unsigned int  )51;
        sqlstm.selerr = (unsigned short)1;
        sqlstm.cud = sqlcud0;
        sqlstm.sqlest = (unsigned char  *)&sqlca;
        sqlstm.sqlety = (unsigned short)256;
        sqlstm.occurs = (unsigned int  )0;
        sqlstm.sqhstv[0] = (         void  *)&emprec.emp_name;
        sqlstm.sqhstl[0] = (unsigned int  )22;
        sqlstm.sqhsts[0] = (         int  )0;
        sqlstm.sqindv[0] = (         void  *)&emprec_ind.emp_name_ind;
        sqlstm.sqinds[0] = (         int  )0;
        sqlstm.sqharm[0] = (unsigned int  )0;
        sqlstm.sqadto[0] = (unsigned short )0;
        sqlstm.sqtdso[0] = (unsigned short )0;
        sqlstm.sqhstv[1] = (         void  *)&emprec.salary;
        sqlstm.sqhstl[1] = (unsigned int  )sizeof(float);
        sqlstm.sqhsts[1] = (         int  )0;
        sqlstm.sqindv[1] = (         void  *)&emprec_ind.sal_ind;
        sqlstm.sqinds[1] = (         int  )0;
        sqlstm.sqharm[1] = (unsigned int  )0;
        sqlstm.sqadto[1] = (unsigned short )0;
        sqlstm.sqtdso[1] = (unsigned short )0;
        sqlstm.sqhstv[2] = (         void  *)&emprec.commission;
        sqlstm.sqhstl[2] = (unsigned int  )sizeof(float);
        sqlstm.sqhsts[2] = (         int  )0;
        sqlstm.sqindv[2] = (         void  *)&emprec_ind.comm_ind;
        sqlstm.sqinds[2] = (         int  )0;
        sqlstm.sqharm[2] = (unsigned int  )0;
        sqlstm.sqadto[2] = (unsigned short )0;
        sqlstm.sqtdso[2] = (unsigned short )0;
        sqlstm.sqhstv[3] = (         void  *)&emp_number;
        sqlstm.sqhstl[3] = (unsigned int  )sizeof(int);
        sqlstm.sqhsts[3] = (         int  )0;
        sqlstm.sqindv[3] = (         void  *)0;
        sqlstm.sqinds[3] = (         int  )0;
        sqlstm.sqharm[3] = (unsigned int  )0;
```

```
        sqlstm.sqadto[3] = (unsigned short )0;
        sqlstm.sqtdso[3] = (unsigned short )0;
        sqlstm.sqphsv = sqlstm.sqhstv;
        sqlstm.sqphsl = sqlstm.sqhstl;
        sqlstm.sqphss = sqlstm.sqhsts;
        sqlstm.sqpind = sqlstm.sqindv;
        sqlstm.sqpins = sqlstm.sqinds;
        sqlstm.sqparm = sqlstm.sqharm;
        sqlstm.sqparc = sqlstm.sqharc;
        sqlstm.sqpadto = sqlstm.sqadto;
        sqlstm.sqptdso = sqlstm.sqtdso;
        sqlcxt((void **)0, &sqlctx, &sqlstm, &sqlfpn);
        if (sqlca.sqlcode == 1403) goto notfound;
        if (sqlca.sqlcode < 0) sql_error("ORACLE error--\n");
}



/* Print data. */
        printf("\n\nEmployee   Salary    Commission\n");
        printf("--------   -------   ----------\n");

/* Null-terminate the output string data. */
        emprec.emp_name.arr[emprec.emp_name.len] = '\0';
        printf("%s      %7.2f      ",
            emprec.emp_name.arr, emprec.salary);

        if (emprec_ind.comm_ind == -1)
            printf("NULL\n");
        else
            printf("%7.2f\n", emprec.commission);

        total_queried++;
        continue;

notfound:
        printf("\nNot a valid employee number - try again.\n");

    } /* end for(;;) */

    printf("\n\nTotal rows returned was %d.\n", total_queried);
    printf("\nG'day.\n\n\n");

/* Disconnect from ORACLE. */
    /* EXEC SQL ROLLBACK WORK RELEASE; */

{
    struct sqlexd sqlstm;
    sqlstm.sqlvsn = 10;
    sqlstm.arrsiz = 4;
    sqlstm.sqladtp = &sqladt;
    sqlstm.sqltdsp = &sqltds;
    sqlstm.iters = (unsigned int  )1;
    sqlstm.offset = (unsigned int  )82;
    sqlstm.cud = sqlcud0;
    sqlstm.sqlest = (unsigned char  *)&sqlca;
    sqlstm.sqlety = (unsigned short)256;
    sqlstm.occurs = (unsigned int  )0;
```

```
    sqlcxt((void **)0, &sqlctx, &sqlstm, &sqlfpn);
    if (sqlca.sqlcode < 0) sql_error("ORACLE error--\n");
}


    exit(EXIT_SUCCESS);
```

Chapter 4. Executing Oracle utilities from a shell script

With the release of OSDI, many of the Oracle utilities are now executable from UNIX System Services. These include:

- SQL*Plus (available in OS/390 UNIX System Services at Oracle 8.0.4)
- SQL*Loader
- Import
- Export
- Pro C Precompiler
- Wrap

Shell scripts previously written for a UNIX environment should run in OS/390 UNIX System Services without requiring any changes to the shell script. At most, minor changes may need to be made for functions that are specific to the brand of UNIX originally used when the shell script was developed.

4.1 The shell scripts

Two shell scripts were written for this redbook research process. One shell script executes SQL*Plus scripts to create four tablespaces and a partitioned table. The other shell script executes SQL*Loader to load the table with data and run a test query against the table created.

4.1.1 Creating the tablespaces

The first script executes five SQL*Plus scripts. The SQL*Plus scripts create four tablespaces to be used for a partitioned table, and a partitioned table. Following is an example of one of the SQL*Plus scripts used to create one of the tablespaces. The other three are similar. We also created log files in order to review the results of the scripts.

```
-- create_pts1.sql
-- create the tablespace for partitioned table

DROP TABLESPACE callup_p1;

CREATE TABLESPACE callup_p1
DATAFILE  '//''osd1.dbf.callp1.db1'''
REUSE
DEFAULT STORAGE (INITIAL 500K NEXT 100K);
```

*Figure 3. SQL*Plus script to create a tablespace*

Dataset naming conventions have changed a bit for Oracle's OSDI. Since they now use LE370, dataset naming conventions follow conventions used in LE370. There are also some changes in the way Oracle interprets SQL commands. Note the line:

```
DATAFILE  '//''osd1.dbf.callp1.db1'''
```

One convention used to name Datasets is `//'dsname' to name datasets`. DDL command processing requires that the data file name be enclosed in single

quotes. The Oracle command processor will parse the two single quotes as a single quote. There must be no spaces within the single quotes.

The database files were pre-created. See Figure 12 on page 39 for a copy of the JCL used for this test. When the data files are pre-created with IDCAMS, the parameter `REUSE` must be used or the DDL command will fail with an `ORA-04101` message code. This is different from the MPM version in that the MPM version would accept the command and would be able to locate the dataset. If the database file was pre-created, then you must not use a `SIZE` parameter or the DDL will also fail with an `ORA-04101` message.

A tablespace was created for each partition of the table. The table was created using the following SQL*Plus script:

```
-- crt_table_part
-- create the table for the partitioned tablespace

DROP TABLE callup_pt;

CREATE TABLE callup_pt (
  pri_key     NUMBER
  CONSTRAINT pk_callup_pt PRIMARY KEY,
  lname       VARCHAR(30),
  fname       VARCHAR(20),
  phone       VARCHAR(10),
  employ_type VARCHAR(5),
  dept        VARCHAR(5)
)
STORAGE (INITIAL 200K NEXT 100K)
PARTITION BY RANGE (lname)
(
  PARTITION a_to_f VALUES LESS THAN ('G')
    TABLESPACE callup_p1,
  PARTITION g_to_l VALUES LESS THAN ('M')
    TABLESPACE callup_p2,
   PARTITION m_to_r VALUES LESS THAN ('S')
    TABLESPACE callup_p3,
   PARTITION s_to_z VALUES LESS THAN (MAXVALUE)
    TABLESPACE callup_p4
);
```

*Figure 4. SQL*Plus to create partitioned table*

To execute the four scripts that build the four tablespaces and the script to build the table from a single SQL*Plus script, the following SQL*Plus script was created.

```
-- create_part_stuff.sql
-- create the tablespaces and table

spool /u/ibmu10/oracle/logs/build_pt.lst

@@/$ORACLE_HOME/bin/create_pts1.sql
@@/$ORACLE_HOME/bin/create_pts2.sql
@@/$ORACLE_HOME/bin/create_pts3.sql
@@/$ORACLE_HOME/bin/create_pts4.sql
@@/$ORACLE_HOME/bin/create_part_table.sql

spool off
```

*Figure 5. SQL*Plus script to execute SQL*Plus scripts*

Note that spooling was enabled in the SQL*Plus script to provide a log in HSF of the results of the DDL commands.

The OS/390 UNIX System Services shell script was then created. The script is a very simple one, intended to demonstrate the ability to run the Oracle SQL*Plus utility from the UNIX shell. The following is the shell script that we used:

```
#!/bin/sh
# This script will build the partitioned table space and perform
# a query to check things out
#
# get the oracle sid,userid and password
#
   echo ' '
   echo '-----------------------------------------------------------'
   echo 'Enter the database name'
   read db_name
   echo 'enter the userid'
   read o_uid
   echo 'enter the password'
   read  o_pwd
   echo ' '
   echo '-----------------------------------------------------------'

   echo ' '
   echo '-----------------------------------------------------------'
   echo 'Building the tablespaces and table.'
   echo '-----------------------------------------------------------'
   echo ' '

   sqlplus $o_uid/$o_pwd@$db_name @$ORACLE_HOME/bin/create_part_stuff.sql

   echo '-----------------------------------------------------------'
   echo ''
   echo 'All done...'
   echo ''
   echo '-----------------------------------------------------------'
```

Figure 6. Shell script used to build the tablespaces and partitioned table

As stated earlier, this is a simple script and does not provide for input checking, validity of the directories and files, or if the files are executable. This script was run successfully. The `build_pt.lst` was created and indicated that the operation

was successful and all the DDL executed successfully. Figure 7 shows the log file that was created.

```
drop tablespace callup_p1

ERROR at line 1:
ORA-00959: tablespace 'CALLUP_P1' does not exist

Tablespace created.

drop tablespace callup_p2
*
ERROR at line 1:
ORA-00959: tablespace 'CALLUP_P2' does not exist

Tablespace created.

drop tablespace callup_p3
*
ERROR at line 1:
ORA-00959: tablespace 'CALLUP_P3' does not exist

Tablespace created.

drop tablespace callup_p4
*
ERROR at line 1:
ORA-00959: tablespace 'CALLUP_P4' does not exist

Tablespace created.

DROP TABLE callup_pt
           *
ERROR at line 1:
ORA-00942: table or view does not exist

Table Created
```

Figure 7. Results of the build_pt.sh shell script

4.1.2 Shell script to load and test the table

The four tablespaces and the partitioned table are now built. It is now time to load the table with data. The data was taken from the Poughkeepsie phone directory. Twenty-six files were created (`A DATA A1` through `Z DATA Z1`) on VM. A REXX script was used to combine all the files into a single file and prepare the data for loading into Oracle. A copy of the REXX exec is provided in Figure 13 on page 40**.** The file:

CALLUP DATA A1

was FTP'd to the OS/390 system as the sequential dataset:

IBMU10.CALLUP.DATA

With the dataset in place in OS/390, a UNIX shell script was created to move the sequential dataset by using the TSO/e command `oput` to move the data file to a directory in HSF. The *control* file was created for SQL*Loader and placed in the directory /u/ibm10/oracle/progs. The control file contains the location of the *discard* file. The discard file contains invalid records. The control file also contains the commands needed by SQL*Loader to execute the load.

The following `control` file was used:

```
LOAD DATA
INFILE '/u/ibmu10/oracle/data/callup.dat'
DISCARDFILE '/u/ibmu10/oracle/logs/loader.dsc'
INSERT INTO TABLE callup_pt
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY'"'
   (PRI_KEY SEQUENCE(1,1),
    LNAME,FNAME,PHONE, EMPLOY_TYPE,DEPT)
```

*Figure 8. SQL*Loader control file*

A *bad* file was identified in the shell script. The bad file contains the records that were not inserted because they did not meet the specifications of column or were syntactically invalid.

As stated earlier, the shell script is very simple. There is not extensive error checking. Its sole purpose is to demonstrate that the utility can be run from the OS/390 UNIX System Services shell. We chose to move the data file from a sequential dataset to HSF. This is not a requirement. By following proper file naming conventions, the sequential dataset name could have been provided to the shell script and passed to the `sqlldr` utility. This is clearly documented in the OSDI for OS/390 Users Guide. The shell script we used is provided in Figure 9.

```
#!/bin/sh
# This script will copy a member of a PDS to the HFS
#
   echo '-------------------------------------------------------'
   echo 'This program will move a dataset from MVS to USS. The     '
   echo 'dataset contains data to be used by the SQL*Loader Program.'
   echo 'You will be asked for the DSN and HSF locations.         '
   echo 'You will also be asked for information regarding the    '
   echo 'database to run the load against. The control file has   '
   echo 'already been created and is /u/ibmu10/oracle/progs.      '
   echo '-------------------------------------------------------'
   echo ' '
#
   echo ' '
   echo '-------------------------------------------------------'
   echo 'Enter the dataset name to move to HSF.'
   echo '-------------------------------------------------------'
   echo ' '
   read  mvsfile

# insure that the input is not null

   if [ $mvsfile ]
   then
      echo ' '
      echo '-------------------------------------------------------'
      echo 'File ' $mvsfile ' ok'
      echo '-------------------------------------------------------'
      echo ' '
   else
      echo ' '
      echo '-------------------------------------------------------'
      echo 'Dataset name not entered.'
      echo 'Exiting.'
      echo '-------------------------------------------------------'
      echo ' '
      exit
   fi
  ussfile='/u/ibmu10/oracle/data/callup.dat'

# move the file from mvs to hfs
   echo 'Copying' $mvsfile 'to' $ussfile
   echo ' '
   tso "oput" "'$mvsfile'" "'$ussfile'"
   echo ' '

#
# get the oracle sid,userid and password
#
   echo '-------------------------------------------------------'
   echo ''
   echo 'Enter the following information in order to run SQL*Loader.'
   echo ''
   echo '-------------------------------------------------------'
   echo ''
   echo 'Enter the database SID'
   read db_name
   echo 'Enter the userid'
   read o_uid
   echo 'Enter the password'
```

```
read o_pwd
   echo 'Enter the control file path and name'
   read contfile
   echo 'Enter the log file path and name'
   read logfile

#  Execute sqlloader to load the callup_pt table
   badfile='/u/ibmu10/oracle/logs/loader.bad'

   echo '-------------------------------------------------------'
   echo ''
   echo 'Executing SQL*Loader'
   echo ''
   echo '-------------------------------------------------------'
   echo ''

   sqlldr $o_uid/$o_pwd@$db_name control=$contfile log=$logfile bad=$badfile

   echo '-------------------------------------------------------'
   echo ''
   echo 'Executing test query. Check log in /u/ibmu10/oracle/log'
   echo 'for results.'
   echo ''
   echo '-------------------------------------------------------'
   echo ''

   sqlplus $o_uid/$o_pwd@$db_name @/orauss/bin/query_pt_table.sql

   echo 'All done...'
```

*Figure 9. Shell script to execute SQL*Loader*

The results of `sqlldr` were placed in the log file we identified during the execution of the shell script. The file was /u/ibmu10/oracle/logs/loader.log. The contents of the loader.log provides all the information a DBA would need to know about the load operation, including why records were not inserted into the table. For this run of the `load.sh` script, the loader.log contained the information shown in Figure 10 on page 36.

```
SQL*Loader: Release 8.1.7.0.50 - Production on Thu Dec 7 07:05:04 2000

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Control File:   /u/ibmu10/oracle/progs/ld_part.ctl
Data File:      /u/ibmu10/oracle/data/callup.dat
  Bad File:     /u/ibmu10/oracle/logs/loader.bad
  Discard File: /u/ibmu10/oracle/logs/loader.dsc
 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 65536 bytes
Continuation:    none specified
Path used:      Conventional

Table CALLUP_PT, loaded from every logical record.
Insert option in effect for this table: INSERT

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
PRI_KEY                                                    SEQUENCE (1, 1)
LNAME                              FIRST     *   ,  O(") CHARACTER
FNAME                               NEXT     *   ,  O(") CHARACTER
PHONE                               NEXT     *   ,  O(") CHARACTER
EMPLOY_TYPE                         NEXT     *   ,  O(") CHARACTER
DEPT                                NEXT     *   ,  O(") CHARACTER

Record 300: Rejected - Error on table CALLUP_PT.
ORA-01401: inserted value too large for column

Record 618: Rejected - Error on table CALLUP_PT.
ORA-01401: inserted value too large for column

-----------------------Lines Deleted-----------------------------------------
Record 8266: Rejected - Error on table CALLUP_PT, column PHONE.
Column not found before end of logical record (use TRAILING NULLCOLS)

Table CALLUP_PT:
  8249 Rows successfully loaded.
  17 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  64337 bytes(49 rows)
Space allocated for memory besides bind array:        0 bytes

Total logical records skipped:          0
Total logical records read:          8266
Total logical records rejected:        17
Total logical records discarded:        0

Run began on Thu Dec 07 07:05:04 2000
Run ended on Thu Dec 07 07:05:07 2000

Elapsed time was:     00:00:02.55
CPU time was:         00:00:00.90
```

*Figure 10. Results of the SQL*Loader operation*

You can see that there were no discards, but there were 17 rejected records. The total time it took to execute sqlldr is also provided in CPU time and in elapsed time.

To view the rejected records, one only needs to look in the bad file we identified, which is:

/u/ibmu10/oracle/logs/loader.bad

While the contents of the loader.log tells us what the problem was with the rejects, the loader.bad actually shows us the invalid records. The loader.bad contained the records shown in Figure 11.

