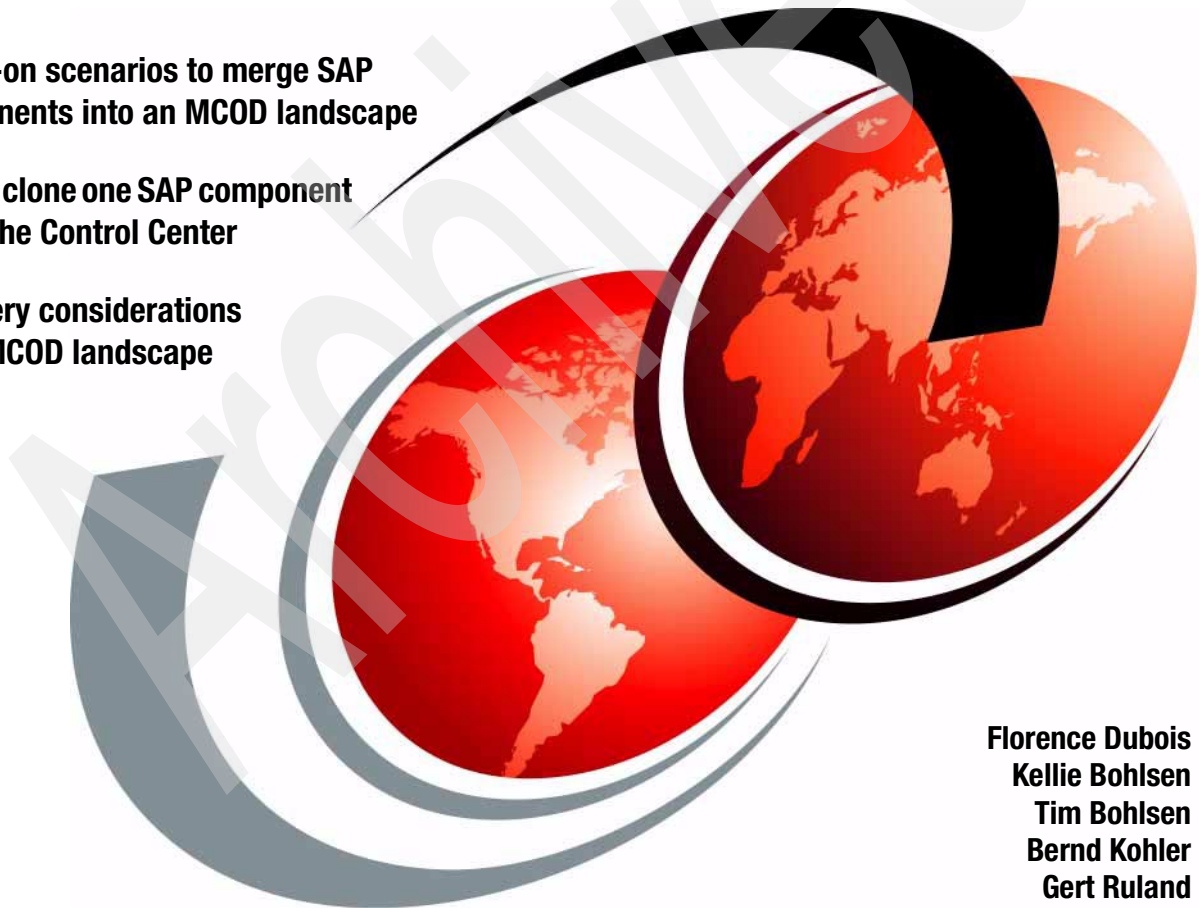


SAP on DB2 Universal Database for OS/390 and z/OS: Multiple Components in One Database (MCOD)

Hands-on scenarios to merge SAP components into an MCOD landscape

How to clone one SAP component using the Control Center

Recovery considerations in an MCOD landscape



Florence Dubois
Kellie Bohlsen
Tim Bohlsen
Bernd Kohler
Gert Ruland



International Technical Support Organization

**SAP on DB2 Universal Database for OS/390 and
z/OS: Multiple Components in One Database
(MCOB)**

February 2003

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (February 2003)

This edition applies to SAP Basis Release 4.6C, 4.6D, 6.10, and 6.20, as well as all SAP products based on these SAP Basis Releases, for use with IBM DB2 Universal Database for OS/390 and z/OS.