```
"BARNES", "Kathryn",  "000-0000",  "ASSGN-IN",  "007 "
"BOTTEGO", "Kathryn",  "*",  "293-5009",  "1-91"
"BRAREN", "Denny",  "*",  "293-3680",  "1-91"
"GANGADHAR", "Jeff",  "R*",  "293-5308",  "710 "
"KOWALSKI", "Denny",  "*",  "293-5629",  "1-91"
"MCTAGGART STONE", "Jeff",  "*",  "295-8111",  "MAIL"
"MENDEL", "Denny",  "000-0000",  "MGR(SUP)",  "R2ZA"
"MULE", "Denny",  "*",  "295-4020",  "1-91"
"ROBERTS", "Jeff",  "*",  "292-8483",  "NA  "
"SHANMUGASUNDARAM", "Lou",  "*",  "293-3618",  "1-91"
"SUMNER", "Jeff",  "*",  "000-0000",  "1   "
"THOMAS", "Jeff",  "*",  "295-7274",  "1-91"
"THOMPSON", "Kathryn",  "*",  "000-0000",  "1-51"
"THYAGARAJAN", "Denny",  "*",  "293-9119",  "0-91"
"WILLIAMS", "Denny",  "R*",  "293-9727",  "1-91"
"WINKELBAUER", "Denny",  "*",  "295-5652",  "1-91"
Zwolak, Mary Ann *VENDOR*          295-3050 Vend  IBMUSM10 MZWOLAK  P317
```

*Figure 11. Rejected records from the SQL*Loader utility*

You should also notice that we ran another SQL*Plus script against the newly loaded table to prove to ourselves that the table did, in fact, load correctly. We ran the script `query_part_table.sql` and viewed the results. The query we executed was:

```
SELECT lname, fname, phone
FROM   callup_pt
WHERE  lname like 'DUT%';
```

The results of the query were recorded in a file by using the spool command in the SQL*Plus script. The results of the query were:

```
LNAME                          FNAME                PHONE
------------------------------ -------------------- ----------
DUTCAVICH                      Jeff                 293-7926
DUTCAVICH                      Kathryn              293-3183
```

This demonstrated that in a single shell script run in OS/390 UNIX System Services we were able to successfully move a dataset from TSO to HSF, execute SQL*Loader and run a query against the newly loaded table.

4.2 Summary

With the announcement of OSDI, the most common utilities used by DBAs to support the Oracle database have been move to OS/390 UNIX System Services. This then helps leverage the Oracle and UNIX skills already in place if the database is moved to the mainframe. The utilities run exactly the same as they would if they were executed from any UNIX platform. The added benefit is that if the data resides on the mainframe as datasets, which may often be the case since data is sourced from a mainframe application, the data does not need to be moved to another platform. In fact, it does not have to be moved from a dataset to an HFS file; the dataset name can be used in a OS/390 UNIX System Services shell script and Oracle will be able to handle it.

4.3 Sample JCL and programs

```
//MAKE01 JOB (0000,OR),'ORACLE INSTALL',CLASS=A,
//           MSGCLASS=X,PRTY=15,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
//*
//*
//* LIBRARY ===> IBMU10.JCL.CNTL
//* MEMBER  ===> CRTFILES
//* AUTHOR  ===> ICC
//*-----------------------------------------------------------------*
//*  JOB DESCRIPTION: DELETE AND ALLOCATE THE  ORACLE/MVS DATABASE,*
//*                   VSAM FILES                                   *
//*-----------------------------------------------------------------*
//*
//STEPCRT1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 /*-----------------------------------------------------*/
 /*                                                     */
 /*       DELETE THE ORACLE DB, CONTROL, AND LOG        */
 /*       CLUSTERS AND THE ALERT LOG FILE.              */
 /*                                                     */
 /*-----------------------------------------------------*/
   DELETE (OSD1.DBF.CALLP1.DB1) -
          CLUSTER PURGE

   SET MAXCC=0    /* REMOVE AFTER FIRST TIME EXECUTION */
   SET LASTCC=0   /* REMOVE AFTER FIRST TIME EXECUTION */
 /*-----------------------------------------------------*/
 /*                                                     */
 /*       DEFINE THE DB (ORACLE DATA BASE) CLUSTERS.    */
 /*                                                     */
 /*-----------------------------------------------------*/
   DEFINE CLUSTER                                  -
     ( LINEAR                                      -
      NAME(OSD1.DBF.CALLP1.DB1)                 -
          VOLUMES(ORA030 )                         -
          MEGABYTES(100)                           -
     )
 /*-------------------------------------------------------*/
 /*                                                       */
 /*       LIST THE CATALOG CLUSTER ENTRY                  */
 /*                                                       */
 /*-------------------------------------------------------*/
   LISTC   ENT(OSD1.DBF.CALLP1.DB1) ALL
//*
```

Figure 12. JCL Example to pre-create database files

```
/* rexx exec to create a datafile to use to load an Oracle table */
trace o
SAY 'Building load file for SQL*Loader'
'EXECIO 0 diskw CALLUP DATA A1 '

n=1
files='ABCDEFGHIJKLMNOPQRSTUVWXYZ'

DO LENGTH(files)
   'EXECIO  * diskr' SUBSTR(files,n,1) 'TEXT A1 1 ( fifo )'
   records =  QUEUED()
   rr_name=records
   DO records-1
      PULL the_record
         the_comma=POS(',',the_record)
         lname=SUBSTR(the_record,1,the_comma-1)
         lname='"'||lname||'"'

         /* take care of first name */
         fname=SUBSTR(the_record,the_comma+2,38-the_comma)
         fname=STRIP(fname,'T')
         fname='"'||fname||'"'

         /* reparse the_record to get rid of the first names */
         len=LENGTH(the_record)
         the_record=SUBSTR(the_record,38,len)
         PARSE VAR the_record phone employ_type . . . . dept .

         /* phone is a fixed field */
         phone='"'||phone||'"'

         /* fixup type of employ_type */
         employ_type='"'||employ_type||'"'

         /* fix up dept */
         dept=SUBSTR(dept,1,4)
         dept='"'||dept||'"'

         input_data=lname||', 'fname||',  'phone||', 'employ_type||',  'dept'

      'EXECIO 1 diskw CALLUP DATA A1 (STR' input_data
   END  /*end do loop */
n=n+1
END
'EXECIO 1 diskw CALLUP DATA A1 ( FINIS )'
DROPBUF
SAY 'Ending program....'
exit
```

Figure 13. REXX EXEC used to create callup data

Chapter 5. Recovery Manager on S/390

This chapter conveys the basic concepts and operational runtime characteristics of Recovery Manager on S/390 (MPM), essentially the base functionality. It should not be viewed as a comprehensive guide to Recovery Manager and all of its features and options. Rather, it is intended to give the user a sense of what Recovery Manager looks like when implemented and run within the context of a traditional S/390 operational setting. For additional, detailed information on the generic capabilities of Recovery Manager, consult the *Oracle8i Backup and Recovery Guide*.

5.1 Overview of Recovery Manager

Several terms have very specific meanings when used in relation to Oracle and Recovery Manager. In this discussion, the following terms are used:

Target database An Oracle database which is the source of a backup or recovery operation.

Recovery catalog Describes a series of Oracle database tables and views which carry backup and recovery information for one or more target Oracle databases.

Auxiliary database A special purpose Oracle instance which participates in TABLESPACE point-in-time recovery operations. In earlier releases of Recovery Manager, the auxiliary database was referred to as the clone database.

Recovery Manager (RMAN) is an Oracle tool that allows you to back up, copy, restore, and recover data files (individual files such as VSAM files which carry all or part of the data of an Oracle TABLESPACE on S/390), control files, and archived redo logs. You can invoke RMAN as a command line utility from the O/S prompt (that is, from native TSO) or by way of a batch process (JCL). You can also use the GUI-based Oracle Enterprise Manager Backup Manager; however, this section does not discuss how to invoke and run RMAN either from the command line or via Oracle Enterprise Manager. The illustrations and discussions in this section are confined exclusively to its invocation via JCL.

RMAN automates many of the backup and recovery tasks that were formerly performed manually. For example, instead of requiring you to locate the appropriate backup files for each data file, copy them to the correct place using O/S commands, and choose which archived logs to apply—RMAN manages these tasks automatically.

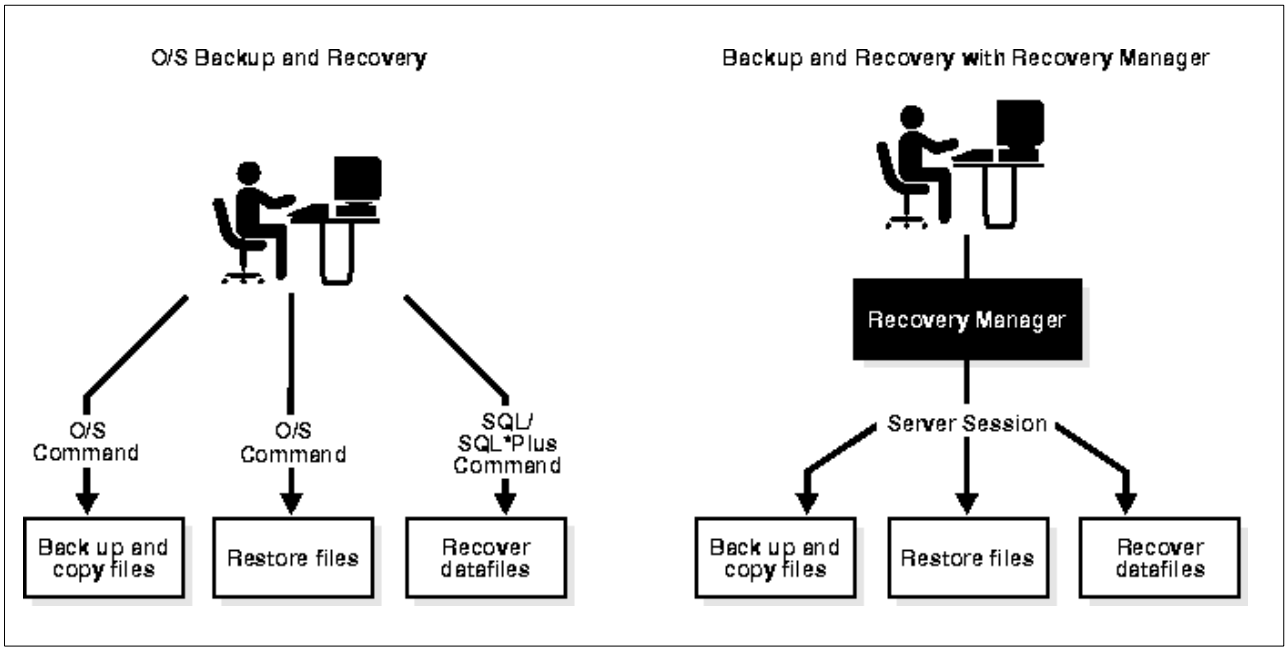

Figure 14. Backup and recovery options

Figure 15 depicts Recovery Manager in relation to the database files it operates upon and the types of backup media it can create. It also shows the Oracle Enterprise Manager as a component that can be used to communicate with and drive Recovery Manager; however, this section does not discuss the Oracle Enterprise Manager and Recovery Manager relationship or operation.

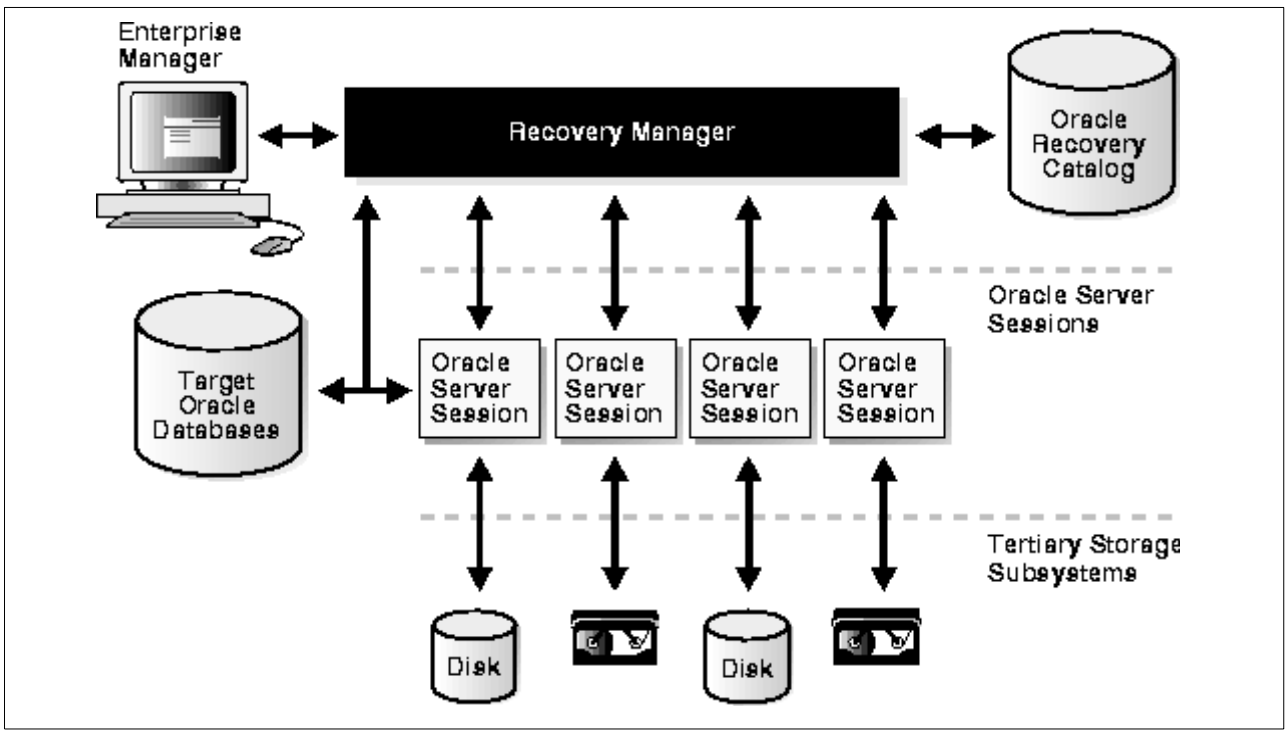

Figure 15. Recovery Manager

5.2 Recovery Manager features

RMAN automates important backup and recovery procedures. For example, Recovery Manager is able to:

- Back up and copy the database, TABLESPACEs, datafiles, control files, and archived redo logs
- Compress backups of data files so that only those data blocks that have been written to are included in a backup
- Store frequently executed backup and recovery operations in scripts
- Perform incremental backups, which back up only those data blocks that have changed since a previous backup
- Create a duplicate of your production database for testing purposes
- Use third-party media management software
- Generate a printable message log of all backup and recovery operations
- Use the Recovery Catalog to automate both restore and recovery operations
- Perform automatic parallelization of backup and restore operations
- Restore a backup using a backup control file and automatically adjust the control file to reflect the structure of the restored datafiles
- Find datafiles that require a backup based on user-specified limits on the amount of redo that must be applied for recovery
- Perform crosschecks to determine whether archived materials in the media management catalog are still available
- Test whether specified backups can be restored

Several points are worth mentioning here:

- At the time of this writing, Recovery Manager is only compatible with Oracle databases of release 8.0 or higher. It is not compatible with the Oracle7 Enterprise Backup Manager (EBU).
- Also, new in Oracle8i (8.1 or higher), the Recovery Catalog is no longer required; the use of a Recovery Catalog is optional. If you choose not to use a Recovery Catalog, you can still use RMAN very effectively. RMAN obtains the information it needs from the control file of the target database. When using a Recovery Catalog, RMAN can perform a wider variety of automated backup and recovery functions. For this reason, Oracle recommends using a Recovery Catalog with RMAN whenever possible. The examples illustrated herein make use of the Recovery Catalog at all times.
- Additionally, a new enhancement, called Proxy Copy, has been introduced in Oracle8i (8.1). This enhancement enables media management vendor software to take over the entire data movement involved in a backup or restore. RMAN merely provides a list of files requiring backup or recovery to the media manager, which in turn makes all decisions regarding how and when to move the data. The PROXY option of the BACKUP command invokes this facility. The facility must be specifically supported by the media management software—not all media management software provides this support. As of this writing, DFSMS/DFDss is the only supported Proxy Copy option on S/390.

5.3 What Recovery Manager is not

RMAN is *not*:

- The only option for performing backup and recovery operations. You can also back up and restore files using O/S commands, and use SQL/SQL*Plus statements to recover them.
- A Recovery Catalog. A Recovery Catalog is an optional schema containing information about RMAN operations. Note also that RMAN does not store backups or copies in the Recovery Catalog; rather, it stores information about backups and copies in the Recovery Catalog.
- Compatible with Oracle databases before release 8.0.
- The Enterprise Backup Utility (EBU), which is an Oracle7 utility. RMAN is not compatible with EBU.
- An O/S-specific backup utility. RMAN is a generic utility that can run on a number of different operating systems.

5.4 Implications of database state and RMAN

Recovery Manager allows you to control the type of “backups” (backups produced by either COPY or BACKUP commands) you produce. RMAN backups can be classified in three ways: full/incremental, open/closed and consistent/inconsistent. The latter two classifications apply to the state of a database's files at the time of a backup operation. The implications of various states of a database (or a particular data file) must be understood, as a practical matter, when using RMAN. However, because we are concerned with base functionality, we will be speaking in those terms without undue consideration or comment on the implications of state. For a complete discussion on the implications of “state” in backup and recovery scenarios, refer to the generic Oracle Backup and Recovery documentation.

5.5 Basic invocation of RMAN

When RMAN begins execution, the following events occur:

- RMAN creates two default server sessions that connect to the target database.

  If you perform I/O on disk or tape, RMAN requires that you allocate one channel for each disk or device. A channel corresponds to a server session.

  If you connect to a Recovery Catalog, RMAN creates a server session on the Recovery Catalog database.
- When it connects to your target database, RMAN uses server sessions to perform the backup and recovery operations through a PL/SQL interface. RMAN physically stores its backups and copies on disk or, if you use media management software (via the “proxy copy” option), on tape.
- RMAN stores metadata about its backup and recovery operations in the Recovery Catalog, which is a centralized repository of information, or exclusively in the control file. Typically, the Recovery Catalog is stored in a separate database. If you do not use a Recovery Catalog, RMAN uses the control file as its repository of metadata.

Remember, throughout this section we always employ a Recovery Catalog, even though it is optional.

Also, in all of our examples, we will use RCAT to designate the Recovery Catalog Oracle instance, TRGT to designate the target Oracle instance, and AUXI for the auxiliary Oracle instance.

5.6 Sample job to invoke Recovery Manager

```
//MCONNOLC JOB (0000,01),'RMAN',MSGCLASS=X,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
//*-----------------------------------------------------*
//*        Sample RMAN Invocation                       *
//*-----------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT        RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
RUN { RMAN command 1; RMAN command 2;...}
```

In the preceding sample job stream, note the SYSPARM input data, in particular the strings following the '@' sign. In addition to identifying the user id and password to be used when connecting to the target and Recovery Catalog instances, the information following the '@' sign conveys the method of communication RMAN will use. In this illustration (and those throughout this section), the “@W” designates XMS (cross memory services) as the mode of communication.

Alternatively, TCP/IP could be designated as the communication protocol between either (or both) the target and Recovery Catalog instances. Had we chosen to use TCP/IP, then some additional information would have been required in the JCL, in particular a TNSNAMES DD statement which pointed to a file containing “connect string” information.

For example, we could have specified “RCVCAT RMAN/RMAN@RCAT” and left the target's communication protocol as XMS:

```
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@RCAT
   CMDFILE "/DD/CMDFILE"
```

What this specification would have required is a TNSNAMES DD statement that referenced a file containing information identifying the “RCAT” (arbitrary protocol “service” identifier) and the particulars associated with its protocol. As an example, the TNSNAMES file pointed to might contain something like the following:

```
RCAT =
```

```
(DESCRIPTION=
 (ADDRESS_LIST=
  (ADDRESS= (SSN=TNS8) (PROTOCOL=TCP) (HOST=ATCIBM2)
  (PORT=1537)) )  )
```

In this example, the RCAT identifier is associated with a string of text that defines the protocol the RMAN server session is to use, as well as the specifics of that protocol. Something which might not be evident from this example is that the Recovery Catalog could be remote to the server where the target instance runs. In other words, the instance where the Recovery Catalog information is stored could be in an Oracle database instance anywhere on the customer's network.

For more information on Oracle server-to-server or client-to-server communication, consult the generic *Oracle Net8 User's Guide,* as well as the *Oracle System Administration* and *Installation Guides for Oracle Enterprise Edition for S/390.*

5.7 Basic setup of RMAN

When we think of RMAN, we should keep in mind that there are three components involved in the backup of any Oracle database file. If we are thinking in terms of a TABLESPACE “point-in-time” recovery, there is an additional component, called the *auxiliary database*, which we will discuss later. The first three components are the *target database* (the source to be backed up), *RMAN* itself (the utility) and the *Recovery Catalog*.

The basic steps involved in setting up RMAN are quite simple:

1. Identify the Oracle database where the Recovery Catalog is to be stored. Although the Recovery Catalog could be created within an Oracle database which itself was a target of some RMAN backup and recovery operation, this is not prudent or recommended. It is much better to identify a different Oracle database as the repository of target database backup and recovery information.
2. Create an RMAN user in the database where the Recovery Catalog is to be stored. This user (and its arbitrary ID) should be the user ID you plan to use whenever you connect to the Recovery Catalog to perform any Recovery Manager operation.
3. Create the Recovery Catalog itself with the Recovery Manager user ID created.
4. Register the target Oracle database(s) in the Recovery Catalog.

5.8 Creating RMAN user

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//          MSGCLASS=X,NOTIFY=&SYSUID
//*-------------------------------------------------------*
//*  JOB DESCRIPTION: CREATE A USER ID FOR RMAN           *
//*-------------------------------------------------------*
//ORADBA   EXEC PGM=SVRMGRL
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//         DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

```
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//ORA@RCAT DD DUMMY      <=== ORA@SSN (ORACLE SUBSYSTEM NAME).
//ORA$FNA  DD DSN=ORA8.ORA817.PARMLIB(FNA),DISP=SHR
//SQL      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSIN    DD *
SET ECHO ON
CONNECT INTERNAL
CREATE TABLESPACE RCVCAT
DATAFILE '/DSN/RCAT.DBF.RCVCAT.DB1'   SIZE 40M;
COMMIT;
CREATE USER RMAN IDENTIFIED BY RMAN TEMPORARY TABLESPACE TEMP
DEFAULT TABLESPACE RCVCAT QUOTA UNLIMITED ON RCVCAT;
GRANT RECOVERY_CATALOG_OWNER TO RMAN;
GRANT CONNECT, RESOURCE      TO RMAN;
/*
```

Note in our example, that by way of the ORA@RCAT DD DUMMY specification, we are connecting to our Recovery Catalog instance. Also, note in our example, that we create a TABLESPACE called RCVCAT and later associate the user ID which will be used when we invoke RMAN with that TABLESPACE. This is not a requirement, of course, but simply an arbitrary decision on our part. Likewise, we called our “RMAN” user ID RMAN, for an equally arbitrary reason. The GRANT of RECOVERY_CATALOG_OWNER is, however, a requirement. The final GRANT for CONNECT and RESOURCE is to permit basic access to instance.

5.9 Creating the Recovery Catalog

The next step we must accomplish is the creation and initialization of our Recovery Catalog. The RMAN Recovery Catalog must be created before any target databases can be registered. Notice, that we connect to the Recovery Catalog instance using our “RMAN” user ID. The syntax for creating the Recovery Catalog is very simple. For example:

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//* CREATE THE RECOVERY CATALOG                                     *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
 RCVCAT RMAN/RMAN@W:RCAT
 CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
CREATE CATALOG TABLESPACE 'RCVCAT';
```

This process can take a minute or two to run, but when it is complete the Recovery Catalog is ready for use.

5.10 Registering a target database in the Recovery Catalog

In order to register the target Oracle database(s) in the Recovery Catalog, RMAN must be invoked and a target database identified. For instance:

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//*-----------------------------------------------------------------*
//* REGISTER THE TARGET DATABASE IN THE RMAN DATABASE               *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
 TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
 CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
REGISTER DATABASE;
```

In our example, we are connecting to the target instance using the INTERAL specification in lieu of a normal user ID. The CHANGE_ON_INSTALL password supplied is simply ignored and the user ID under which the job itself is executed will be compared to the PRIVUSER list specified in the target instance's MPMPARM specification. This is not necessarily a bad practice, but one which you should consider carefully employing, given that in the OSDI version of Oracle for S/390 the PRIVUSER specification is obsolete and no longer exists.

The registration process can also take a few minutes to run, but when it is complete, all the information necessary about the target database(s) will have been stored in the Recovery Catalog. Once the Recovery Catalog has been constructed and one or more target databases registered within it, you can begin to work with RMAN.

5.11 Creating and using backup files

One of the most valuable aspects of RMAN is the ability to easily create “backups” of an Oracle database or parts of one (data files, archived redo logs, and control files), and have that information recorded automatically in a Recovery Catalog. Another equally important feature of the Recovery Catalog is the ability to query its contents. For instance, imagine how valuable it is to simply execute an RMAN report query and obtain backup information (files, dates, etc.) about a specific TABLESPACE or group of TABLESPACES.

However, before we consider the RMAN Recovery Catalog reporting features, we need to go over a few elementary concepts about backup and recovery within the context of RMAN. Even though concepts of backup and recovery are probably familiar to the reader, we need to discuss what is meant by two basic pairs of RMAN commands which are used for backup and recovery operations: COPY/RESTORE and BACKUP/RECOVER. These two pairs of commands can be used in conjunction with one another to produce many different schemes of backup and recovery, but for the purposes of simple illustration and explanation we will discuss them separately. Where necessary, however, we will mention

other commands which may have an overlapping or tandem requirement (for example, RECOVER following RESTORE) in a particular example.

5.12 COPY and RESTORE commands

The COPY command creates an “image copy” of a file and the output file is always written to disk. You can copy datafiles (current or copies), archived redo logs and control files. Again, datafiles are files such as VSAM files which carry the data of an Oracle TABLESPACE on S/390. Archived redo logs, which are simply sequential files, can also be copied. An Oracle instance's control files (also VSAM files) can be copied as well. It is important to note that a COPY operation results in an output file with the same physical attributes as the input file.

A copy of a single datafile, archived redo log file, or control file is usable as-is to perform recovery. In other words, an image copy can be used immediately by an Oracle instance; it is already in an instance-usable format. This is an important distinction, because the output of a BACKUP command operation is in a specific (and compressed) RMAN format and must therefore always be processed by RMAN. However, there are times when you should use RMAN when working with datafile COPYs.

For instance, you can use the RMAN COPY command to make copies of all the datafiles of a given target instance and then use those copies to “clone” another instance on another server (or on the same one, if you change the database name) without using RMAN. But you should use RMAN if you wish to use the datafile copies against the target instance from which they were created. This is because you will want the Recovery Catalog to reflect important information about the initial COPY operation, as well as subsequent RESTORE operations against the target, in order to reduce the possibility of error.

When you issue the COPY command, an Oracle server session—not an O/S routine—reads the file and writes the copy out to disk. Remember, by default you can only make image copies to disk. If you wanted to make copies to tape, you would have to use the Proxy Copy facility. Proxy copies are not image copies, but a type of media-managed backup generated with the “proxy” option of the backup command (discussed later). On S/390, for instance, RMAN could be instructed to use DFSMS/Copy to produce a copy of the datafile.

If you are already familiar with Oracle, you may be aware of the ALTER TABLESPACE {BEGIN/END} BACKUP commands. You do not need to precede an RMAN COPY command with the ALTER TABLESPACE BEGIN BACKUP statement. In fact, you should never put a TABLESPACE in “hot backup” mode (target TABLESPACE is online) with the ALTER TABLESPACE BEGIN BACKUP statement when using RMAN. RMAN uses an internal method to guarantee consistency in hot backups.

5.12.1 Sample COPY operation

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//* COPY DATAFILE                                                   *
//*-----------------------------------------------------------------*
```

```
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
RUN {
    ALLOCATE CHANNEL C1 TYPE DISK;
    COPY DATAFILE '/DSN/TRGT.DBF.TBS01'
         TO       '/DSN/TRGT.DBF.TBS01.COPY';
    RELEASE CHANNEL C1;
  }
```

In the preceding illustration, the COPY DATAFILE operation would result in a VSAM dataset which contains a “copy” of the 'TRGT.DBF.TBS01' VSAM dataset. It should be noted that in this example the TABLESPACE which is supported by the 'TRGT.DBF.TBS01' VSAM file was online and open for update during this operation. The implication for this COPY DATAFILE operation is that output VSAM file is not a duplicate, point in time image of the TABLESPACE.

If a duplicate, consistent point in time image of the TABLESPACE's VSAM dataset had been desired, then the TABLESPACE would have to have been taken offline before the COPY DATAFILE operation was executed. This could have been accomplished by preceding the COPY DATAFILE command with:

```
SQL 'ALTER TABLESPACE TBS01 OFFLINE';
```

Alternatively, for a consistent point in time image we could have mounted, but not opened, the target database prior to doing the COPY of any of its datafiles.

5.12.2 Sample RMAN messages from the COPY DATAFILE operation

```
Recovery Manager: Release 8.1.7.0.1 - Production
RMAN-06005: connected to target database: TRGT (DBID=2652807449)
RMAN-06008: connected to recovery catalog database
RMAN> 2> 3> 4> 5> 6> 7> 8> 9>
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: C1
RMAN-08500: channel C1: sid=12 devtype=DISK

RMAN-03022: compiling command: copy
RMAN-03024: performing implicit full resync of recovery catalog
RMAN-03023: executing command: full resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete
RMAN-03023: executing command: copy
RMAN-08000: channel C1: copied datafile 6
RMAN-08501: output filename=/DSN/TRGT.DBF.TBS01.COPY recid=1
stamp=413218736
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
```

Notice in the output of the RMAN messages from the COPY DATAFILE operation the RMAN-08501 informational message, in particular the “stamp=413218736”. This “stamp” identifier is unique to this copy of the datafile.

Restoring a data file is illustrated in the following sample.

5.12.3 Sample RESTORE operation

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//* RESTORE DATAFILE                                                *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
RUN {
    ALLOCATE CHANNEL C1 TYPE DISK;
    SQL 'ALTER TABLESPACE TBS01 OFFLINE';
    RESTORE DATAFILE '/DSN/TRGT.DBF.TBS01';
    RECOVER TABLESPACE TBS01;
    RELEASE CHANNEL C1;
  }
```

Because the TABLESPACE “TBS01” is presumably online in our example, it must first be taken offline before the RESTORE operation is attempted because the target instance will have TABLESPACE's underlying datafiles allocated and open. In this particular example, we are allowing RMAN to determine which image copy to use. By default, RESTORE DATAFILE causes RMAN to choose the most recent image copy (or backup set pieces) that needs the least media recovery. If the most recent image copy had been “TRGT.DBF.TBS01.COPY” then this file would have been chosen by RMAN to replace “TRGT.DBF.TBS01” in our example. Remember, the COPY command created a “usable as-is” image copy of the target datafile. However, an important point to consider in this example is the potential need for media recovery on the TABLESPACE which is supported by the newly restored datafile. Note the “RECOVER TABLESPACE TBS01” which follows the “RESTORE DATAFILE” statement in the preceding example.

Before a restored data file can be used, the TABLESPACE which it supports must be checked to determine whether or not “media recovery” is required.

Media recovery involves applying all applicable incremental backups and/or archived redo log data, as well as all currently applicable redo log data, against the TABLESPACE supported by the restored datafile. Because we cannot be certain about the time interval between the creation of image copy selected and

used by RMAN for the RESTORE operation, we should issue the RECOVER TABLESPACE command. The range of log data examined and applied (as needed) to the TABLESPACE during the RECOVER operation is from the creation time of the image copy datafile and forward to the current point in time for the instance.

In our example, after the RESTORE command completed, we issued the command:

```
RECOVER TABLESPACE TBS01;
```

This RECOVER operation would determine whether or not any redo log or archived redo log data needed to be applied to the TABLESPACE “TBS01”. If there were log data (current or archived) that needed to be applied, it would be done at this time.

5.12.4 Sample RMAN messages from RESTORE DATAFILE/RECOVER

```
Recovery Manager: Release 8.1.7.0.1 - Production

RMAN-06005: connected to target database: TRGT (DBID=2652807449)
RMAN-06008: connected to recovery catalog database

RMAN> 2> 3> 4> 5> 6> 7> 8> 9>
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: C1
RMAN-08500: channel C1: sid=12 devtype=DISK
RMAN-03022: compiling command: sql
RMAN-06162: sql statement: ALTER TABLESPACE TBS01 OFFLINE
RMAN-03023: executing command: sql

RMAN-03022: compiling command: restore
RMAN-03024: performing implicit full resync of recovery catalog
RMAN-03023: executing command: full resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete

RMAN-03022: compiling command: IRESTORE
RMAN-03023: executing command: IRESTORE
RMAN-08019: channel C1: restoring datafile 00006
RMAN-08507: input datafilecopy recid=1 stamp=413218736
filename=/DSN/TRGT.DBF.TBS01.COPY
RMAN-08509: destination for restore of datafile 00006: /DSN/TRGT.DBF.TBS01
RMAN-08007: channel C1: copied datafilecopy of datafile 00006
RMAN-08501: output filename=/DSN/TRGT.DBF.TBS01 recid=2 stamp=413218750
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: recover

RMAN-03022: compiling command: recover(1)

RMAN-03022: compiling command: recover(2)

RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
```

```
RMAN-08054: starting media recovery
RMAN-08055: media recovery complete

RMAN-03022: compiling command: recover(4)

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: C1
```

Notice in the output of the RMAN messages from the RESTORE DATAFILE operation the RMAN-08507 informational message. This message indicates which input datafile copy is being used during the RESTORE operation. You may remember the “stamp=413218736” identifier associated with the datafile copy created in the COPY DATAFILE sample above.

5.13 BACKUP and RECOVER commands

The corollary to the COPY/RESTORE command pair is BACKUP/RECOVERY. When you invoke a backup command, RMAN backs up the files into one or more backup sets. You can set parameters for the backup command to specify the filenames for the backup pieces, the number of files to go in each set, and which channel should operate on each input file. Remember, a channel is an Oracle server process under which an RMAN operation is performed. Files produced with the BACKUP command are in a compressed and special format which cannot be readily used by an Oracle instance and must be processed by RMAN before they can be accessed by the server. This is a very important distinction between the output of the COPY and BACKUP commands.

5.13.1 Full database backups

Although we are now discussing the BACKUP and RECOVER commands, this is a good time to make another important distinction about the use of COPY and BACKUP. Whether you use the COPY command to produce a consistent image copy of an entire database or you use the BACKUP command to do so, the decision is fundamentally one of how you plan to use the output files of those commands. Using COPY to create an “image copy” of a database can be faster in terms of elapsed time and cheaper in terms of CPU utilization, but it can be more cumbersome when it comes time to specify which datafiles are to be copied.

For instance, within the generic Oracle Backup and Recovery documentation, you will see that the COPY command format specification includes a “datafileSpec.” This datafile spec has two options. First, you can specific a specific file by name or you can specify the datafile by its numeric value. In order to obtain a datafile's assigned numeric value, you would have to invoke RMAN, either in batch or from a command prompt, and execute an RMAN query like:

```
report schema;
```

This RMAN query would produce a listing similar to the following:

```
RMAN-03022: compiling command: report
Report of database schema
File K-bytes    TABLESPACE           RB segs Name
---- --------- -------------------- ------- -------------------
1         47104 SYSTEM               YES     /TRGT.DB.SYSTEM.DB1
2          8000 SYSTEM               YES     /TRGT.DB.SYSTEM.DB2
```

```
3         23000 TBS01                NO      /TRGT.DB.TBS01.DB1
4         12000 USER                 NO      /TRGT.DB.USER.DB1
5          2000 TEMP                 NO      /TRGT.DB.TEMP.DB1
```

If you wanted to back up, for instance, the USER TABLESPACE and reference it by its file number, you would execute the RMAN COPY command in the following manner:

```
RUN {
     ALLOCATE CHANNEL CH1 TYPE DISK;
     COPY DATAFILE 4  TO '/DSN/TRGT.DB.USER.DB1.COPY';
}
```

The problem with this is pretty clear: you must keep track of the datafile numbers and these could change over time. For instance, if you were to drop and recreate the USER TABLESPACE, there's no guarantee that it would reacquire its original datafile number. Along the same lines, if you were to add a new TABLESPACE, or ALTER ADD DATAFILE to an existing TABLESPACE, you would have to modify your COPY DATAFILE specifications or employ some mechanism for doing so automatically.

In contrast to the specificity required when using the COPY command, the BACKUP command is considerably more flexible. For example, if you were to create a FULL backup of an instance, you could do that without naming all the underlying datafiles. For example, in the following illustration, we simply allocate a few channels and say “BACKUP FULL (DATABASE)”, supply a few FORMATing parameters, and that's it. RMAN takes care of everything else.

5.13.2 Creating a full backup of an Oracle database

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//*         FULL BACKUP                                             *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
RUN {
    ALLOCATE CHANNEL T1 TYPE DISK;
    ALLOCATE CHANNEL T2 TYPE DISK;
    BACKUP
     FULL
     FILESPERSET 1
     FORMAT "/DSN/TRGT.DBF.FULL.S%S.P%P.T%T"
     (DATABASE);
     RELEASE CHANNEL T1;
     RELEASE CHANNEL T2;
    }
```

In the preceding example, we are creating a backup of the entire database including the current control file. The specification of FULL causes all blocks to be copied into the backup set, skipping only datafile blocks that have never been used. All the blocks of the current control file are copied, however.

As you can see in the example, we have a FILESPERSET specification. This parameter is a means to govern and effect channel utilization. RMAN divides the total number of files requiring backups by the number of allocated “channels” in order to calculate the number of files to place in each backup set. When you specify the FILESPERSET parameter, RMAN compares the FILESPERSET value to this calculated value and takes the lowest of the two, thereby ensuring that all channels are used.

Notice also, we have specified a file name pattern for the backup set pieces as well as having indicated the number of files to go in each backup set. When and as needed, RMAN will create the necessary backup pieces which are collectively referred to as the “backup set.” There is quite a bit of flexibility in naming the individual backup pieces and these are documented in the generic Oracle Backup and Recovery documentation. But to illustrate, notice within the format specification from our example above:

```
FORMAT "/DSN/TRGT.DBF.FULL.S%S.P%P.T%T"
```

the “%S” and the “%P”. This specification causes RMAN to insert numeric values which represent the “backup set” to which the particular “backup piece” belongs, respectively. The “backup set” value is stored within the control file and is incremented from the last backup set number established and recorded within the target instance. The “backup piece” starts at 1 for each backup set and is incremented by 1 as each backup piece is created within the “backup set”.

5.13.3 Sample RMAN messages from full database BACKUP operation

```
Recovery Manager: Release 8.1.7.0.1 - Production

RMAN-06005: connected to target database: TRGT (DBID=2652822458)
RMAN-06008: connected to recovery catalog database

RMAN> RUN {
2>     ALLOCATE CHANNEL T1 TYPE DISK;
3>     ALLOCATE CHANNEL T2 TYPE DISK;
4>     BACKUP
5>      FULL
6>      FILESPERSET 1
7>      FORMAT "/DSN/TRGT.DBF.FULL.S%S.P%P.T%T"
8>      (DATABASE);
9>      RELEASE CHANNEL T1;
10>      RELEASE CHANNEL T2;
11>     }
12>
13>
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: T1
RMAN-08500: channel T1: sid=11 devtype=DISK

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
```

```
RMAN-08030: allocated channel: T2
RMAN-08500: channel T2: sid=12 devtype=DISK

RMAN-03022: compiling command: backup
RMAN-03025: performing implicit partial resync of recovery catalog
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-03023: executing command: backup
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=4 set_stamp=413226062 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00004 name=/DSN/TRGT.DBF.CONTEXT.DB3
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S4.P1.T0CA2L2E comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:21
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=5 set_stamp=413226083 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00006 name=/DSN/TRGT.DBF.RMANTEST.DB1
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S5.P1.T0CA2L33 comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:01
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=6 set_stamp=413226085 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00001 name=/DSN/TRGT.DBF.SYSTEM.DB1
RMAN-08011: including current controlfile in backupset
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S6.P1.T0CA2L35 comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:02:08
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=7 set_stamp=413226214 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00003 name=/DSN/TRGT.DBF.ROLLBACK.DB3
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S7.P1.T0CA2L76 comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:01:18
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=8 set_stamp=413226293 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00002 name=/DSN/TRGT.DBF.ROLLBACK.DB2
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S8.P1.T0CA2L9L comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:38
RMAN-08008: channel T2: starting full datafile backupset
RMAN-08502: set_count=9 set_stamp=413226331 creation_time=09-NOV-00
RMAN-08010: channel T2: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00005 name=/DSN/TRGT.DBF.USER4.DB4
RMAN-08013: channel T2: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.FULL.S9.P1.T0CA2LAR comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:09
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
```

```
RMAN-08031: released channel: T1

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: T2

Recovery Manager complete.
```

Note in the preceding output, how the “%S” and the “%P” variables were applied in the creation of the backup pieces. In particular, note the effect “FILESPERSET 1” has in the production of each backup dataset file name (in bold). Remember, this effect of “FILESPERSET” is primarily a means to govern and effect channel utilization, and not necessarily a means to group backup file pieces by “backup set” identifiers.

5.13.4 Backing up single tablespaces

In contrast to making a backup of an entire database, RMAN can be very selective and designate an individual tablespace. For instance, in the following illustration, the RMAN command to backup a single tablespace amounts simply to allocating a channel and specifying the tablespace to back up along with a FORMAT statement to be used in generating the output file backup set piece names.