© Copyright International Business Machines Corporation 2003. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	ix
Notices	xi
Trademarks	xii
Preface	xiii
The team that wrote this redbook	xiv
Become a published author	xvi
Comments welcome	xvi
Chapter 1. MCOD in a DB2 UDB for OS/390 and z/OS environment	1
1.1 Introduction	2
1.1.1 Motivation	2
1.1.2 Benefits	2
1.1.3 Drawbacks	3
1.1.4 Availability	4
1.2 Technical realization	4
1.2.1 General implementation	4
1.2.2 DB2-specific modifications	5
1.2.3 Independence of components	8
1.2.4 Implementation	9
1.3 Setup options	10
1.3.1 Basic setup considerations	10
1.3.2 Non-data sharing DB2	11
1.3.3 Data sharing DB2	12
Chapter 2. Planning for an MCOD landscape	15
2.1 Planning for MCOD	16
2.1.1 Keep things as separate as possible	16
2.1.2 Size of the DB2 subsystem	16
2.1.3 Number of SAP systems in the MCOD landscape	17
2.1.4 Bufferpool tuning	18
2.1.5 Recovery of the DB2 subsystem	18
2.1.6 Resources for daily tasks	19
2.1.7 License key	19
2.1.8 Checklist	19
2.2 Install directly	20
2.3 Merging two systems	20
2.3.1 Three methods of merging	21

2.3.2 Additional tasks	23
2.3.3 Checklist for merging	23
2.4 Our system configuration	24
Chapter 3. MCOD installation and merge using SAP tools	27
3.1 Introduction	29
3.1.1 SAP Basis Releases 4.6C and 4.6D	29
3.1.2 SAP Basis Releases 6.10 and later	30
3.2 Installing a new component.	30
3.2.1 SAP Basis Releases 4.6C and 4.6D	30
3.2.2 SAP Basis Releases 6.10 and later	30
3.3 Merging components into an MCOD landscape	32
3.3.1 SAP Basis Releases 4.6C and 4.6D	33
3.3.2 SAP Basis Release 6.10 and later	37
3.4 Steps after installation or merge	37
3.5 Minimizing migration down time	38
3.5.1 Incremental migration	38
3.5.2 Merging without moving the data	39
Chapter 4. Merging SAP components without moving data	41
4.1 Considerations for using this procedure	43
4.1.1 Reasons for choosing the merge in place procedure	43
4.1.2 Limitations	43
4.1.3 Tools we used	44
4.2 Planning considerations	44
4.2.1 Decide on source and target DB2 subsystem.	44
4.2.2 Decide on naming conventions for merged system	45
4.2.3 System availability issues	45
4.2.4 System backup and recovery issues.	46
4.2.5 Hardware-based backup options	47
4.2.6 Merge in place procedure checklist.	47
4.3 Merge in place scenario	49
4.4 Preparation steps	51
4.4.1 Source: Create full backup of source system	51
4.4.2 Target: Create full backup of catalog and directory of target	52
4.4.3 Source: Redefine empty tablespaces as DEFINE NO in source	52
4.4.4 Source: Reorganize tablespaces	52
4.4.5 Source and Target: Create and populate metadata tables	53
4.4.6 Source and Target: Define source objects in target system	55
4.4.7 Target: Update metadata tables	59
4.4.8 Target: Stop newly created databases	59
4.4.9 Target: Prepare RUNSTATS JCL	60
4.5 Migration steps	60