5.13.5 Creating a backup of a single TABLESPACE

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//*          BACKUP TABLESPACE                                      *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORAPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
RUN {
    ALLOCATE CHANNEL T1 TYPE DISK;
    BACKUP TABLESPACE RMANTEST
     FORMAT "/DSN/TRGT.DBF.RMANTEST.S%S.P%P.T%T"
     ;
    }
```

5.13.6 Sample RMAN messages from backup of a single TABLESPACE

```
Recovery Manager: Release 8.1.7.0.1 - Production

RMAN-06005: connected to target database: TRGT (DBID=2652822458)
RMAN-06008: connected to recovery catalog database

RMAN> RUN {
2>    ALLOCATE CHANNEL T1 TYPE DISK;
```

```
3>     BACKUP TABLESPACE RMANTEST
4>      FORMAT "/DSN/TRGT.DBF.RMANTEST.S%S.P%P.T%T"
5>      ;
6>     }
7>
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: T1
RMAN-08500: channel T1: sid=11 devtype=DISK

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel T1: starting full datafile backupset
RMAN-08502: set_count=2 set_stamp=413224442 creation_time=09-NOV-00
RMAN-08010: channel T1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00006 name=/DSN/TRGT.DBF.RMANTEST.DB1
RMAN-08013: channel T1: piece 1 created
RMAN-08503: piece handle=/DSN/TRGT.DBF.RMANTEST.S2.P1.T0CA2JFQ comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:01
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: T1

Recovery Manager complete.
```

When and where to employ the BACKUP TABLESPACE command (or any RMAN command, for that matter) is largely an issue of your backup and recovery requirements. But as a general matter, the BACKUP TABLESPACE command's tablespace level specificity does lend itself well to situations where there is a periodic need to back up a selected set tablespace. This is especially true when there is a single table to single tablespace relationship or where a group of logically or functionally related sets of tables are stored within a single tablespace.

5.13.7 Recovering a single TABLESPACE to a point in time

The generic Oracle documentation refers to Recovery Manager point-in-time tablespace recovery as the “automated tablespace point-in-time recovery (TSPITR).” The use of the adjective *automated* should not be overlooked, because this is in fact a major “ease of use” enhancement over the pre-RMAN days, when “manual” methods were needed to employ a point in time tablespace recovery. In early releases of Oracle, this was, perhaps, one of the most daunting aspects of recovery in that there were many (13-14) manual steps that were required simply to recover a single tablespace to a particular point in time. It was not for the faint of heart, because not only did it involve identifying the appropriate backup set datafile copies or backups, but it also included the creation of a second transient (clone) Oracle server instance. The use of the clone, which is now called the auxiliary instance, was and still is a necessary and unavoidable ingredient to the recovery of a tablespace to a specific point in time.

You can perform RMAN TSPITR either with or without a recovery catalog. But if you do not use a recovery catalog, there are a few restrictions which you should be aware of.

First, because RMAN has no historical record of the rollback segments in TSPITR, RMAN assumes that the current rollback segments were the same segments present at the time to which recovery is performed. Secondly, if RMAN recovers to a remote time, Oracle may have reused the records of the copies and backups, thus making it impossible to perform TSPITR.

But, as we indicated in the beginning, we have assumed the use of a Recovery Catalog throughout all of our examples herein. It should be noted, however, that there are some restrictions on the use of RMAN's tablespace point in time recovery which should be reviewed and understood prior to implementation and/or use. This information can be found in Oracle Generic documentation on Backup and Recovery (*Oracle8i Backup and Recovery Guide Release 8.1.x*) in the sections entitled, “Performing Tablespace Point-in-Time Recovery with Recovery Manage” and “Understanding General Restrictions.”

Like a table export, RMAN TSPITR enables you to recover a consistent data set; however, the data set is the entire tablespace (which can contain multiple tables and/or indexes) rather than one object (a single table, for instance). The implications of this require careful consideration during the design and implementation of your application, as well as its backup and recovery requirements, both operational and application.

As the following figure illustrates, Recovery Manager does the following:

1. Restores the specified tablespace backups to a temporary auxiliary instance
2. Recovers the tablespace
3. Exports metadata from the auxiliary instance
4. Points the target database control file to the newly recovered datafiles
5. Imports metadata into the target database

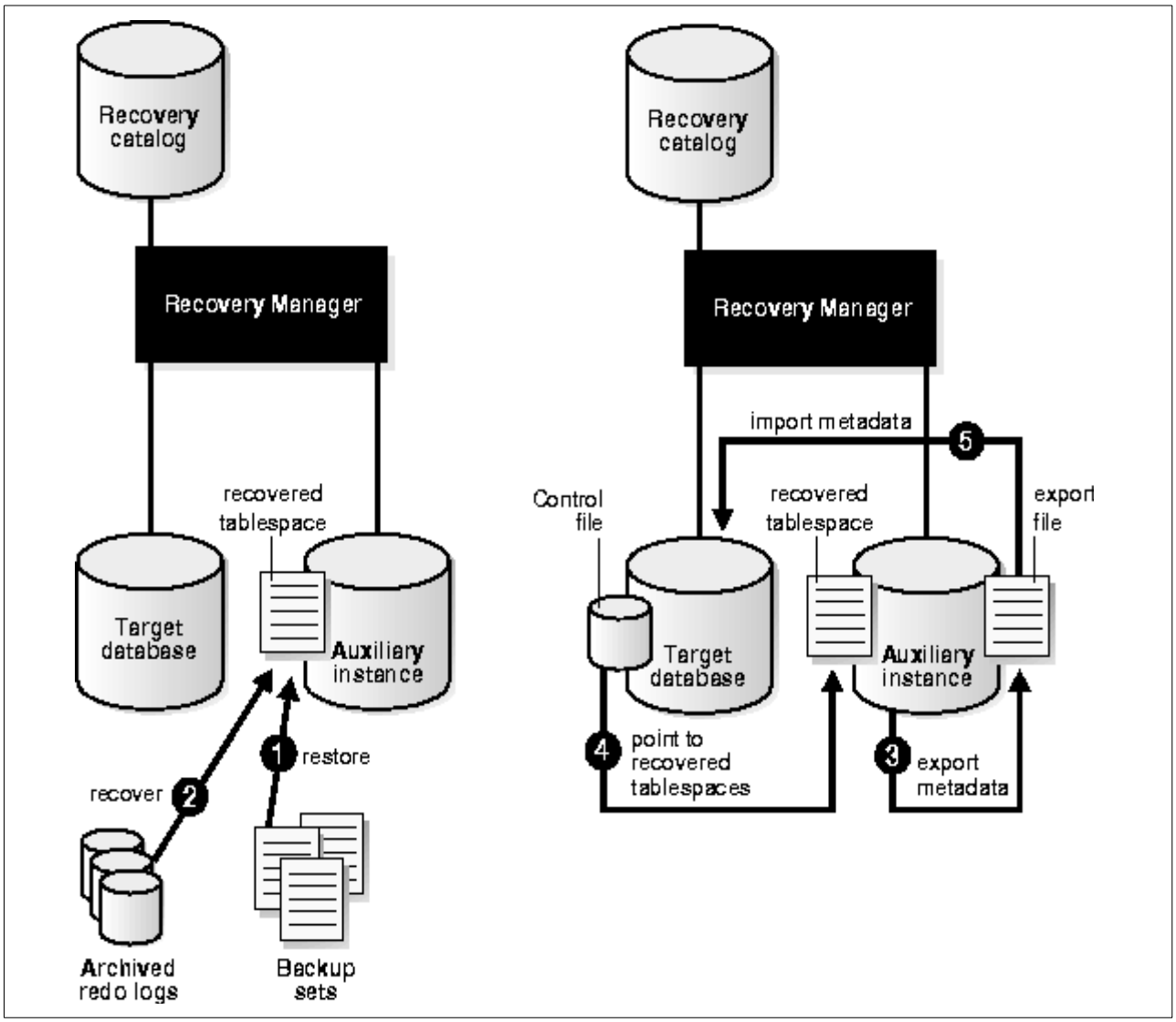

Figure 16. Point in time backup

The job required to perform a tablespace point in time recovery is quite simple, and at first glance, this simplicity masks the underlying complexity of the entire operation. For instance, in the following sample JCL, we can see that the specification for recovery of a tablespace to a particular point in time is fairly terse in its construction. Besides the channel allocations for the CLONE (auxiliary) and the RMAN instances, all we must say is:

```
RECOVER TABLESPACE "{some tablespace name}"
      UNTIL TIME "{some point in time}";
```

The complete job itself, is also fairly simple as in the following illustration.

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//* TABLESPACE RECOVERY UNTIL TIME {}                               *
//*-----------------------------------------------------------------*
//ORARMN   EXEC PGM=RMAN,REGION=0M,PARM='++/DD/SYSPARM'
```

```
//STEPLIB  DD DSN=ORA8.ORA817.CMDLOAD,DISP=SHR
//BSQ      DD DSN=ORA8.ORA817.SQL,DISP=SHR
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//ORA$VSAM DD DSN=TRGT.DBF.SNAPCF,DISP=SHR
//ORAPRINT DD SYSOUT=*
//ORA$ENV  DD *
NLS_LANG=AMERICAN
NLS_DATE_FORMAT='MON DD YYYY HH24:MI:SS'
//SYSIN    DD DUMMY
//SYSPARM  DD *
   TARGET INTERNAL/CHANGE_ON_INSTALL@W:TRGT RCVCAT RMAN/RMAN@W:RCAT
   CMDFILE "/DD/CMDFILE"
//CMDFILE  DD *
CONNECT CLONE INTERNAL/CHANGE_ON_INSTALL@W:CLON;
RUN {
    ALLOCATE CLONE CHANNEL C1 TYPE DISK;
    ALLOCATE CHANNEL C2 TYPE DISK;
    RECOVER TABLESPACE "RMANTEST" UNTIL TIME "NOV 07 2000 15:17:31";
    RELEASE CHANNEL C2;
    SQL 'ALTER TABLESPACE RMANTEST ONLINE';
    }
//SYSIN    DD DUMMY
```

Under OS/390, however, there are a few things, operationally speaking, which need to be accomplished before you can perform a point in time tablespace recovery as illustrated in the preceding sample JCL. Some of these can be done beforehand, while others must be performed immediately before the tablespace point in time recovery is executed.

First, you must define a subsystem ID for the auxiliary instance. In our examples, we chose CLON as the subsystem ID for the auxiliary instance. You can do this dynamically, via the SETSSI command or by updating IEFSSNxx and IPLing.

You will also need JCL for the auxiliary instance similar to the following:

```
//ORACLON  PROC SYSOUT='SYSOUT=X', JES SYSOUT CLASS.
//          INDEX='ORA8',
//          LIBV='ORA817',
//          START=NOMOUNT
//*
//*-----------------------------------------------------------------*
//* RMAN AUXILIARY INSTANCE
//*-----------------------------------------------------------------*
//*
//IEFPROC  EXEC PGM=ORACLE,PARM='>/OPER/',REGION=0M,
//             DYNAMNBR=40,
//             TIME=1440
//STEPLIB DD DSN=&INDEX..&LIBV..AUTHLOAD,
//            DISP=SHR
//SYSOUT   DD &SYSOUT,DCB=(LRECL=133,BLKSIZE=137,RECFM=VB,BUFNO=1)
//SYSERR   DD &SYSOUT,DCB=(LRECL=133,BLKSIZE=137,RECFM=VB,BUFNO=1)
//ORA$LIB DD DSN=&INDEX..&LIBV..AUTHLOAD,
//            DISP=SHR
//SQLBSQ   DD DSN=&INDEX..&LIBV..SQL(SQLBSQ),
//            DISP=SHR
//SYSPRINT DD DSN=CLON.DB.ALERT,DISP=SHR
//INITORA  DD DSN=CLON.DB.PARMLIB(INITORA),
```

```
//            DISP=SHR
//MPMTNS   DD DSN=CLON.DB.PARMLIB(MPMTNS),
//            DISP=SHR
//TNSNAMES DD DSN=CLON.DB.PARMLIB(TNSNAMES),
//            DISP=SHR
//STARTUP  DD DSN=CLON.DB.PARMLIB(&START),
//            DISP=SHR
//SYSIN    DD DSN=CLON.DB.PARMLIB(MPMPARM),
//            DISP=SHR
//         DD DSN=CLON.DB.PARMLIB(SQLDBA),
//            DISP=SHR
//SNAPCF   DD DSN=CLON.DB.SNAPCF.DB1,
//            DISP=SHR
//ORA$VSAM DD DSN=CLON.DB.SNAPCF.DB1,
//            DISP=SHR
```

Note in our JCL for the auxiliary instance, the NOMOUNT and the INITORA parameter library member specifications. The Oracle Auxiliary instance must be started NOMOUNT. This is required, and your member which is pointed to by the STARTUP DD name must contain the following statements:

```
SET ECHO ON
CONNECT INTERNAL
STARTUP PFILE=/DD/INITORA NOMOUNT
```

The INITORA is a bit more involved because it specifies some name translation parameters which will be employed during the creation of the auxiliary instance's datafiles. The following represents the contents of the entire INITORA file which we used. In particular, take note of the CONTROL_FILES, NLS_DATE_FORMAT and the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT parameters:

```
CONTROL_FILES     = "/DSN/CLON.DB.CONTROL"
SHARED_POOL_SIZE  = 4000000
DB_BLOCK_BUFFERS  = 500
DB_FILES          = 256
DB_NAME           = TRGT
LOG_BUFFER        = 65536
LOG_CHECKPOINT_INTERVAL = 3000
OPEN_CURSORS      = 120
TRANSACTIONS      = 55
SESSIONS          = 55
PROCESSES         = 50
DML_LOCKS         = 220
COMPATIBLE        = 8.0.4
NLS_DATE_FORMAT   = "MON DD YYYY HH24:MI:SS"
NLS_LANGUAGE      = "AMERICAN"
NLS_NUMERIC_CHARACTERS = "."
NLS_SORT          = BINARY
NLS_TERRITORY     = "AMERICA"
LOCK_NAME_SPACE   = _TRGT
DB_FILE_NAME_CONVERT=("/DSN/TRGT.DB","/DSN/CLON.DB")
LOG_FILE_NAME_CONVERT=("/DSN/TRGT.DB","/DSN/CLON.DB")
```

The CONTROL_FILES parameter designates the copy of the CONTROLFILE of the target instance you will have made before you start the auxiliary instance prior to a point in time tablespace recovery. Keep in mind, at this point we are simply discussing the contents of the INITORA parameter file's contents. When

you set up your auxiliary instances INITORA parameter file, the CONTROL_FILES pointed to will not exist, or at least not a current copy of it.

A little later you will be given an example of some JCL and the Oracle commands required to produce a backup copy of a CONTROLFILE for an instance.

Continuing on, NLS_DATE_FORMAT is not required, but is recommended in order to covey explicitly to the Oracle instance (the auxiliary, in this case) the format of date and time strings that will be used in any commands or SQL that the instance processes. NLS_DATE_FORMAT varies depending upon the NLS_TERRITORY variable and if the default format is not desirable you can change it, as we have, by explicitly setting the NLS_DATE_FORMAT's value. For a complete treatment of NLS, consult the Generic Oracle8i *National Language Support Guid*e.

Aside from the CONTROL_FILES parameter, the DB_FILE_NAME_CONVERT and the LOG_FILE_NAME_CONVERT are perhaps the most important parameters of the INITORA file.

These two parameters convey the transformation in datafile names which is to occur during the creation of the auxiliary instance's datafiles. In our example INITORA above, the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT specifications would take any reference to “TRGT.DB” and convert them to “CLON.DB”. For instance, if we were recovering a tablespace in the target which was called “RMANTEST” and the underlying VSAM dataset which housed it was called “TRGT.DBF.RMANTEST.DB1,” then it would be recreated as “CLON.DBF.RMANTEST.DB1” in the auxiliary instance.

A potentially confusing parameter is the DB_NAME parameter, especially considering that we are talking about the INITORA parameters for the auxiliary instance. Note in our example, that the DB_NAME parameter's value is 'TRGT” and that is also the name of the target database instance.

When the auxiliary instance is first started, it will read the copy of the target database instance's control file you will have made, and it will begin to reconstruct a copy of the target database instance.

The last parameter that needs to be mentioned is the LOCK_NAME_SPACE parameter. The generic Oracle RMAN documentation simply defines this parameter as “A value different from any database in the same $ORACLE_HOME.” In our case, we simply use the DB_NAME parameter's value and prefix it with a underscore (_), for example, “_TRGT”. Note that we are talking here about the INITORA parameters of the auxiliary instance.

Although the information on this parameter was scant in the RMAN documentation, the name seems to suggest a requirement for name collision avoidance that could perhaps occur when the target and auxiliary instances exchange data. In other words, it could be a means to establish the context in which a particular object reference is made during a recovery. But this is only speculation. In any case, it is required.

After you have your auxiliary instance's subsystem ID defined, the started task JCL written and the parameters coded up, you will then need to create a job which creates a backup copy of target instance's control file. The file this job

creates will be specified in the CONTROL_FILES parameter of the auxiliary instance's INITORA parameter file.

As you can see in the following illustration, the JCL and Oracle SQL required to make a copy of the CONTROLFILE is very simple:

```
//MCONNOLC JOB (0000,O7),'ORACLE INSTALL',CLASS=A,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//*  CREATE BACKUP OF TARGET DATABASE'S CONTROL FILE
//*-----------------------------------------------------------------*
//CNTLBACK EXEC ORASQLI2,REGION=4M
//*
//ORA@TRGT DD DUMMY      <=== ORA@SSN (ORACLE SUBSYSTEM NAME).
//SYSIN    DD *
SYSTEM/MANAGER
SET ECHO ON
ALTER DATABASE BACKUP CONTROLFILE TO 'CLON.DBF.CONTROL';
```

Note in this example that we are connecting via cross memory services (ORA@TRGT DD DUMMY) and that we are creating a CONTROLFILE called 'CLON.DB.CONTROL'.

After the copy of the target's CONTROLFILE is made, we can start the auxiliary instance. After the auxiliary instance has gone through initialization, you can begin tablespace point in time recovery.

The following is an example of the output generated from a tablespace point in time recovery. As you study the output, one of the most important things to note is the reconstruction of all the necessary datafiles for the tablespaces (SYSTEM, ROLLBACK, and RMANTEST) in the auxiliary instance which will be required to build a copy of the tablespace being restored. This can be observed by noting the datafile RESTOREs, that are followed by SQL which places the tablespaces ONLINE and followed finally by RECOVERY of each.

Remember, the tablespace being recovered is actually being reconstructed in the auxiliary instance and after it has be recovered up to the point in time specified, it will then be exported and re-imported into the target instance. You will see messages which reflects this export and import activity.

Before RMAN, this was a cumbersome process, because all of the steps had to be done manually, and without the benefit if the Recovery Catalog.

5.13.8 Sample RMAN messages from preceding recovery

```
Recovery Manager: Release 8.1.7.0.1 - Production

RMAN-06005: connected to target database: TRGT (DBID=2652822458)
RMAN-06008: connected to recovery catalog database

RMAN> CONNECT CLONE INTERNAL/CHANGE_ON_INSTALL@W:CLON;
2> RUN {
3>     ALLOCATE CLONE CHANNEL C1 TYPE DISK;
4>     ALLOCATE CHANNEL C2 TYPE DISK;
5>     RECOVER TABLESPACE "RMANTEST" UNTIL TIME "NOV 09 2000 17:14:00";
6>     RELEASE CHANNEL C2;
7>     SQL 'ALTER TABLESPACE RMANTEST ONLINE';
```

```
8>     }
9>
RMAN-06020: connected to auxiliary database

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: C1
RMAN-08500: channel C1: sid=9 devtype=DISK

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: C2
RMAN-08500: channel C2: sid=12 devtype=DISK

RMAN-03022: compiling command: recover

RMAN-03027: printing stored script: Memory Script
{
# set the until clause
set until  time 'NOV 09 2000 17:14:00';
# restore the controlfile
restore clone controlfile to clone_cf;
# replicate the controlfile
replicate clone controlfile from clone_cf;
# mount the controlfile
sql clone 'alter database mount clone database';
# archive current online log for tspitr to a resent until time
sql 'alter system archive log current';
# resync catalog after controlfile restore
resync catalog;
}
RMAN-03021: executing script: Memory Script

RMAN-03022: compiling command: set

RMAN-03022: compiling command: restore

RMAN-03022: compiling command: IRESTORE
RMAN-03023: executing command: IRESTORE
RMAN-08016: channel C1: starting datafile backupset restore
RMAN-08502: set_count=6 set_stamp=413226085 creation_time=NOV 09 2000
17:01:25
RMAN-08021: channel C1: restoring controlfile
RMAN-08505: output filename=/DSN/CLON.DBF.CONTROL
RMAN-08023: channel C1: restored backup piece 1
RMAN-08511: piece handle=/DSN/TRGT.DBF.FULL.S6.P1.T0CA2L35 tag=null
params=NULL
RMAN-08024: channel C1: restore complete

RMAN-03022: compiling command: replicate
RMAN-03023: executing command: replicate
RMAN-08058: replicating controlfile
RMAN-08506: input filename=/DSN/CLON.DBF.CONTROL

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database mount clone database
RMAN-03023: executing command: sql
```

```
RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter system archive log current
RMAN-03023: executing command: sql

RMAN-03022: compiling command: resync
RMAN-03023: executing command: resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete

RMAN-03027: printing stored script: Memory Script
{
# generated TABLESPACE point-in-time recovery script
# set the until clause
set until  time 'NOV 09 2000 17:14:00';
plsql <<<-- tspitr_2
declare
  sqlstatement       varchar2(512);
  offline_not_needed exception;
  pragma exception_init(offline_not_needed, -01539);
begin
  sqlstatement := 'alter TABLESPACE '||' RMANTEST'||' offline for recover';
  krmicd.writeMsg(6162, sqlstatement);
  krmicd.execSql(sqlstatement);
exception
  when offline_not_needed then
    null;
end; >>>;
# set a destination filename for restore
set newname for datafile  1 to
 "/DSN/CLON.DBF.SYSTEM.DB1";
# set a destination filename for restore
set newname for datafile  2 to
 "/DSN/CLON.DBF.ROLLBACK.DAB2";
# set a destination filename for restore
set newname for datafile  3 to
 "/DSN/CLON.DBF.ROLLBACK.DB3";
# set a destination filename for restore
set newname for datafile  6 to
 "/DSN/TRGT.DBF.RMANTEST.DB1";
# restore the TABLESPACEs in the recovery set plus the auxilliary
TABLESPACEs
restore clone datafile  1, 2, 3, 6;
switch clone datafile all;
#online the datafiles restored or flipped
sql clone "alter database datafile  1 online";
#online the datafiles restored or flipped
sql clone "alter database datafile  2 online";
#online the datafiles restored or flipped
sql clone "alter database datafile  3 online";
#online the datafiles restored or flipped
sql clone "alter database datafile  6 online";
# make the controlfile point at the restored datafiles, then recover them
recover clone database TABLESPACE  RMANTEST, SYSTEM, ROLLBACK;
alter clone database open resetlogs;
# PLUG HERE the creation of a temporary TABLESPACE if export fails due to
lack
# of temporary space.
# For example in Unix these two lines would do that:
```

```
#sql clone "create TABLESPACE aux_tspitr_tmp
#           datafile ''/tmp/aux_tspitr_tmp.dbf'' size 500K";
}
RMAN-03021: executing script: Memory Script

RMAN-03022: compiling command: set

RMAN-03022: compiling command: PLSQL
RMAN-03023: executing command: PLSQL
RMAN-06162: sql statement: alter TABLESPACE  RMANTEST offline for recover

RMAN-03022: compiling command: set

RMAN-03022: compiling command: set

RMAN-03022: compiling command: set

RMAN-03022: compiling command: set

RMAN-03022: compiling command: restore

RMAN-03022: compiling command: IRESTORE
RMAN-03023: executing command: IRESTORE
RMAN-08016: channel C1: starting datafile backupset restore
RMAN-08502: set_count=6 set_stamp=413226085 creation_time=NOV 09 2000
17:01:25
RMAN-08089: channel C1: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00001 to /DSN/CLON.DBF.SYSTEM.DB1
RMAN-08023: channel C1: restored backup piece 1
RMAN-08511: piece handle=/DSN/TRGT.DBF.FULL.S6.P1.T0CA2L35 tag=null
params=NULL
RMAN-08024: channel C1: restore complete
RMAN-08016: channel C1: starting datafile backupset restore
RMAN-08502: set_count=7 set_stamp=413226214 creation_time=NOV 09 2000
17:03:34
RMAN-08089: channel C1: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00003 to /DSN/CLON.DBF.ROLLBACK.DB3
RMAN-08023: channel C1: restored backup piece 1
RMAN-08511: piece handle=/DSN/TRGT.DBF.FULL.S7.P1.T0CA2L76 tag=null
params=NULL
RMAN-08024: channel C1: restore complete
RMAN-08016: channel C1: starting datafile backupset restore
RMAN-08502: set_count=8 set_stamp=413226293 creation_time=NOV 09 2000
17:04:53
RMAN-08089: channel C1: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00002 to /DSN/CLON.DBF.ROLLBACK.DB2
RMAN-08023: channel C1: restored backup piece 1
RMAN-08511: piece handle=/DSN/TRGT.DBF.FULL.S8.P1.T0CA2L9L tag=null
params=NULL
RMAN-08024: channel C1: restore complete
RMAN-08016: channel C1: starting datafile backupset restore
RMAN-08502: set_count=10 set_stamp=413226835 creation_time=NOV 09 2000
17:13:55
RMAN-08089: channel C1: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00006 to /DSN/TRGT.DBF.RMANTEST.DB1
RMAN-08023: channel C1: restored backup piece 1
RMAN-08511: piece handle=/DSN/TRGT.DBF.RMANTEST.S10.P1.T0CA2LQJ tag=null
params=NULL
```

```
RMAN-08024: channel C1: restore complete

RMAN-03022: compiling command: switch
RMAN-03023: executing command: switch
RMAN-08015: datafile 6 switched to datafile copy
RMAN-08507: input datafilecopy recid=8 stamp=413227671
filename=/DSN/TRGT.DBF.RMANTEST.DB1

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database datafile  1 online
RMAN-03023: executing command: sql

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database datafile  2 online
RMAN-03023: executing command: sql

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database datafile  3 online
RMAN-03023: executing command: sql

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter database datafile  6 online
RMAN-03023: executing command: sql

RMAN-03022: compiling command: recover

RMAN-03022: compiling command: recover(1)

RMAN-03022: compiling command: recover(2)

RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery

RMAN-03022: compiling command: recover(4)
RMAN-06050: archivelog thread 1 sequence 14 is already on disk as file
/DSN/TRGT.DBF.ARCH.T0001.S0000014.LOG
RMAN-03023: executing command: recover(4)
RMAN-08515: archivelog filename=/DSN/TRGT.DBF.ARCH.T0001.S0000014.LOG
thread=1 sequence=14
RMAN-08055: media recovery complete

RMAN-03022: compiling command: alter db
RMAN-06400: database opened

RMAN-03027: printing stored script: Memory Script
{
# export the TABLESPACEs in the recovery set
host 'exp userid ="INTERNAL/CHANGE_ON_INSTALL@W:CLON as sysdba"
point_in_time_recover=y TABLESPACEs=
 RMANTEST file=
tspitr$a';
# shutdown clone before import
shutdown clone immediate
# import the TABLESPACEs in the recovery set
host 'imp userid ="INTERNAL/CHANGE_ON_INSTALL@W:TRGT as sysdba"
point_in_time_recover=y file=
tspitr$a';
```

```
# online/offline the TABLESPACE imported
sql "alter TABLESPACE  RMANTEST online";
sql "alter TABLESPACE  RMANTEST offline";
# resync catalog after tspitr finished
resync catalog;
}
RMAN-03021: executing script: Memory Script

RMAN-03022: compiling command: host
RMAN-06134: host command complete

RMAN-03022: compiling command: shutdown
RMAN-06405: database closed
RMAN-06404: database dismounted
RMAN-06402: Oracle instance shut down

RMAN-03022: compiling command: host
RMAN-06134: host command complete

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter TABLESPACE  RMANTEST online
RMAN-03023: executing command: sql

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: alter TABLESPACE  RMANTEST offline
RMAN-03023: executing command: sql

RMAN-03022: compiling command: resync
RMAN-03023: executing command: resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: C2

RMAN-03022: compiling command: sql
RMAN-06162: sql statement: ALTER TABLESPACE RMANTEST ONLINE
RMAN-03023: executing command: sql

Recovery Manager complete.
```

This ends the section on Recovery Manager on S/390 (MPM). Again, it is not intended to be a comprehensive treatment of backup and recovery options for Oracle on S/390, but rather is designed to give you a general sense of the architecture and functionality.

Chapter 6. Using partitioned tables

With the release of Oracle8, several database changes were made that improve the performance and manageability of the Oracle database. These changes are an intergral part of the Oracle database on OS/390. One of the improvements is the addition of partitioned tables.

This chapter demonstrates some of the basics of creating a partitioned table and shows some of the performance improvements that can realized from partitioned tables in conjunction with the Oracle Parallel Query feature. Please do not confuse this with Oracle Parallel Server. Both partitioned tables and parallel query should be used together to maximize performance. Partitioning is available under Oracle's Enterprise Edition only.

6.1 Partitioned tables

Partitioning tables allows the user to break up tables into smaller and more manageable pieces. For example, as tables grow larger and larger, this can help performance by reducing the number of rows that have to be scanned to determine the projection of the SQL command executed. Another advantage is the ability to back up and recover partitions separately.

The following list presents other advantages to using partitioned tables:

- Very Large Database - Both OLTP and DSS databases can benefit from partitioned tables. Large tables used in a DSS system can be partitioned into segments, such as date or region, thereby helping large queries perform better. OLTP segments (partitions) can be maintained or physically placed on disk drives more easily.
- Reduced downtime for maintenance and outages.
- Partition transparency - Table appears as a single structure but can also be referenced as a single partition.

6.2 Parallel query

First, parallel query is not the same as the Oracle Parallel Server (OPS). OPS runs multiple instances of a database that run against the same tablespaces (database files). Parallel query means that there is parallel execution of SQL statements. This mode of executing SQL can dramatically improve performance with data-intensive operations such as decision support applications or very large databases, VLDBs. SMP processors will also gain performance improvements. If parallel query is not used, all SQL statement execution is serialized.

Some of the operation type that can be parallelized are:

- Select distinct
- Certain index operations
- Order by and group by
- Update or delete
- PL/SQL functions called from SQL

A system's scalability can be improved by using parallel query, thus making optimal use of the system's hardware resources.

6.3 Creating a partitioned table

To create a partitioned table, you must create a tablespace and at least one datafile for each partition that the table will contain. The `CREATE TABLE` DDL command is used to create the partitioned table. The statement must specify the following:

1. The logical attributes of the table, including column and constraint definitions.
2. The physical attributes of the table, such as `PCTFREE`, `PCTUSED`, `STORAGE CLAUS`, and so forth.
3. The partition specification information. This includes the method or algorithm to map rows to partitions and the description of the partition. This description would include the tablespace name and the parameters for selecting rows for the partition.

Before you try to create the table, be sure that you selected the Partitioned Table option during the install process. Check the `MPMPARM` member to ensure that partitioning is available. In our database, the following parameter was in the `ORA8.ORA804.PARMLIB(MPMPARM)` member:

```
SERVER_OPTS=(OBJS,PART)
```

There are some datatype restrictions with partitioned tables, specifically:

- `LONG`, `LONG RAW` and `LOB` data types (`BLOB`, `CLOB`, `NCLOB`, or `BFILE`) cannot be used with partitioned tables.
- Bitmap Indexes can be used on partitioned tables, but they must be local to the partitioned table and can not be global indexes.
- The cost-based optimizer must be used when executing SQL that is used to access a partitioned table. Rules-based optimization is not available for partitioned tables.

Tables can be partitioned by either a range or value, as follows:

```
PARTITION BY RANGE ( column_list )
```

or

```
PARTITION LESS THAN ( value_list )
```

The values in the column_list constitute the row's partitioning key; a value_list is either a literal or can be resolved by a `TO_DATE()` or `RPAD()` function.

6.3.1 Creating the partitioned table

We decided to create a partitioned table with four partitions, partitioned on the first letter of a character string. The data used came from the local IBM Callup phone book system.

We used a batch job to create four `VSAM` clusters — `MPM8.CALLUPP1.DB1` through `MPM8.CALLUPP4.DB1` – and ran `CCF` against them as part of the batch job.

We then defined four tablespaces — callup_p1 through callup_p4 — using the following SQL:

```
CREATE TABLESPACE callup_p1
DATAFILE '/DSN/MPM8.CALLUPP1.DB1' SIZE 8M
DEFAULT STORAGE (INTIIAL 500K NEXT 100K);
```

After the tablespaces were created, we used the following (SQL Figure 17) to create the partitioned table.

```
CREATE TABLE callup_pt (
   pri_key         NUMBER
          CONSTRAINT pk_callup_pt PRIMARY KEY,
   lname           VARCHAR(30),
   fname           VARCHAR(20),
   phone           VARCHAR(10),
   employ_type     VARCHAR(10),
   dept            VARCHAR(5)
   )
STORAGE (INTIAL 200K, NEXT 100K)
PARTITION BY RANGE (lname)
(
   PARTITION a_to_f VALUES LESS THAN ( 'G' )
        TABLESPACE callup_p1,
   PARTITION g_to_l VALUES LESS THAN ( 'M' )
        TABLESPACE callup_p2,
   PARTITION m_to_r VALUES LESS THAN ( 'S' )
        TABLESPACE callup_p3,
   PARTITION s_to_z VALUES LESS THAN ( MAXVALUE )
        TABLESPACE callup_p4,
)
DEGREE OF PARALLEL 4;
```

Figure 17. CREATE TABLE DDL for partitioned table

After creating the partitioned table, we ran the query against DBA_PART_TABS.

```
SELECT partition_name, partition_position
FROM dba_part_tabs
WHERE table_name = 'CALLUP_NT';
```

The results were as shown in Figure 18.

```
SQL >

PARTITION_NAME          PARTITION_POSITION
------------------------------ ------------------
G_TO_L                                  2
M_TO_R                                  3
S_TO_Z                                  4
A_TO_F                                  1

SQL >
```

Figure 18. Display of Partition Positions for partitioned table CALLUP_PT

This information, particularly the PARTITION_POSITION, was needed later when we did the explain plans for queries against the partitioned table.

The CREATE and STORAGE clauses are standard for creating any table. The PARTITION BY RANGE statement determines how rows will be selected for each partition. In this case, they are determined by the last name (lname) of the person. In the PARTITION section, the partition names are declared (i.e. a_to_f) and the value used to select

the partition. Note that VALUES LESS THAN must be one increment higher than the maximum value in the range. The last partition uses MAXVALUE, which will catch all strings with an initial character greater than "S".

At this point, queries can be constructed against this table. It looks to SQL as if it were one big table. Depending on the query, Oracle will search through each partition in sequence. Any I/O needed will also be serialized by partition. If you know which partition contains the data you want to select, that partition can be addressed directly with SQL.

With a little effort, we can add definitions which allow Oracle to significantly improve SQL operations.

6.3.2 Running the partitioned table with parallel query

There are a few additional tasks that must be completed to enable parallel query of a partitioned table . As mentioned previously, the Parallel Query option must have been selected when the database was installed. Additionally, two parameters must be placed in the initora.

```
PARALLEL_MIN_SERVERS = 2
PARALLEL_MAX_SERVERS = 8
```

The first parameter, PARALLEL_MIN_SERVERS is the number of available parallel process server processes Oracle starts when the instance is brought up. The second parameter determines the maximum number of parallel processes that are available to execute SQL. We selected 2 and 8 as the minimum and maximum. There were no particular performance reasons in this selection. It ensured that there were at least four server processes available to run against the four partitions of the table.

Next, a Degree of Parallel must be selected. The Oracle parallel coordinator first looks at the actual SQL command for a hint. If the SQL does not contain a hint, it then looks at the table for a degree of parallel (DOP) attribute. If neither had a DOP set, it determines a default DOP based on several factors, such as the number of CPUs, the number is disks the table is stored on, or the number of partitions that will be involved in the selection process.

A hint supplied in the SQL looks like the following:

```
SELECT /* + PARALLEL(callup_pt,4) */ *
FROM callup_pt;
```

This example tells the SQL processor that a DOP of four should be used on the table.

The DOP can also be specified as an attribute of the table. Refer to Figure 17 on page 73 for the SQL we used to create the partitioned table. With the DOP set on the table, no hint is required in the SQL.

6.4 Examples of queries

We used the MVS GTF trace facility to trace the I/O activity on the partitioned table with and without the parallel query. To get the results without parallel query, we dropped the DOP from the table and set the PARALLEL_MIN_SERVERS to 0.

The GTF trace JCL is as follows:

```
//GTFORA  PROC MEMBER=GTFORA
//* 'S GTFORA.GTF,MEMBER=GTFORA
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
//  TIME=1440,REGION=2880K
//IEFRDER DD   DSNAME=ORACLE1.GTF.TRACE,DISP=OLD
//SYSLIB  DD   DSNAME=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

Figure 19. JCL for GTF

Looking at the following output of the GTF trace, you can see that the I/O is sequential. There is a start subchannel followed by the IO. This is obviously a serialized operation, and slower than if this were done as a parallel operation.

```
- SSCH.... 2468     ASCB.... 00F5FD00 CPUID... 0000     JOBN.... ORAMPM8
                    CC...... 00       ORB..... 00F22810 83C0F000 3121C448
                    OPT..... 14       FMSK.... 08       DVRID... 03
                    BASE.... 2468
                GMT-03/09/2000 21:05:19.595400  LOC-03/09/2000 16:05:19.5
0 IO...... 2468     ASCB.... 00F5FD00 CPUID... 0001     JOBN.... ORAMPM8
                    IRB..... 80C04007 3121C460 0C000000 00800008 00000000
                    OPT..... 14       DVRID... 03       IOSLVL.. 01
                GMT-03/09/2000 21:05:19.596863  LOC-03/09/2000 16:05:19.5
0 SSCH.... 2467     ASCB.... 00F5FD00 CPUID... 0000     JOBN.... ORAMPM8
                    CC...... 00       ORB..... 00F22788 83C0F000 3121C448
                    OPT..... 14       FMSK.... 08       DVRID... 03
                    BASE.... 2467
                GMT-03/09/2000 21:05:19.841438  LOC-03/09/2000 16:05:19.8
0 IO...... 2467     ASCB.... 00F5FD00 CPUID... 0000     JOBN.... ORAMPM8
                    IRB..... 80C04007 3121C460 0C000000 00200006 00000000
                    OPT..... 14       DVRID... 03       IOSLVL.. 01
                GMT-03/09/2000 21:05:19.842591  LOC-03/09/2000 16:05:19.8
```

Figure 20. GTF output for nonparallel query

We then set the DOP on the table to four using the following SQL:

```
ALTER TABLE callup_pt
DEGREE OF PARALLEL 4;
```

The GTF was rerun. Notice the results now. Several SSCH operations are executed without waiting for the IO to complete. These I/O operations are processed in parallel under control of the Oracle parallel coordinator. For large tables, for example, those used in a Business Intelligence application, processing all the partitions in parallel can significantly reduce the overall response time for the query.

```
0 SSCH.... 2468      ASCB.... 00F56400 CPUID... 0002     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F22170 83C0F000 311F3528
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2468
                  GMT-03/29/2000 19:15:16.841244  LOC-03/29/2000 14:15:16.8
0 SSCH.... 2527      ASCB.... 00F56400 CPUID... 0000     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F242E8 83C0E000 311F3B10
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2527
                  GMT-03/29/2000 19:15:16.841307  LOC-03/29/2000 14:15:16.8
0 SSCH.... 2467      ASCB.... 00F56400 CPUID... 0000     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F220E8 83C0F000 1E2B0B10
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2467
                  GMT-03/29/2000 19:15:16.841843  LOC-03/29/2000 14:15:16.8
0 SSCH.... 2626      ASCB.... 00F56400 CPUID... 0002     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F2B960 83C0F000 09E9E528
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2626
```

Figure 21. GTF output for parallel query on partitioned table.

More information on running a GTF trace is shown in Appendix B, “Using GTF Trace, ESTAT/BSTAT and RMF” on page 107.

6.5 Indexes and partitioned tables

Partitioned tables can also be indexed to further enhance performance. After creating the partitioned table callup_pt, we created an index on the *dept* column.

To create the index, another tablespace was created to hold the index on the callup_pt table. To accomplish this, a VSAM cluster was created and CCF run against it using a batch job. The following DDL was used to create the tablespace:

```
CREATE TABLESPACE callup_indx
DATAFILE '/DSN/MPM8.CALLIDX.DB1'
DEFAULT STORAGE (INTIAL 500K NEXT 100K);
```

After creating the tablespace, the index was created with the following DDL:

```
CREATE INDEX callup_indx1 ON callup_pt(lname)
LOCAL(
   PARTITION a_to_f TABLESPACE callup_indx,
   PARTITION g_to_l TABLESPACE callup_indx,
   PARTITION m_to_r TABLESPACE callup_indx,
   PARTITION s_to_z TABLESPACE callup_indx
);
```

Note that this is a *local* index. A local index is one that refers only to rows stored in a single underlying table partition. A *global* index is one where the keys refer to rows stored in more than one underlying partition. There are rules that govern the use of these indexes; they are thoroughly defined in the Oracle8 Concepts document A58383-01 and the Oracle Tuning Guide document A58246. A complete explanation of partitioned keys is beyond the scope of this project.

6.5.1 The explain plan

We wrote a script to project results of the EXPLAIN PLANs we ran. The script selected a few basic indicators that are germane to this project. The script we used is shown in Figure 22.

```
-- explain plan script for partitioned tables

COLUMN operation   FORMAT a20
COLUMN options     FORMAT a15
COLUMN object_name FORMAT a20
COLUMN part_start  FORMAT a11
COLUMN part_stop  FORMAT a11
COLUMN id          FORMAT 99
COLUMN parent_id   FORMAT 99
COLUMN position    FORMAT 99
--COLUMN WRAP

SELECT operation,options,object_name,id, position,object_name,partition_start AS PART_START,
       partition_stop AS PART_STOP
FROM   plan_table
WHERE  statement_id='&statement_id'
ORDER BY id;

CLEAR COLUMNS
```

Figure 22. EXPLAIN PLAN script

The important indicators are:

- Operation - what was executed during the SQL command
- Object_name - name of the table or index used
- ID and position - the sequence and sub-sequence of operations
- Part_start - the starting partition number for the operation
- Part_stop - the ending partition number for the operation

We wrote scripts to execute the EXPLAIN PLAN SQL. A sample script is in Figure 23.

```
EXPLAIN PLAN
FOR SELECT lname, fname, dept
FROM   callup_pt
WHERE  lname LIKE 'H%'
ORDER BY lname;
```

Figure 23. Sample EXPLAIN PLAN

6.5.2 Query without index

A simple query was entered without the index applied to the column of the partitioned table we queried. The query was:

```
SELECT lname, fname, dept
FROM callup_pt
WHERE dept = 'DIRE';
```

Here are two results. First an EXPLAIN PLAN was executed; the results are in Figure 24. Notice that without the index, a full table scan is performed. This is

indicated by TABLE ACCESS FULL on line three of the output. You can also see that the query started in partition NUMBER(1) and completed in partition NUMBER(4) . It scanned all four partitions of the table.