4.5.1	Source: Stop all update activity on source system	60
4.5.2	Source: Perform full backup of source DB2 subsystem	61
4.5.3	Source: Start source databases for UT access	61
4.5.4	Source: Execute REPAIR on page set header pages	61
4.5.5	Source: Stop source DB2 subsystem	62
4.5.6	Target: Delete target data sets and rename source data sets	62
4.5.7	Target: Start target databases in target system	63
4.5.8	Target: Execute REPAIR LEVELID on tablespaces and indexes	63
4.5.9	Cold start target system (optional)	64
4.5.10	Target: Execute RUNSTATS	65
4.5.11	Target: Perform IMAGE COPY	65
4.5.12	Configure SAP application server	65
4.6	Post-migration steps	68
4.6.1	Target: Verify access to objects using RSDB2MAS (optional)	68
4.6.2	Target: Alter the space allocations for TS and IX (optional)	68
Chapter 5. Cloning one component out of an MCOD landscape		69
5.1	MCOD cloning: Just another homogeneous system copy	70
5.1.1	Homogeneous system copy (HSC)	70
5.1.2	MCOD cloning	72
5.1.3	Control Center support for HSC and MCOD cloning	72
5.2	Cloning scenario	73
5.3	Using the Control Center cloning wizard	75
5.3.1	System requirements	75
5.3.2	Skill requirements	76
5.3.3	Prepare to create a cloning session	77
5.3.4	Running the Control Center cloning wizard	78
5.3.5	Prepare for submitting the generated JCL	95
5.3.6	XMAP member and MCOD cloning	96
5.3.7	When do you have to regenerate JCL	98
5.3.8	Running the generated JCL	98
5.3.9	Additional considerations	108
Chapter 6. PPT recovery of one system in an MCOD landscape		115
6.1	Different reasons for MCOD usage	117
6.1.1	Use of SAP MCOD for consistency	117
6.1.2	Use of SAP MCOD for consolidation	118
6.2	Recovery scenarios	118
6.2.1	Determination of the scope of SAP components affected	119
6.2.2	Work with the application owners	119
6.2.3	Determining if a DB2 recovery is needed	120
6.3	Scenario: Recovery of one component	121
6.3.1	Description of the cause of error	122

6.3.2	Determination of recovery steps	123
6.4	Additional considerations	125
6.4.1	Application of scenario to other applications	125
6.4.2	Considerations for multiple SAP component recovery	126
6.4.3	Testing in advance	126
Chapter 7.	Computer Center Management System (CCMS) and MCODE	127
7.1	Setup of CCMS in an MCODE landscape	128
7.1.1	Required patches	128
7.1.2	Setup of SAP collector tools saposcol and rfcoscol	128
7.1.3	Central DBA planning calendar	131
7.2	MCODE enhancements	133
7.2.1	Tables and indexes monitor (transaction DB02)	133
7.2.2	Central DBA planning calendar (transaction DB13C)	134
7.2.3	DBA planning calendar (transaction DB13): Backup	135
7.2.4	Backup monitor (transaction DB12)	136
7.2.5	CCMS monitor set (transaction RZ20)	137
7.2.6	Database performance analysis (transaction ST04)	138
Appendix A.	Database layout	145
Overview	146
General remarks	146
R/3 Releases 3.0F, 3.1H, 3.1I, and 4.0B	147
R/3 Releases 4.5A and 4.5B	148
Basis Releases 4.6B, 4.6C, and 4.6D	149
Basis Releases 6.10 and 6.20	149
Rename procedure	150
General procedure	150
Mixed layouts	150
Appendix B.	Merge in place: Working with metadata tables	153
Creating metadata tables	154
DDL for creation of ZMCOddb	154
DDL for creation of ZMCOd_STOGROUP	154
DDL for creation of ZMCOd_DATABASE	155
DDL for creation of ZMCOd_TABLESPACE	156
DDL for creation of ZMCOd_TABLES	157
DDL for creation of ZMCOd_INDEXES	157
Populating metadata tables	159
SETUPSTG	159
SETUPDB	162
SETUPTS	165
SETUPTAB	170
SETUPIX	173

JCL to run the population REXX procedures	177
Moving metadata tables across systems	178
JCL to unload metadata tables on the source system	178
JCL to load metadata tables into the target system	180
JCL to reset copy pending status on tablespaces	181
Working with metadata tables	182
DUPLICDB	182
JCL to run the duplicate resolution REXX procedure	188
Updating metadata tables	189
POSTDDL	189
POSTDDL2	192
JCL used to run the update REXX procedures	195
Appendix C. Merge in place: Defining source objects in target system	197
Generating DDL using DB2 Admin.	198
Invoking DB2 Admin	198
JCLGEN	199
JCLGENDB	205
JCLGENTS	207
JCLGENTB	210
JCLGENIX	212
JCL to run generation REXX procedure and submit the generated JCL	215
Tailoring the output from DB2 Admin	217
TAILORTS	217
TAILORIX	219
TAILORTB	221
JCL to run the tailoring REXX procedures	225
Sample DDL for creating objects	227
Using filler tablespaces to reserve OBIDs	230
Appendix D. Merge in place: Migrating the data	233
Running the migration procedures	234
PDS libraries used	234
Input parameters for all migration procedures	235
Extracting information from the metadata tables	236
\$ZMCDBSR	236
\$ZMCINSR	237
\$ZMCTSSR	238
PQUERYTB	239
Issuing DB2 START and STOP DATABASE commands	241
PDSNDBST	241
Executing REPAIR on page set header pages	244
PREPTSOB	244

PREPINOB	248
Deleting created target data sets and renaming source data sets	248
PVSATSDE	248
PVSAINDE	250
PVSATSAL	250
PVSAINAL	252
Executing REPAIR LEVELID on tablespaces and indexes	253
PREPTSLV	253
PREPINLV	255
Generating DDL for views	255
\$IBMVIST	256
PCREATVI	256
Generating DDL for altering object sizes	259
\$IBMTSSR	259
\$IBMINSR	260
PTEPTSAL	261
PTEPINAL	264
Appendix E. Additional material	265
Locating the Web material	265
Using the Web material	265
How to use the Web material	266
Abbreviations and acronyms	267
Related publications	269
IBM Redbooks	269
Other References	269
SAP Notes	271
Referenced Web sites	272
How to get IBM Redbooks	272
IBM Redbooks collections	272
Index	273

Figures

1-1	MCOD: Database user and schema	5
1-2	MCOD: No data exchange within the database	9
1-3	MCOD landscape with DB2	12
1-4	MCOD and data sharing	13
2-1	Our initial configuration	25
3-1	Installing a new dialog instance in an MCOD landscape	31
3-2	SAPinst: Specifying DB2-specific installation parameters	32
3-3	Merging components with SAP tools: Initial configuration	33
3-4	Merging components with SAP tools: Final configuration	34
4-1	Merge in place scenario: Initial configuration	50
4-2	Merge in place scenario: Final configuration	51
5-1	Cloning scenario: Initial configuration	74
5-2	Cloning scenario: Final configuration	75
5-3	Create a new session in the cloning wizard	79
5-4	Create cloning session wizard: Introduction	80
5-5	Create cloning session wizard: JCL storage	81
5-6	Create cloning session wizard: Source subsystem	82
5-7	Create cloning session wizard: Target subsystem	83
5-8	Create cloning session wizard: Source data sets	84
5-9	Create cloning session wizard: Target data sets	86
5-10	Create cloning session wizard: Copied data sets	88
5-11	Create cloning session wizard: DB2 libraries	89
5-12	Create cloning session wizard: Work Load Manager	90
5-13	Create cloning session wizard: Execution parameters	91
5-14	Create cloning session wizard: Job statements	92
5-15	Create cloning session wizard: Summary (1/2)	93
5-16	Create cloning session wizard: Summary (2/2)	94
5-17	Generated JCL	95
5-18	Modifying the XMAP file for MCOD cloning	97
6-1	Important reference times required for recovery planning	119
6-2	Prior point in time (PPT) recovery scenario: Initial configuration	122
6-3	Steps involved in non-disruptive recovery of one SAP component	125
7-1	Transaction DB2: SAP collector settings	129
7-2	Transaction DB2: Output of the bind of SAP collector	130
7-3	Transaction SM59: Create connection	131
7-4	Transaction SM59: Test connection	132
7-5	Transaction DB13C: Configuration of DBA planning calendar	132
7-6	Transaction DB02: Initial screen	133