```
SQL> @explain_single

OPERATION          OPTIONS       OBJECT_NAME         ID POSITION OBJECT_NAME        PART_START  PART_STOP
-------------------- -------------- -------------------- -- -------- -------------------- ---------- ---------
SELECT STATEMENT                                         0       19
PARTITION           CONCATENATED                         1        1                   NUMBER(1)   NUMBER(4)
SORT                ORDER BY                             2        1
TABLE ACCESS        FULL          CALLUP_PT              3        1 CALLUP_PT          NUMBER(1)   NUMBER(4)
```

Figure 24. EXPLAIN PLAN without index applied

6.5.3 Query with index

We did another query as follows to take advantage of the index we created:

```
SELECT lname, fname, detp
FROM callup_nt
WHERE lname LIKE 'H%';
```

To verify the fact that the index was used, we ran another EXPLAIN PLAN.

```
SQL> @explain_part
Enter value for statement_id: query_1
old   4: WHERE  statement_id='&statement_id'
new   4: WHERE  statement_id='query_1'

OPERATION          OPTIONS         OBJECT_NAME        ID POSITION OBJECT_NAME       PART_START  PART_STOP
-------------------- -------------- -------------------- -- -------- -------------------- ---------- ---------
SELECT STATEMENT                                         0        3
TABLE ACCESS       BY LOCAL INDEX  CALLUP_PT            1        1 CALLUP_PT         NUMBER(2)   NUMBER(2)
                   ROWID

INDEX              RANGE SCAN      CALLUP_INDX_NAME     2        1 CALLUP_INDX_NAME  NUMBER(2)   NUMBER(2)
```

Figure 25. Results of query on partitioned table

These results demonstrate how the query was executed. The first line indicates that we did a select. The next line shows that we selected the rows by ROWID using the local index. We ran against the table callup_pt and we ran against partition NUMBER(2). Earlier we ran a query against the catalogue and found that partition_position 2, or NUMBER(2) in the part_start and part_stop columns of the EXPLAIN PLAN, was the partition G_TO_L, so it included all last names LIKE 'H%' or all last names beginning with the letter "H."

Line 3 of the EXPLAIN PLAN shows that the index was used for this SELECT.

One final example was to run an EXPLAIN PLAN on the following query:

```
SELECT lname, fname, dept
FROM callup_pt
WHERE lname LIKE 'H%'
or lname = 'DUTCAVICH';
```

This should use the indexes to determine that only partitions NUMBER(1) and NUMBER(2) need to be scanned to find the data. Partition NUMBER(1) was scanned to find the name DUTCAVICH and partition NUMBER(2) was scanned to find the names beginning with the letter "H". The results are in Figure 26 on page 79.

```
SQL> @explain_single

OPERATION         OPTIONS       OBJECT_NAME        ID POSITION OBJECT_NAME        PART_START  PART_STOP
-----------------------------------------------------------------------------------------------------------
SELECT STATEMENT                                      0      13
SORT               ORDER BY                          1       1
CONCATENATION                                        2       1
TABLE ACCESS        BY LOCAL INDEX  CALLUP_PT          3        1 CALLUP_PT        NUMBER(1)  NUMBER(1)
                   ROWID

INDEX            RANGE SCAN     CALLUP_INDX_LNAME    4      1 CALLUP_INDX_LNAME     NUMBER(1)  NUMBER(1)
TABLE ACCESS       BY LOCAL INDEX  CALLUP_PT          5      2 CALLUP_PT             NUMBER(2)  NUMBER(2)
                   ROWID

INDEX            RANGE SCAN     CALLUP_INDX_LNAME    6      1 CALLUP_INDX_LNAME     NUMBER(2)  NUMBER(2)

7 rows selected.
```

Figure 26. EXPLAIN PLAN for selects to multiple partitions

From the results of the EXPLAIN PLAN, we can see that the results are as we predicted earlier. A TABLE ACCESS using the index was performed on partitions NUMBER(1) for the name DUTCAVICH and partition NUMBER(2) for the names beginning with the letter H.

A word of caution here. Partitioning is a valuable tool in managing an Oracle instance. If you are considering moving an Oracle database from UNIX to OS/390 with partitioned tables, you need to pay close attention to the column data that is acting as the partition key. There are differences in the sort sequences between ASCII and EBCDIC. This can effect what data goes in what partition, and the results may not be what you expected.

Chapter 7. Installing and configuring Oracle Enterprise Manager

The purpose of this chapter is to document our experience installing the Oracle Enterprise Manager (OEM), the Intelligent Agent (IA) on an NT system, and the IA on OS/390 using OS/390 UNIX System Services. We used OEM V2.0.4.0. This version uses a three-tiered approach. We chose to install the Oracle Management Server (OMS) on the same NT box that we will run the OEM Console on. The OMS can be installed on one of several Web servers, including the Oracle Applications Web server. The OEM console can then be accessed from a browser such as Netscape or Internet Explorer.

The focus of the chapter is to demonstrate that an Oracle instance on OS/390 can be monitored and managed using OEM in the same way you would manage any other instance of Oracle on any platform other than S/390. It is based on experiences with Oracle8, not Oracle8i.

There are several documents available that describe the installation and configuration processes. They are available on the Oracle documentation CD-ROMs available from Oracle. It is recommended, although not necessary, to print the documents. They are available in PDF formats that can be printed and in HTML format that can be viewed with a browser. The documents are:

- *Oracle for OS/390 Installation Guide* - A68790-01
- *Oracle Intelligent Agent Users Guide* - A67825-01
- *Oracle Enterprise Manager Quick Start Guide* Doc Note:107241
- *Oracle Enterprise Manager Configuration Guide* A67823-01
- *Oracle Enterprise Manager Administrators Guide* A67820-01

7.1 Introduction to Intelligent Agents and OEM

Intelligent Agents are autonomous since they function without requiring that the console or Management Server be running. The agent that services the Oracle instance (or instances) can run when the database is down. This allows the IA to start and shut down instances. The IAs can also perform administrative tasks and run applications without direct intervention from the DBA. They can also detect errors and issue predetermined corrective actions to resolve problems. All job and event messages are queued, enabling the work to take place when the OMS is not active.

The OEM Version 2 is composed of three tiers:

1. First tier. This tier comprises all the client tools, such as the EM Console, DBA Management Pack, Performance Pack, and other integrated tools. The tools can only be launched from a Windows95/98/NT client or from Sun Solaris. The client can also be run from a Web browser, and all the applications can be downloaded from the OMS as Java applets.
2. Middle Tier. The second tier is the Oracle Management Server. The OMS must be installed on either NT or Sun Solaris. One OMS must be configured in order to use the Enterprise Manager console and to use the job and event system. OMS provides:
   - Access to the Repository. The Repository holds information about the instances being managed.
   - Event monitoring.
   - Request dispatching.

- Event notification (email or pager).
- Multiple OMSs can be used to share and balance workloads.

3. Third Tier. This is the Intelligent Agent. It must be installed on all severs that are to be managed by OEM. Managed objects include network nodes (servers), databases, listeners, and other Oracle Services.

All three tiers can exist on the same machine or spread over any number of separate machines in essentially any combination.

We installed the OMS and EM Console (tiers one and two) on a single Windows NT 4.0 Workstation and IAs on the same NT box and on an OS/390 using OS/390 UNIX System Services. In this configuration we can manage an Oracle 8.0.5 database on NT that will also contain the OEM Repository and an instance of Oracle8i (release 8.1.5) on OS/390.

7.2 Installing and configuring an OEM configuration

We performed three major steps to install OEM so that it will support an instance on NT and an instance on OS/390. First, we installed an Oracle8 database on NT. Next, we installed and configured the OEM on NT and ensured everything worked to this point. We then installed Oracle8i on the OS/390, including the Intelligent Agent in OS/390 UNIX System Services.

Before you can get OEM to support your Oracle databases, your environment must meet the following basic requirements:

1. A database for the Repository
2. Intelligent Agents that are correctly configured and running on the servers with the databases you want to monitor and administer
3. OEM installed and correctly configured
4. A dedicated IP address for the machine that will be running OM server. This is particularly important if you use DHCP. Request that the same IP address be provided through the DNS server for the machine name that is running OEM.

We executed the following steps to install and configure all the components needed to support an instance of Oracle on OS/390:

1. Installed Oracle 8.0.5 on NT Workstation 4.0 with ServicePack 4
   - Configured the Intelligent Agent and ensured its proper operation
2. Installed OEM in NT
   - Configured and created a repository
   - Ensured that auto-Discovery worked for the local Oracle instance
3. Installed the Intelligent Agent in OS/390 UNIX System Services
   - Configured and insured proper operation
   - Discovered multiple instances
4. Used OEM to connect and manage the nodes on S/390

7.2.1 Installing and configuring an IA on NT

Our first step was to install an instance of Oracle on NT. This was done on the same machine where we will be installing OEM. This instance will be used for the repository needed by OEM to store data about the instances being managed. The IA is installed during the database installation and runs as a service under Windows NT.

If you install Oracle8 or earlier on NT, you will need two ORACLE_HOMEs since OEM must run on an 8.1.5 or above ORACLE_HOME. We selected orant for the 8.0.5 database and orant2 as the ORACLE_HOME for OEM.

After Oracle8 was installed on the NT machine, we checked to see that the following services were installed and running:

- OracleAgent80 - The Intelligent Agent
- OracleServiceORCL - This is the Oracle Instance. The SID is the default SID ORCL.
- OracleStartORCL - This service will automatically start the instance ORCL when NT is booted
- OracleTNSListener80ORCL - The TNS listener

Note: ORCL is the SID of the database we were using. It is appended to the service name.

If these services are not set to `AUTOMATIC`, it is advisable to do so, unless you do not want the Oracle database to start when the machine is booted.

There are six files that are important and that must be configured correctly for the IA to discover the node (database). The files are:

- listener.ora - Describes the listener, protocols and ports.
- tnsnames.ora - Contains all the service names and network parameters needed to connect to any database on the local machine or on the network.
- sqlnet.ora - Contains parameters needed for Net8.
- snmp_ro.ora - Lists the nodes this agents works with. Do not write in this file; Oracle will make the appropriate entries.
- snmp_rw.ora - The same as the above file except the user can make changes here. The most common change that may be made is to enable tracing if something doesn't work.
- services.ora - This will contain the descriptors needed to contact the databases that are supported. Oracle writes to this file as well.

All of the above files are in C:\orant\Net80\Admin except for services.ora, which is in C:\orant\Net80\Agent.

These steps will ensure all the files are configured correctly to enable discovery of the databases on the local machine.

1. We started with the sqlnet.ora file; an example of our file is in Figure 27.

```
#C:\orant\Net80\Admin\sqlnet.ora

TRACE_LEVEL_CLIENT = OFF
#sqlnet.authentication_services = (NONE)
names.directory_path = (TNSNAMES)
names.default_domain = world
name.default_zone = world
automatic_ipc = off
```

Figure 27. Sample of sqlnet.ora file

The important entry in this file is the parameter `names.default_domain = world`. If `world` is used, then all service names must use world appended to it. For example, if you add an entry in your tnsnames.ora file for a database and identify it as MPM8, the entry in tnsnames would be identified by `MPM8.world`. Anything can be used here, although world is a common parameter and is the default.

2. The next file to check is the listener.ora. Figure 28 is a sample of ours.

```
# C:\orant\Net80\Admin\listener.ora

LISTENER  =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = IPC)
        (KEY = orcl.world)
       )

      (ADDRESS =
        (PROTOCOL = IPC)
        (KEY = ORCL)
      )

      (ADDRESS =
        (PROTOCOL = TCP)
        (Host = dutch)
        (Port = 1521)
      )
    )

STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = 0
SID_LIST_LISTENER =
      (SID_LIST =
              (SID_DESC =
                      (GLOBAL_DBNAME = orcl.world)
                      (SID_NAME = ORCL)
              )
     )
PASSWORDS_LISTENER = (oracle)
```

Figure 28. Sample of listener.ora file

There may be many entries in the default file provided by the installer. Three entries are required. There must be two that are `PROTOCOL = IPC` and one with `PROTOCOL=TCP`. The following entries are important:

- The first two entries are IPC protocols. The first IPC entry must have:

  `Key = orcl.world`

  and the second entry must have:

  `KEY = ORCL` (note capitalization)

  These are for local communications, interprocess communications, on the platform. This is also referred to as pipes.

Next, under the SID_LIST_LISTENER, is GLOBAL_DBNAME. This entry provides a global name to refer to the database that is being listened for by this listener. The recommend name to use for NT is the machine_name or host_name. Both the host_name and machine_name should be the same values. In our case, the SID is orcl, so our GLOBAL_DBNAME = orcl.world. The world is what we decided to use from the entry names.default_domain = world in the sqlnet.ora file. Next we made sure the SID_NAME was correct for our instance, ORCL..

3. The next file we configured was the tnsnames.ora. Note that this tnsnames file is in the ORACLE_HOME\Net80\Admin directory. There are tnsnames.ora files installed, make sure you use this one. The tnsnames.ora file in the ORACLE_HOME\Network\Admin directory is used when a tnsping command is executed. Below is a partial copy of our tnsnames file.

```
#C:\ORANT\Net80\Admin\Tnsnames.ora

syslab03.world =
  (DESCRIPTION =
     (ADDRESS =
        (PROTOCOL = TCP)
        (Host = syslab03)
        (Port = 1521)
      )
  (CONNECT_DATA = (SID = ORCL))
   )

MPMJ.world =
   (DESCRIPTION =
     (ADDRESS = (PROTOCOL = TCP)(HOST = 9.12.2.10))(Port = 1532))
     (CONECT_DATA = (SID = gb107))
   )

********************lines not shown*****************************
```

Figure 29. Sample tnsnames.ora file

The syslab03 entry is what the IA uses for discovery purposes. The service name should be the host_name of the machine, syslab03 in this case. The protocol must also be TCP. We used the default port of 1521. If there are additional instances that are managed with this IA, then the list of instances can be expanded by adding an ADDRESS LIST = list. An example of pseudo syntax is in Figure 30.

```
syslab03.world =
       (DESCRIPTION =
          (ADDRESS LIST =
             (ADDRESS =
                (descriptors))
             (ADDRESS =
                (descriptors))

       )
    )
```

Figure 30. Example of Address List

It is also advisable to put this as the first entry in the tnsnames.ora file.

4. There are three other files that are important to the discovery process:
   - ORACLE_HOME\Net80\Admin\snmp_ro.ora

     This file will possibly have one entry in before discovery:

     ifile = C:\orant\Net80\Admin\snmp_rw.ora

     This is an imbed file, indicating that it will use the descriptors in the snmp_rw.ora file. Do not write anything in this file, it is controlled by Oracle
   - ORACLE_HOME\Net80\Admin\snmp_rw.ora

     This file will contain information on what listeners to contact for nodes being managed by OEM. This file will be empty until the discovery process takes place. Oracle writes the needed information in the file. If there are problems with the IA, there are entries that can be placed in this file to turn on tracing. The procedure is in the *Intelligent Agent Users Guide* A67826.
   - ORACLE_HOME\Net80\Agent\services.ora

     This file will be empty until a successful discovery process. Oracle will list all the services that it is managing with OEM. Do not write in this file. It is recreated every time the IA is stopped and started.
5. After ensuring that these files looked correct, we selected **Start -> Control Panel -> Services** and stopped the following services:
   - OracleAgent80 - Stops the IA.
   - OracleTNSListener80ORCL - Stops the listener.
   - OracleServiceORCL - Brings down the database. This will also stop OracleStartORCL. This is the service that automatically starts the database when NT boots up.

   It you want to bring up these services when you boot NT, use the Services panel, then the StartUp... button, to select Automatic.
6. Next, we started all the services we stopped. We did it in this order: first we started the database, next the listener, and finally the agent. All started with no problems.
7. We checked the snmp_rw.ora, snmp_ro.ora and services.ora files to ensure that discovery was successful. The contents of our files are included as an example of successful discovery. First is an example of the snmp_ro.ora, in Figure 31.

```
snmp.shortname.LISTENER = LISTENER
snmp.longname.LISTERNER = LISTENER_syslab03
snmp.configfile.LISTENER = C:\orant\net80\admin\listener.ora
snmp.servicename.LISTENER_syslab03 = OracleTNSListener80
snmp.SID.syslab03.world = ORCL
snmp.oraclehome.syslab03.world = c:\orant
snmp.address.syslab03.world = (DESCRIPTION= (ADDRESS= (PROTOCOL = IPC)(KEY =
orcl.world)) (ADDRESS =(PROTOCOL = IPC)(KEY = ORCL)) (ADDRESS = (PROTOCOL =
TCP)(Host = syslab03)(Port = 1521)))(CONNECT_DATA=(SID=ORCL)(SERVER=
DEDICATED)))
ifile = C:\orant\net80\admin\snmp_rw.ora
```

Figure 31. Successful Discovery, snmp_ro.ora

Next is an example of the snmp_rw.ora, in Figure 32.

```
snmp.contact.LISTENER = ""
snmp.index.LISTENER = 1
snmp.contact.syslab03.world = ""
snmp.index.syslab03.world = 2
```

Figure 32. Successful Discovery, snmp_rw.ora

And finally, our services.ora is shown in Figure 33.

```
LISTENER_syslab03 = (ORACLE_LISTENER, syslab03, (ADDRESS_LIST = (ADDRESS =
PROTOCOL = TCP)(HOST = syslab03)(PORT=1521))))

syslab03.world = (ORACLE_DATABASE, syslab03, (DESCRIPTION=(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = syslab03)(PORT = 1521)))(CONNECT_DATA = (
SID=ORCL)(SERVER=DEDICATED))), LISTENER_syslab03)
```

Figure 33. Successful Discovery, services.ora

With the IA configured and running correctly, we went to the next step of our plan and installed OEM and ensured it discovered the Oracle instance on the same machine.

7.2.2 Installing the Oracle Enterprise Manager

Installing OEM is a fairly easy task. It is installed from a CD, and there is little interaction needed until it is time to create the repository. In preparation for installing OEM, there are a few steps that should be completed first. We found out about them during the install. Doing them now makes the installation process and configuring OEM easier. Here are our recommendations:

1. Get an IP address for the NT machine if you are using DHCP. OEM needs the same IP address for the machine name each time it runs. During startup it verifies that the IP address matches between the local hostname and hostname in the DNS server. The local machine's hostname must agree with the name from the DNS server. The entries in the Hosts file for OEM should be as follows:

   ```
   localhost      127.0.0.1      # local id and tcp loopback
   syslab03       9.36.0.9       # OEM
   ```

2. Change the process parameter in the initora in the local database that will be acting as the repository. OEM needs at least 200 processes. If not changed now, there will an error message stating you need to change the processes. You can continue, this just makes it simpler. This is the entry we made:

   ```
   processes = 250
   ```

3. When the repository is created, many database objects are created as part of the process. OEM asks what tablespace they should be installed in. You can use the default tablespace user_data that is created during the installation of the database. We chose to create another tablespace. We issued the following DDL from SQL*Plus while logged on to the orcl instance:

   ```
   CREATE TABLESPACE oemrepos
   DATAFILE 'C:\orant\database\oemrepos1.ora
   SIZE 5M;
   ```

We used the default storage parameters for the tablespace. If this were a tool to support several production instances, it would be worth the effort to more accurately define this tablespace.

Chapter 8. Installing the Intelligent Agent and Data Gatherer

This chapter documents our experience installing and configuring the Oracle Intelligent Agent on OS/390 Release 2.7 using OS/390 UNIX System Services.

There are two documents available for the installation and configuration processes. It is recommended, although not necessary, to print these documents. They are available in PDF formats, which can be printed, and HTML format. The generic document on the Intelligent Agent is available on the Oracle CD or from Metalink. The OS/390 document is available on Metalink only. Both formats can be viewed from a browser. The documents are:

- *Oracle for OS/390 OSDI System Administration Guide* - A85482
- *Oracle Intelligent Agent Users Guide* - A67825
  This is a generic guide for Intelligent Agent information and can be useful when configuring the Intelligent Agent.

8.1 Intelligent Agents and Oracle Enterprise Manager

Intelligent Agents are autonomous since they function without requiring the Oracle Enterprise Manager (OEM) console or Management Server to be running. The primary communications path from the Oracle Management Server to the database instance is through the Intelligent Agent. The agent that services an Oracle instance (or instances) can run when the database is down. This allows the Intelligent Agent to start and shut down instances. The IA can also perform administrative tasks and run applications without direct intervention from the DBA. It can detect errors and issue predetermined corrective actions to resolve the problem. All job and event messages are queued, enabling the work to take place when the OMS is not active.

8.1.1 Installation of the Intelligent Agent and Data Gatherer in OS/390 UNIX System Services

The installation process is simple and does not take much time. If you have done this before with an MPM version of the database, be advised that things have changed a bit. The changes will be pointed out in this chapter.

During the database installation, job `ORIJU01` was run to install scripts and to move the TAR files to OS/390 UNIX System Services. The `ussinst.sh` shell script was also run at that time. It installed all the utilities and OS/390 UNIX System Services functions selected during the installation. As a result of running this, a shell script, `customize.sh`, was placed in the $ORACLE_HOME/bin directory. The `customize.sh` shell script is used to install the Oracle Intelligent Agent.

Before you run the `customize.sh` script, check your .profile to insure you have a valid ORACLE_HOME or you can enter the command from OS/390 UNIX System Services:

```
echo $ORACLE_HOME
```

When we did this, the response was correct. The response was:

```
/orauss
```

which was the same value we entered on the install panel `ORINIP50` during the database installation.

We ran the `customize.sh` script and answered yes to the `oem` prompt. The information requested was enter at the prompt:

```
Customize: Now it is time to set up support for the OEM Backup Utility.
Do you want to continue [ Y / N(default) ] ?
 ===> Y
```

At the next prompt we took the default (Enter):

```
Customize: (OEM Backup Utility) Please specify the dataset high-level
qualifiers for your Oracle8 for OS/390 database [ OSDI8.ORA817 ] ?
```

The next prompt asks if all the necessary environmental variables should be added to the user's `.profile`. We answered yes.

```
Do you like those changes to be applied to your .profile file
     Y          - apply to .profile of current user
     N(default) - do not apply the changes
     O          - apply to other file (if it exists, to append,
                  if not, create the file to use it later).
 ===> Y
```

After our .profile was updated, it contained the following information:

```
************************LINES DELETED**********************************

# ======================================================================
# End of c89/cc/c++ customization section
# ======================================================================

#sets for Oracle environment
export ORACLE_HOME=/orauss
export TWO_TASK=OSDX
                                                  ##agent##
                                                  ##agent##
#  the following lines generated during           ##agent##
#  customization of Intelligent Agent             ##agent##
                                                  ##agent##
                                                  ##agent##
export PATH=$PATH:/orauss/bin:                    ##agent##
export ORANLS=/orauss/ocommon/nls/admin/data      ##agent##
export ORA_NLS=$ORANLS                            ##agent##
export ORA_NLS33=$ORANLS                          ##agent##
export TNS_ADMIN=$ORACLE_HOME/network/admin       ##agent##
export NET_ADMIN=$TNS_ADMIN                       ##agent##
export ORATAB=$TNS_ADMIN/oratab                   ##agent##
export LIBPATH=$LIBPATH:/orauss/lib               ##agent##
export MVS_ORA_HOME=OSDI8.ORA817                  ##agent##
export MVS_CMD_HOME=OSDI8.ORA817.CMDLOAD          ##agent##
export MVS_SQL_HOME=OSDI8.ORA817.SQL              ##agent##
export MVS_TMP_HOME=OSDI8.ORA817.TEMP             ##agent##
export MVS_REX_HOME=OSDI8.ORA817.REXX             ##agent##
```

Figure 34. Example of OS/390 UNIX System Services profile updated for the Intelligent Agent

The `customize.sh` script copied the following four files into the $ORACLE_HOME/network/admin directory:

- tnsnames.ora
- oratab
- snmp_rw.ora
- listener.ora

This version is different in that two files, db_s390.ora and snmp_mcs.ora are no longer used. If any of the files become unusable during the customizing process, you can get clean copies from the $ORALE_HOME/network/admin/sample directory.

During our installation, the files listed above were not copied from the sample directory. We needed to use the `cp` command to do the copy manually. We opened a TAR on this problem. At the time of this writing, the TAR was not resolved.

8.2 Configuring the Intelligent Agent and Data Gatherer

All the configuration steps are completed in OS/390 UNIX System Services. The first file we configured was tnsnames.ora. There are some differences from earlier versions. The OSDI version uses Net8 drivers for communications, as opposed to SQL*Net that was used in MPM versions.

In the past, to set up the cross-memory drivers for local database access, the connect string looked like:

```
sqlplus system/manager@W:osdi001
```

where the `@W` selected the cross-memory drivers. This was true for SQL*Plus in OS/390 UNIX System Services and for SQL*Plus when run from the TSO command line. If you did not include the `@W`, the communications path was through TCPIP and SQL*Net. This could cause unnecessary overhead and potential performance problems. If SQL*Net was not running, then you would not be able to connect to the database and would receive a `NO TNS LISTENER` message.

8.2.1 Configuring tnsnames.ora

To be consistent with Oracle on other platforms, a new entry must be made in the tnsnames.ora file that identifies a database service for cross-memory services. The reasons for this are not clearly documented in the Oracle Enterprise Manager Intelligent Agent and Data Gatherer chapter in the *Oracle8i Enterprise Edition with OSDI System Administration Guide*. They are, however, well documented in the *Oracle8i Enterprise Edition with OSDI OS/390 Users Guide*.

We changed the tnsnames.ora entries to the following:

```
OSDX =
 (description =
   (address=(protocol=XM)(subsys=OSD1)(service=OSDI001))
 )
OSDI001 =
 (description=
   (address_list = (address=(protocol=tcp)(host=MVS80)(port=1577)))
   (connect_data = (sid=OSD1))
 )
```

The first entry, OSDX, is used to access the database locally through cross-memory services. The protocol entry must be XM for cross-memory services. The host parameter in our case is MVS80. Either of the following two commands can be used to determine your hostname. The first is hometest, which is supplied by OS/390. Running hometest from the TSO command line should produce output that looks similar to Figure 35.

```
 Running IBM MVS TCP/IP CS V2R7 TCP/IP Configuration Tester

The TCP/IP system parameter file used will be "SYS1.TCPPARMS(TCPDATA)".
***
The FTP configuration parameter file used will be "SYS1.TCPPARMS(FTPDATA)".

 TCP Host Name is: MVS80.SPC.IBM.COM

 Using Name Server to Resolve MVS80.SPC.IBM.COM
 Jobname of Caller:        IBMU10
 TCP Host Name:            MVS80
 Domain Origin:            SPC.IBM.COM
 Jobname of TCP/IP:        TCPIP
 Communicate Via:          UDP
 OpenTimeOut:              5
 MaxRetrys:                1
 NSPort:                   53
 NameServer Jobname:       NAMESRV
 NSInternetAddress(.1.) := 9.97.32.1
 Data Set Prefix used:     TCPIP

 Resolving Name:           MVS80.SPC.IBM.COM
 Result from InitResolver: OK
 Building Name Server Query:

 * * * * * Beginning of Message * * * * *
****************************LINES DELETED*********************************
 Query Id:                 1
 Flags:                    0000 0001 0000 0000
 ***
* * * Hometest Successful* * *

 ***
```

Figure 35. Output of hometest command

You only need to use the hostname, MVS80 in our case. You do not need to use the domain name, SPC.IBM.COM .

Alternatively, the hostname command can be used. It will return one line containing your hostname. The source code and instructions to implement hostname in OS/390 UNIX System Services can be found at:

http://www.s390.ibm.com/products/oe/bpxaltoy.htm

The subsys parameter we used is OSD1, which is what we named our OSDI subsystem, and the service is the database service name. With the entries as such we connected to the database with cross-memory services using the following connect string:

```
sqlplus system/manager@osdx
```

Oracle has a specific search order to resolve the service name, OSDX in this case, to select the correct database, and then provide the connection to it. Check what the tnsnames.ora entry is. The first place Oracle looks is in the .profile and the

TWO_TASK variable. If an entry is found there, then Oracle will try to resolve that from an entry in the tnsnames.ora, otherwise it searches the tnsnames.ora directly for the entry. What this means is that if the variable TWO_TASK looks like this:

```
export TWO_TASK=OSDX
```

The OSDI001 database can be accessed by entering SQL*Plus:

```
sqlplus system/manager
```

Because the TWO_TASK is set to OSDX, Oracle will find the OSDX entry in tnsnames.ora and connect to that database service.

The OSDI001 entry is used to access the database through TCPIP and TNS. While this is mostly to connect to databases off this platform, this entry is needed for the Intelligent Agent since it must have a TCPIP connection to the database.

There are two other entries in tnsnames.ora that are important to the Intelligent Agent and how it is accessed from the Oracle Enterprise Manager. Those are the dbnsnmp1 and dbsnmp2 entries. The only change we needed to make to it is the hostname. Our entries were:

```
dbsnmp1=(description=(address=(protocol=tcp)(host=MVS80)(port=1748)))
dbsnmp2=(description=(address=(protocol=tcp)(host=MVS80)(port=1754)))
```

The dbsnmp entries are used by TNS to listen for incoming requests. This is why there must be a TCPIP entry in the tnsnames.ora even though the Intelligent Agent is on the same platform as the database. It might be thought that cross-memory connection would be used, but not for the Intelligent Agent. TCPIP is used so as to be consistent across platforms.

Ports 1748 and 1754 are registered and granted to Oracle by the Internet Assigned Number Authority (INANA). Do not change these port addresses.

Port 1748 listens for incoming requests and is used in the discovery services.Changing this port will cause auto-discovery to not work. Port 1754 is used by the agent to accept RPCs. If this port is changed, the OEM Console will not be able to detect the intelligent agent.

8.2.2 Configuring listener.ora

The entries in this file are straightforward. We used this to identify our hostname and the port number of the database listener. We then created a listener list of all the databases that we will connect to with this listener. The last entries we made were to identify where the trace and log files go. This is generally the ORACLE_HOME/network/admin/trace directory. Our listener.ora was configured as shown in Figure 36 on page 94.

```
###
#  OE390 listener
###

#
#  This is the definition of the listener named LISTENER.
#  Instead of dummy lines containing % sign enter actual database
#  parameters.  For instance, if your database name is ABCD and it
#  is configured for port 1812, add the following line:
#
#     (ADDRESS=(PROTOCOL=TCP)(HOST=ABCD)(PORT=1812))
#
LISTENER= (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=TCP)(HOST=MVS80)(PORT=1577))
)

#
#  This is the list of SIDs for listener LISTENER. You should
#  specify one line for each SID that this listener will serve.
#  Enter the actual path name in place of %ORACLE_HOME% and
#  the database SID in place of %DB%.
#

# configured for OSD1 - 12/05/00 djd

SID_LIST_LISTENER= (SID_LIST=
     (SID_DESC=(ORACLE_HOME=/orauss) (SID_NAME=OSD1))
)


STARTUP_WAIT_TIME_LISTENER=1
CONNECT_TIMEOUT_LISTENER=0
TRACE_LEVEL_LISTENER=0
TRACE_DIRECTORY_LISTENER=/orauss/network/admin/trace
TRACE_FILE_LISTENER=lis.trc
LOG_DIRECTORY_LISTENER=/orauss/network/admin/trace
LOG_FILE_LISTENER=tns.log
OPS_DOLLAR_LOGIN_ALLOWED=(OCOO)
```

Figure 36. Listener.ora example

The only other parameter worth noting is `TRACE_LEVEL_LISTENER=0`. If there are problems in getting anything in this area to work correctly, change this to `TRACE_LEVEL_LISTENER=16` and a trace will be provided to help diagnose the problem. There are other levels of tracing, but 16 provides the most information and is generally what Oracle Support will ask to see.

8.2.3 Configuring snmp_rw.ora

There are two files, snmp_rw.ora that can be configured and snmp_ro.ora that is created from the snmp_rw.ora entries. Do not change entries in the snmp_ro.ora file. If a problem is suspected in that file, use the `rm` command to delete the file. It will be recreated the next time the Intelligent Agent is started. There are only four entries to be concerned with. The location of the two trace and log files and the `dbnsmp` entries at the beginning of the file. Change only the `host` parameter. We did not change anything else in these entries. Figure 37 on page 95 shows what our snmp_rw.ora file looked like.

```
dbsnmp.address=(address=(protocol=tcp)(host=MVS80)(port=1748))
dbsnmp.spawnaddress=(address=(protocol=tcp)(host=MVS80)(port=1754))
nmi.secure_only = FALSE
nmi.register_with_names=false
nmi.trace_level = 0
nmi.trace_directory = /orauss/network/admin/trace
nmi.trace_file = agent.trc
nmi.log_directory = /orauss/network/admin/trace
nmi.log_file = agent.log
snmp.contact.listener = ""
snmp.index.listener = 1
snmp.contact.OSD1 = ""
snmp.index.OSD1 = 2
```

Figure 37. SNMP_RW.ORA sample file

In a fashion similar to the listener.ora file, if there are problems and a trace is needed, change the `nmi.trace_level=0` to `nmi.trace_level=16` to create the traces.

8.2.4 Configuring oratab

The last file to configure is oratab. There is only one entry in this file per database that will be supported. This entry determines what databases to start when the OEM is started. Generally this is set to `N` so that no databases on OS/390 are started. The entry from our oratab is:

```
OSDI001:/orauss:N
```

8.3 Starting the Intelligent Agent

After we were as certain as we could be that the Intelligent Agent was correctly configured, we tried to start it.

The Intelligent Agent is started as follows from OS/390 UNIX System Services:

```
lsnrctl dbsnmp_start
```

After entering this you will get the following message:

```
LSNRCTL for OE390: Version 8.1.7.0.50 - Production on 13-DEC-2000 05
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
```

However, this does not mean that the Intelligent Agent started successfully. To ascertain the status of the Intelligent Agent, wait a minute or so after starting the agent and enter the following:

```
lsnrctl dbsnmp_status
```

If the agent started successfully, it will issue the following message:

```
LSNRCTL for OE390: Version 8.1.7.0.50 - Production on 13-DEC-2000 05:06:43
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
The db subagent is already running.
```

If the agent did not start the last line of the message will state the following:

```
LSNRCTL for OE390: Version 8.1.7.0.50 - Production on 13-DEC-2000 05:10:32
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
The db subagent is not started.
```

To ensure the agent is running correctly once started, we did a `tnsping` to three addresses (as advised by the *Oracle8i Enterprise Edition with OSDI System Administration Guide)*. One is the system database (or databases) that will be supported by this Intelligent Agent. The other two addresses that we pinged using `tnsping` are:

```
dbsnmp1
dbsnmp2
```

A successful ping will return the following message:

```
IBMU10:/orauss/network/admin>tnsping dbsnmp1

TNS Ping Utility for OE390: Version 8.1.7.0.50 - Production on 13-DEC-2000
06:53:56
(c) Copyright 1997 Oracle Corporation.  All rights reserved.
Attempting to contact (address=(protocol=tcp)(host=MVS80)(port=1748))
OK (300 msec)
```

If you do not get this message for both `dbsnmp1` and `dbsnmp2`, the Intelligent Agent will not autodiscover the databases you want to support with OEM. The OEM tools must be used by logging directly into the database with the tools.

To stop the agent, use the command:

```
lsnrctl dbsnmp_stop.
```

There are two files to look at to verify that configuration was done correctly. One file is the $ORACLE_HOME/network/agent/services.ora. It will be an empty file unless your installation is configured correctly. If the agent starts correctly, the file should like Figure 38.

```
MVS80=(ORACLE_NODE, MVS80, MVS80, (PLATFORM=(osName=OS/390)(osVersion=07.00)))

listener_MVS80 = (ORACLE_LISTENER, MVS80, (ADDRESS_LIST =
(ADDRESS=(PROTOCOL=TCP)(HOST=MVS80)(PORT=1577))))

OSD1 = (ORACLE_DATABASE, MVS80, (description=(address_list =
(address=(protocol=tcp)(host=MVS80)(port=1577)))(connect_data = (sid=OSD1) )),
listener_MVS80)
```

Figure 38. SERVICES.ORA on successful start of the Intelligent Agent

The hostname, port, operating system, and services that the Intelligent Agent will communicate with are identified. If there is a problem starting the agent and there is any incorrect information, do not change it. Erase the file instead. It is recreated every time the agent is started successfully.

When the agent starts successfully, the snmp_ro.ora file is created. If the agent starts successfully, the snmp_ro.ora should look similar to Figure 39 on page 97.

```
snmp.visibleservices = (listener, OSD1)
snmp.shortname.listener = LISTENER
snmp.longname.listener = listener_MVS80
snmp.configfile.listener = /orauss/network/admin/listener.ora
snmp.oraclehome.listener = /orauss
snmp.SID.OSD1 = OSD1
snmp.oraclehome.OSD1 = /orauss
snmp.address.OSD1 = OSD1
snmp.remoteaddress.OSD1 = (description=(address_list = (address=
(protocol=tcp)(host=MVS80)(port=1577)))(connect_data = (sid=OSD1) ))
ifile = /orauss/network/admin/snmp_rw.ora
```

Figure 39. SNMP_RO.ORA on successful start of Intelligent Agent

Hostname, service name, sid, and port number can be identified. If any of this is incorrect, do not change it. Simple remove the file; it will be recreated the next time the agent is started.

Appendix A. Cloning/customizing an IBM Web server on OS/390

The definitive guide for the HTTP server installation process is the *IBM HTTP Server for OS/390: Planning, Installing and Using*, SC31-8690. We will not duplicate that information here. Instead we will summarize some of the key steps.

There are several steps to customize a Web Server. The preliminary steps are:

1. Do the SMP/E installation of the HTTP server for OS/390. (If you are installing in the SYSPLEX, this work has already been done).
2. Define RACF profiles, RACF groups, RACF user IDs, and RACF surrogate IDs to support the Web server environment. (If you are installing in the SYSPLEX, this work has already been done).
3. Install the latest Java Virtual Machine for OS/390 from:

   `http://www.s390.ibm.com/java`
4. Edit /usr/lpp/internet/sbin/setup.sh, and point JAVA_HOME to where you have the JDK installed. For example:

   JAVA_HOME=${JAVA_HOME:-</usr/lpp/java18/J1.1>}
5. Run the script /usr/lpp/internet/sbin/setup.sh. This shell script will create httpd.conf and httpd.envvars files in /usr/lpp/internet/etc.

Once the installation tasks for the default HTTP server for OS/390 have been completed and some basic configuration done, you should clone this environment before doing any more customization. This is described in A.9, “Cloning Web servers” on page 104.

A.1 Customizing your HTTP Web server

The following steps should always be run when building/cloning a Web server:

1. You should have already run /usr/lpp/internet/sbin/setup.sh to create default httpd.conf and httpd.envvars files in the /etc directory.
2. Allocate a new HFS data set for your Web content. Allocate a 20-25 cylinder HFS (call it OMVS.systemID.WEB.yyy on the SYSPLEX and OMVS.WEByyy.HFS on the non-SYSPLEX systems). Mount it at /web/yyy where yyy is a name that corresponds with the server you are cloning, like apple, oracle, suf or peg. Mount this HFS to a mount point such as /web/oracle/

   ```
   mkdir -p /web/oracle
    /samples/mountx /web/oracle OMVS.SC48.WEB.ORAC
   ```
3. Create the following directories:

/web/oracle/pub	Default directory containing your HTML and other files
/web/oracle/pub/images	Default directory containing your GIF files
/web/oracle/our-cgi	Default directory containing your CGI programs
/web/oracle/reports	Directory containing the server reports
/web/oracle/logs	Directory containing the server logs
/web/oracle/servlets	Directory containing your servlets

4. Issue the following command to change the permission bit settings for all the directories that you have created:

   ```
   chmod -R 755 /web/oracle
   ```

5. Copy /etc/httpd.conf and /etc/httpd.envvars to the /web/yyy directory.

   The following assumes you are configuring a server called ORAC.

   ```
   cp /usr/lpp/internet/etc/httpd.conf    /web/oracle/
   cp /usr/lpp/internet/etc/httpd.envvars  /web/oracle
   ```

6. Edit the /web/oracle/httpd.conf file and change the ServerRoot statement to /web/oracle.
7. Change UserID %%CLIENT%% to UserID PUBLIC

   This assumes that you are using an OS/390 (RACF) user ID of PUBLIC as your default access user ID, which we are in the SYSPLEX.

8. Change the PidFile statement to /web/oracle/httpd-pid
9. Change the logging and reporting statements to the appropriate directories:

   ```
   AccessLog        /web/oracle/logs/httpd-log
    AgentLog        /web/oracle/logs/agent-log
    ReferLog        /web/oracle/logs/referer-log
    ErrorLog        /web/oracle/logs/httpd-errors
    CgiErrorLog     /web/oracle/logs/cgi-error
   AccessLogArchive          purge
    ErrorLogArchive           purge
    AccessLogExpire           5
     ErrorLogExpire            5
   AccessReportRoot           /web/oracle/reports
     ReportDataArchive         purge
    ReportDataExpire          5
   ```

10. Edit the httpd.conf file and modify the Exec and Pass statements as shown below:

    ```
    #
     Exec     /our-cgi/*     /web/oracle/our-cgi/*
     Exec /cgi-bin/*     /usr/lpp/internet/server_root/cgi-bin/*
     Exec /admin-bin/*    /usr/lpp/internet/server_root/admin-bin/*
     Exec /Docs/admin-bin/*   /usr/lpp/internet/server_root/admin-bin/*
     #     These are the pass rules for server administration
     #
     Pass /icons/*       /usr/lpp/internet/server_root/icons/*
     Pass /Admin/*.jpg    /usr/lpp/internet/server_root/Admin/*.jpg
     Pass /Admin/*.gif    /usr/lpp/internet/server_root/Admin/*.gif
     Pass /Admin/*.html   /usr/lpp/internet/server_root/Admin/*.html
     Pass /Docs/*       /usr/lpp/internet/server_root/Docs/*
     Pass /reports/javelin/*   /usr/lpp/internet/server_root/pub/reports/javelin/*
     Pass /reports/java/*  /usr/lpp/internet/server_root/pub/reports/java/*
     #Pass    /reports/*     /usr/lpp/internet/server_root/pub/reports/*
     Pass     /reports/*     /web/oracle/reports/*
     Pass /img-bin/*     /usr/lpp/internet/server_root/img-bin/*
     Pass     /images/*     /web/oracle/pub/images/*
     # *** ADD NEW PASS RULES HERE ***
     #Pass     /*           /usr/lpp/internet/server_root/pub/*
     Pass     /Server/*   /usr/lpp/internet/server_root/pub/*
     Pass     /*            /web/oracle/pub/*
    ```

    This last Pass statement will force the Web server to search in your /web/oracle/pub directory for everything but the IBM-provided default content.

11. Edit the file /web/oracle/httpd.envvars. Make sure you are pointing to the correct directories, that is, /usr/lpp/internet/ and /usr/lpp/java18.

A.2 Create a default home page for your Web server

The following HTML example shows you how to set up your own home page using the settings shown in the previous section, and still be able to access the remote administration forms as before.

1. This file should be called index.html or Welcome.html and should be placed in /web/oracle/pub:

```
<html><head>
<title>WEBORAC and the HTTP Server for OS/390</title>
</head><body bgcolor="#FFFFFF">
<h1>Welcome to my home page </h1>
```

   This is the HTTP server for OS/390 serving WEBORAC pages and applets.

   <hr>

   Follow this link to access the

```
<a href="/admin-bin/webexec/cfgstart.html"> Server Administration</a>
</html>
```

2. Change the permission bits of this file to 755. Either use the ISHELL or issue the `chmod` command from the OMVS shell.

A.3 Create/update a started procedure for your Web server

1. Update the started procedure to point to the new httpd.conf and httpd.envvars files. Use the -p parameter to pass a unique port number for your Web server. An example follows:

```
//WEBORAC PROC P1='-B',
// P2='-r /web/oracle/httpd.conf',
// P3='-p 99 ',
// LEPARM='ENVAR("_CEE_ENVFILE=/web/oracle/httpd.envvars")'
//*
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
//   PARM=('&LEPARM/&P1 &P2 &P3')
//SYSIN    DD DUMMY
//OUTDSC   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

2. Define a RACF STARTED profile for this Web server, setting the associated user ID to WEBSRV and group to IMWEB. If running in a SYSPLEX, check if there is already a WEB*.* (G).

3. Make sure the following data sets are defined to the RACF PROGRAM * profile:

   - SYS1.LINKLIB
   - CEE.SCEERUN
   - CBC.SCLBDLL

   In our SYSPLEX, these data sets were already defined.

4. Start the Web server. You should see the following message when your Web server is ready:

```
IMW3536I SA 184549402 0.0.0.0:99 * * READY
```

Now your Web server is up as a HTTP server. To run JAVA servlets you will need to customize the WebSphere Application Server.

A.4 WebSphere customization

1. If you haven't installed the Java Development Kit (JDK) yet, do it now.
2. After the JDK is installed, enter the following command from the OMVS shell to set the JAVA_HOME environment variable.

   ```
   export JAVA_HOME=/usr/lpp/java18/J1.1
   ```

   Note: /usr/lpp/java18/J1.1 is the path where you install the JDK.
3. Create a directory called /was in the root.
4. Allocate 20 cylinder HFS (call it OMVS.systemID.WAS.yyy) SYSPLEX and OMVS.WASORAC.HFS on the non-SYSPLEX systems) and mount it at /was/yyy where yyy is a name that corresponds with the server you are cloning, like /was/apple, /was/oracle or /was/peg.
5. Issue the following command to change the permission bit settings for all the directories that you have created:

   ```
   chmod -R 755 /was/oracle
   ```
6. Go into the /usr/lpp/WebSphere/AppServer/config directory and run the makeserver.sh command:

   ```
   cd /usr/lpp/WebSphere/AppServer/config
   makeserver.sh /was/oracle/a  /web/oracle/httpd.conf   /usr/lpp/java18/J1.1
   You will see the following messages:
   Start - OS/390 WebSphereAS 1.2 makeserver: Monday 02/21/00 02:26:46 PM
   Starting utility 'updateproperties' ....   - 14:26:56 on 02-21-2000
   App Server Root Directory : /was/oracle/a/
   App Server Log File        : /was/oracle/a/logs/updateproperties.log
   Properties in all Files successfully updated - 14:26:56 on 02-21-2000
   Ended Successfully - OS/390 WebSphereAS 1.2 makeserver: Monday 02/21/00
   02:26:57 PM
   ```
7. Check /was/oracle/a/logs/updateproperties.log for any errors.

A.5 WebSphere httpd.conf updates

1. Edit /web/yyy/httpd.conf and comment out the following lines to disable the Proxy server support. (Most Web servers we build do not require proxy support).

   ```
   #   ====================================================
   #   *** OS/390 Web Traffic Express Support directives ***
   #   ====================================================
   #ServerInit /usr/lpp/internet/bin/Jav_dll.so:Javelin_init
   #Service    /cgi-bin/dogc.icapi   /usr/lpp/internet/bin/Jav_dll.so:doGC
   #PreExit
   /usr/lpp/internet/bin/Jav_dll.so:Javelin_preFilter
   #Enable     ICSERRORLOG
   /usr/lpp/internet/bin/Jav_dll.so:Javelin_errorLog
   ```

2. Add the following right after the Web Traffic Express text that you just commented:

```
#
#  WebAS 1.2 - following added by Rich
#
ServerInit  /was/oracle/a/lib/libadpter.so:AdapterInit /was/oracle/a
Service     /*.shtml     /was/oracle/a/lib/libadpter.so:AdapterService
Service     /*.jhtml     /was/oracle/a/lib/libadpter.so:AdapterService
Service     /*.jsp       /was/oracle/a/lib/libadpter.so:AdapterService
Service     /servlet/*   /was/oracle/a/lib/libadpter.so:AdapterService
ServerTerm  /was/oracle/a/lib/libadpter.so:AdapterExit
```

3. Comment out the following line:

```
#Service    /servlet/*
/usr/lpp/WebSphere/AppServer/lib/libadpter.so:AdapterService
```

A.6 WebSphere was.conf updates

The /was/oracle/a/properties/was.conf file is the file that you use when making updates for CLASSPATH, LIBPATH, PATH, and if you need to do any servlet tracing. After running makeserver.sh, all configuration files should be updated and ready to go. If you want to turn on servlet tracing, see the section on testing your web server later in this procedure.

A.7 Program control bit - Java virtual machine

Certain JDK dll libraries also need to have the PROGRAM CONTROL bit turned on.

```
cd /usr/lpp/java18/J1.1/lib/mvs/native_threads
extattr +p *.*
ls -E
```

Note: If you receive:
FOMF0303I libjava.a: chattr() error: rv=-1, errno=8B, rsn=0924041B
this indicates you are not authorized to execute this command. Make sure you are executing the `extattr` command from a superuser ID that is permitted to the BPX.FILEATTR.PROGCTL RACF FACILITY class.

A.8 Testing your Web server

1. Restart your Web server. You should see the following message when your Web server is ready:

```
IMW3536I SA 187488759 0.0.0.0:99 * * READY
```

2. Go to your Web browser and select the URL of your Web server.
3. To test that WebSphere is working, type in the following URL to execute a servlet:

```
http://wtsc58oe:99/servlet/HelloWorldServlet
```

The HelloWorldServlet should execute and echo back the words ” Hello World.”

If you get a message like the following:

```
ICH408I USER(PUBLIC  ) GROUP(SYS1    ) NAME(########
/was/oracle/servlets/HelloWorldServlet.servlet
CL(DIRSRCH ) FID(01E2C2D6E7F1F5002E0B000000000003)
INSUFFICIENT AUTHORITY TO LOOKUP
ACCESS INTENT(--X)  ACCESS ALLOWED(GROUP ---)
```

you need to check the permission bit settings of all the directories in the path. Make sure the permission bit settings are at least 755 for all directories in the path. If you still can't figure out the problem, go into the was.conf file and make the following changes to turn on tracing/logging.

- Change `ncf.native.logison=false`
  to `ncf.native.logison=true`
- Change `ncf.jvm.stdoutlog.enabled=false`
  to `ncf.jvm.stdoutlog.enabled=true`
- Change `ncf.native.os390.debug=0`
  to `ncf.native.os390.debug=1`
- Change `trace.enable=false`
  to `trace.enable=true`

After the changes are made, run /was/oracle/a/config/updateproperties. This will copy the updates you just made in was.conf to the /was/oracle/a/properties/server/servlet/servletservice/jvm.properties file. Then execute the commands:

```
cd /was/oracle/a/config
updateproperties /was/oracle/a
```

You will see the following messages:

```
Starting utility 'updateproperties' ....   - 15:56:59 on 02-21-2000
App Server Root Directory : /was/oracle/a/
App Server Log File        : /was/oracle/a/logs/updateproperties.log
Properties in all Files successfully updated - 15:56:59 on 02-21-2000
```

Stop and restart your Web server and try the HelloWorld servlet again. Check the following log files for any hints:

```
/was/oracle/a/logs/trace.log
/was/oracle/a/logs/native.log
/was/oracle/a/logs/ncf.log
```

A.9 Cloning Web servers

It is possible to clone a working HTTP and WAS 1.2 server by dumping the /web and /was HFS data sets for the working Web server and restoring and renaming them to new copies. After the copy, you will need to update the /web/xxx/httpd.conf and /web/xxx/httpd.envvars files. The following WAS 1.2 changes also must be performed in the cloned copy:

1. Edit /was/xxx/a/properties/was.conf - make changes to update the name of the cloned server.
2. Run `updateproperties`, which will update
   `/was/xxx/a/properties/server/servlet/servletservice/jvm.properties`

3. For some reason, `updateproperties` doesn't update the following statements in /was/xxx/a/properties/server/servlet/servletservice/jvm.properties and they need to be edited by hand:

   ```
   server.root=/was/oracle/a
   ncf.native.httpd.cnf.path=/web/oracle/httpd.conf
   ```

Note: The preferred way of getting new classes in jvm.properties or to turn on tracing is to make your changes in was.conf, then run `updateproperties`. It not only updates jvm.properties but other files in /was/xxx/a/properties/server/servlet/servletservice/

Appendix B. Using GTF Trace, ESTAT/BSTAT and RMF

The steps to run a GTF trace are:

- Start and stop the Oracle subsystem to clean out memory.
- Start the Oracle database and the listener.
- Start gtfora.gtf to have the trace running (in sys1.proclib).

  You will have to reply /r xxx,u where xxx is the number of the message.
- Submit the query to scan the partitioned table ORACLE2.JCL.CNTL(KASELECT8).
- Stop the gtftrace /p gtf.
- Run the job GTF print in ORACLE2.JCL.CNTL(GTFPRINT).
- Look at the output SD H PRE ORAG*
- Send to VM by overtyping the local line with wtscpok.kathryna.

  Then print with rotate of 90 degrees.

B.1 JCL to run GTF trace

```
//GTFORA  PROC MEMBER=GTFORA
//* 'S GTFORA.GTF,MEMBER=GTFORA
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
//  TIME=1440,REGION=2880K
//IEFRDER DD   DSNAME=ORACLE1.GTF.TRACE,DISP=OLD
//SYSLIB  DD   DSNAME=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

B.2 JCL to run GTFPRINT

```
//ORAM14G  JOB (0000,OR),'ORACLE INSTALL',CLASS=A,   FY=ORACLE2
 //           MSGCLASS=X,PRTY=15,MSGLEVEL=(1,1),NOTIFY=ORACLE2
 //*     REFER RUSSELL.JCL.CNTL(GTFPRINT)
 //PRINT EXEC PGM=IKJEFT01,REGION=7M
 //GTFIN    DD    DSN=ORACLE1.GTF.TRACE,DISP=SHR
 //*TFIN    DD    DSN=RUSSELL.GTF.TRACE,DISP=SHR
 //*TFIN    DD    DSN=RUSSELL.CDATRUST.GTF,DISP=SHR
 //SYSPRINT DD SYSOUT=*
 //SYSTSPRT DD SYSOUT=*
 //IPCSPRNT DD SYSOUT=* OUTLIM=20000
 //IPCSDDIR DD DSN=RUSSELL.DEBUG,DISP=SHR
 //*PCSPARM DD DSN=RUSSELL.JCL.CNTL,DISP=SHR
 //*YSPROC  DD DSN=SYS1.ESA.CLIST,DISP=SHR
 //SYSIN    DD DUMMY
 //SYSTSIN  DD *
 IPCS NOPARM
 DROPDUMP FILE(GTFIN)
 SETDEF CONFIRM
 GTFTRACE  FILE(GTFIN) NOTERMINAL PRINT
Y
 //
 GTFTRACE  FILE(GTFIN) NOTERMINAL PRINT JOBNAME(*MASTER*, +
                                                RUSSELL$ )
 GTFTRACE  FILE(GTFIN) NOTERMINAL PRINT JOBNAME(*MASTER*, +
```

```
                                                CNA1TCNX, +
                                                BWK       )
 GTFTRACE  FILE(GTFIN) NOTERMINAL PRINT JOBNAME(*MASTER*, +
                                                 CNP1JICX, +
                                                 CNP1JCNX, +
                                                 CNP1JSEC, +
                                                 CNP1JHPO, +
                                                 CNP1JATI, +
                                                 CNP1JEFI, +
                                                 CNP1JP01, +
                                                 CNP1JA01, +
                                                 CNP1JA02, +
                                                 CNP1JA03, +
                                                 CNP1JA04, +
                                                 CNP1JA05, +
                                                 CNP1JA06, +
                                                 CNP1JA07, +
                                                 CNP1JA08, +
                                                 CNP1JA09, +
                                                  CNP1JA10, +
                                                  CNP1JA11, +
                                                  CNP1JA12, +
                                                  CNP1JA13, +
                                                  CNP1JA14, +
                                                  CNP1JA15, +
                                                  CNP1JA16, +
                                                  CNP1JA17, +
                                                  CNP1JA18, +
                                                   CNP1JA19, +
                                                   CNP1JA20, +
                                                   BWK      )
   Y
//*
```

B.3 Sample Output

```
0 SSCH.... 2468     ASCB.... 00F56400 CPUID... 0002     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F22170 83C0F000 311F3528
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2468
                GMT-03/29/2000 19:15:16.841244  LOC-03/29/2000 14:15:16.8
 0 SSCH.... 2527     ASCB.... 00F56400 CPUID... 0000     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F242E8 83C0E000 311F3B10
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2527
                GMT-03/29/2000 19:15:16.841307  LOC-03/29/2000 14:15:16.8
 0 SSCH.... 2467     ASCB.... 00F56400 CPUID... 0000     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F220E8 83C0F000 1E2B0B10
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2467
                GMT-03/29/2000 19:15:16.841843  LOC-03/29/2000 14:15:16.8
 0 SSCH.... 2626     ASCB.... 00F56400 CPUID... 0002     JOBN.... ORAMPM8
                     CC...... 00       ORB..... 00F2B960 83C0F000 09E9E528
                     OPT..... 14       FMSK.... 08       DVRID... 03
                     BASE.... 2626
```

Appendix C. Special notices

This publication is intended to help those you support an Oracle database on OS/390. The information in this publication is not intended as the specification of any programming interfaces that are provided by Oracle or IBM prooducts. See the PUBLICATIONS section of the IBM Programming Announcement for Oracle and OS/390 for more information about what publications are considered to be product documentation.

Information concerning Oracle's products was provided by Oracle. The material in this document has been produced by a joint effort between IBM and Oracle S/390 Specialists. The material herein is copyrighted by both IBM and Oracle.

IBM makes no warranties regarding Oracle and other non-IBM products. IBM has not tested Oracle and other non-IBM products and cannot confirm the accuracy of perfromance, compatibility, or any other claims relative to non-IBM products. Questions on the capabilities of Oracle and non-IBM products should be addressed to the suppliers of those products.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®	Redbooks
IBM ®	Redbooks Logo
AS/400	RS/6000
AT	S/390
CICS	ServicePac
CT	SP
Current	System/390
DB2	VTAM
IBM.COM	WebSphere
MVS/ESA	XT
Netfinity	400
OS/390	Lotus
RACF	Notes
RMF	NetView

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 113.

- *Experiences with Migrating Oracle to OS/390*, SG24-4981
- *Experiences Installing Oracle Release 3 (8.1.7) for OS/390*, SG24-5973
- *Experiences Installating Oracle8i Release 3 (8.1.7) with OSDI,* SG24-5973

D.2 Other IBM publications

- *OS/390 MVS Planning: Workload Management* GC28-1761
- *IBM HTTP Server for OS/390, Planning, Installing and Using,* SC31-8690

D.3 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `ibm.com/redbooks` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.4 Oracle Publications

These publications are also relevant as further information sources and are available on METALINK or docs.oracle.com:

- *Oracle8i Enterprise Edition for OS/390 with OSDI Installation Guide*, A85481-1
- *Oracle8i Enterprise Edition for OS/390 with OSDI Messages Guide*, A85187-1
- *Oracle8i Enterprise Edition for OS/390 with OSDI System Administration Guide*, A85482-1
- *Oracle8i Enterprise Edition for OS/390 with OSDI User’s Guide*, A85483-1
- *Oracle8i Enterprise Edition for OS/390 Installation Guide*, A86037-1 (for MPM)
- *Oracle8i Enterprise Edition for OS/390 Messages Guide*, A86035-1 (for MPM)

- *Oracle8i Enterprise Edition for OS/390 Systems Administration Guide*, A86038-1 (for MPM)
- *Oracle8i Enterprise Edition for OS/390 User's Guide*, A86036-1 (for MPM)
- *Oracle Intelligent Agent Users Guide* - A67825-01
- *Oracle Enterprise Manager Quick Start Guide Doc Note:107241*
- *Oracle Enterprise Manager Configuration Guide*, A67823-01
- *Oracle Enterprise Manager Administrators Guide* A67820-01
- *Oracle Backup and Recovery Guide*
- *Oracle ProC Precompilers*
- *Oracle Programmer's Guide to the Pro*C/C++ Precompiler*, A76942
- *Oracle for OS/390 Installation Guide* - A68790-01
- *Intelligent Agent Users Guide* A67826

D.5 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.oracle.com/support/` for Metalink
- `http://docs.oracle.com/` for publications
- `http://www.mvsoraclesig.org` for Oracle MVS Users group

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `ibm.com/redbooks`

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

S

T

U

V

W

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at `ibm.com/redbooks`
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Document Number Redbook Title	SG24-5972-00 Managing Oracle 8.1.7 on OS/390
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	
	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. `ibm.com/privacy/yourprivacy/`

IBM Redbooks

Managing Oracle 8.1.7 on OS/390

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Managing Oracle 8.1.7 on OS/390

Experiences with Oracle 8i for OS/390

Using WebSphere and UNIX System Services functions

Managing Oracle on OS/390 with Workload Manager

This IBM Redbook describes some aspects of how to manage Oracle 8.1.7 and above on OS/390, such as:

- Using Workload Manager to manage Oracle workloads
- Setting up an Oracle JDBC driver for a Web client
- Using the Oracle pre-compilers and utilities on OS/390 and OS/390 UNIX System Services
- Using Recovery Manager (RMAN) with Oracle8i on OS/390
- Using partitioned tables to improve performance
- Using Oracle Enterprise Manager

The book is based on real-world experiences gained during installation of these products at:

- The IBM/Oracle EMEA Joint Solution Center, Montpellier, France
- The IBM/Oracle International Competency Center, San Mateo, California
- The IBM ITSO S/390 Center in Poughkeepsie, New York
- The Oracle Advanced Technology Center, Reston, Virginia

The use of Workload Manager, the OS/390 UNIX System Services pre-compilers and utilities are only available on Oracle 8.1.7 with OSDI and higher. Facilities which are available on earlier versions of Oracle8 on OS/390 are mentioned throughout the book, but the primary focus is the newest capabilities of the product at the time of writing.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5972-00 ISBN 0738418390