7-7	Transaction DB02: Detailed analysis output for indexes	134
7-8	Transaction DB13: Backup	135
7-9	Transaction DB13: Log output	136
7-10	Transaction DB12: List of last successful backups	137
7-11	Transaction ST04: Initial screen	138
7-12	Transaction ST04: Long-running URs	139
7-13	Transaction ST04: Long-running URs (Detail)	140
7-14	Transaction ST04: Statement Cache Statistics	141
7-15	Transaction ST04: Statement Cache Statistics (Details 1/2)	142
7-16	Transaction ST04: Statement Cache Statistics (Details 2/2)	143

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.



This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	FlashCopy®	Redbooks (logo)  ™
DB2®	IBM®	S/390®
DB2 Universal Database™	MVS™	S/390®
DFSMSdss™	OS/390®	z/OS™
DRDA®	QMF™	z/VM™
eServer™	RACF®	zSeries™
 ™	RAMAC®	
Enterprise Storage Server™	Redbooks™	

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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.



Preface

The Multiple Components in One Database (MCOD) feature of SAP enables a reduction in the number of DB2® systems that need to be installed and maintained. This significantly simplifies overall database administration and is considered one of the major DB2 competitive advantages.

This IBM® Redbook will help systems administrators, database administrators, managers, and operation staff to plan, implement, and administer an SAP MCOD landscape with DB2 Universal Database™ (UDB) for OS/390® and z/OS™ as the database management system.

We describe how to merge existing systems into a single DB2 subsystem. Two different methods are developed, each of them addressing different needs. For small-to-medium SAP systems where high availability is not a requirement, we explain how to use SAP tools. For large systems, where the down time needed by SAP standard procedures is not acceptable, we document a technique to merge SAP components without moving the data.

We also provide a cloning procedure using the Control Center. We show how to clone one component out of an MCOD landscape, but this cloning technique is not specific to SAP applications and can apply to any DB2 subsystem.

We address the backup and recovery implications in an MCOD environment, to help database administrators plan accordingly. We also describe how to set up and use the Computer Center Management System (CCMS) in an MCOD landscape.

The book addresses an audience familiar with SAP system requirements, and for some specific topics, an in-depth knowledge of DB2 UDB for OS/390 and z/OS is assumed.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at IBM EMEA ATS Products and Solutions Support Center, Montpellier, France.

Florence Dubois is an IT Specialist for DB2 on z/OS at the IBM EMEA ATS Products and Solutions Support Center, Montpellier, France. She has been with IBM since 1997, taking part in SAP performance benchmarks and providing on-site performance reviews to large SAP customers in EMEA. She teaches IBM classes on SAP implementation on zSeries™ and has written extensively on SAP and DB2 for z/OS.

Kellie Bohlsen is a DB2 for OS/390 Specialist with 13 years experience with the OS/390 platform. She worked for BHP Information Technology Australia as a CICS®/DB2 System Programmer before moving to the U.S. as a Technical DB2 DBA and SAP Basis Administrator. Most recently she worked for BMC Software in San Jose performing quality assurance on DB2 Performance products. She now lives in Australia and holds an Information Technology Degree from Newcastle University.

Tim Bohlsen is an SAP, DB2, and OS/390 Specialist. He has worked mainly in the U.S. with various SAP on DB2 for OS/390 customers during the last five years, as well as IBM Learning Services instructing. He has seven years experience with SAP and 16 years experience in the IT industry. Prior to working with SAP R/3, Tim was an MVS™ System Programmer in Australia for five years, specializing in large system performance and automation. He holds a degree in Computer Engineering from Newcastle University.

Bernd Kohler is a software developer with SAP AG, Walldorf in Germany. He joined the DB2/390 porting team in 1996 and has worked on a variety of development areas: data dictionary, upgrade, database layout, PTF check tool, and incremental migration. Bernd holds a Ph.D. in Theoretical Physics from the Technical University, Berlin.

Gert Ruland is a System Engineer at IBM in Germany. He has more than 20 years of experience in all areas of data processing. Currently, he is at the IBM SAP International Competence Center (ISICC) at Walldorf, Germany, doing technical support for SAP installations on zSeries. His areas of expertise include SAP on zSeries. He was coauthor of two other redbooks.

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

Viviane Anavi-Chaput

Richard Conway

Roy Costa

Mike Ebbers

Vasilis Karras

International Technical Support Organization, Poughkeepsie, NY, U.S.

Namik Hrle

IBM eServer™ Software Development, Boeblingen, Germany

Andreas Mueller

Holger Scheller

Johannes Schuetzner

Dietmar Stemmler

IBM mySAP Technology on DB2 for z/OS Development, Walldorf, Germany

Yves Tolod

Peter Wansch

IBM DB2 Tools & Connectivity Development, Toronto, Canada

James Teng

IBM DB2 for z/OS Development, Silicon Valley Lab, CA, U.S.

Wilhelm Burger

IBM SAP International Competence Center (ISICC), Walldorf, Germany

Noel Richard

IBM EMEA ATS Products and Solutions Support Center, Montpellier, France

Don Geissler

Michael Gordon

IBM U.S.

Helmut Roesner

IBM/SAP Integration Center, IBM Silicon Valley Labs, U.S.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

MCOD in a DB2 UDB for OS/390 and z/OS environment

This chapter covers the following topics:

- ▶ Introduction to Multiple Components in One Database (MCOD)
- ▶ Technical details of the MCOD implementation with DB2 Universal Database (UDB) for OS/390 and z/OS¹
- ▶ Discussion of setup options with DB2

For additional information, refer to:

- ▶ *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836
- ▶ SAP Service Marketplace quick link MCOD at:

<http://service.sap.com/mcod>

You must be registered as an SAP Service Marketplace user to access this resource. To register, go to:

<http://service.sap.com>

¹ In the rest of this book, we refer to DB2 UDB for OS/390 and z/OS using the short name of DB2.

1.1 Introduction

MCOD is a feature that enables the installation of several SAP components² independently in one single database management system (DBMS) instance. In this section, we discuss the reasons that motivated the implementation of this feature.

1.1.1 Motivation

Without MCODE, each SAP component has to be installed in a DBMS instance of its own. On the one hand, this is advantageous, because it gives maximum flexibility in terms of scalability and choice of the operating system (OS) and database server (DB). On the other hand, there are severe drawbacks that come with the administration of separate systems. The following tasks are particularly costly and difficult to handle:

- ▶ Maintaining a separate DBMS instance for each SAP component
- ▶ Setting up high-availability solutions for business applications that use multiple SAP systems
- ▶ Synchronizing the backup and restore for an extended SAP system landscape

1.1.2 Benefits

Using the MCODE functionality leads to a system landscape that is leaner and easier to maintain. The advantages are as follows:

- ▶ Reduced costs
 - Multiple different and independent software solutions are located in one logical and physical DBMS instance. The administrative effort is reduced considerably. Experience gained from several projects show that companies can achieve savings in the following areas:
 - Reduction of the costs related to disk space, with fewer DB2 subsystems and associated libraries and DB2 logs.
 - Reduction of the hardware backup costs, with fewer separate subsystems to consider.

² SAP components are defined as any of the mySAP components based on the SAP Web Application Server (SAP WAS, formerly SAP R/3 Basis) technology. Presently, this includes mySAP R/3, mySAP CRM, mySAP BW, and mySAP Workplace among others. Because they are based on SAP WAS technology, their underlying database server and database layout requirements are similar.

- Lower operating costs, such as managing backups or high-availability solutions. This is particularly the case where multiple, critical SAP components are located in an MCOD landscape. Here, the costs of high-availability solutions are shared by the SAP components, instead of being required separately, which would increase both costs and complexity.

These savings stem from reduced overall maintenance and operating costs, as well as a streamlined process for backup and high-availability solutions and an efficient use of your available resources.

- ▶ System spanning consistency

Point-in-time recovery of semantically related systems (for example, R/3 and CRM) is possible. There is no additional effort required for synchronization, as opposed to a single component installation.

- ▶ Simplified database administration, synchronized backup and recovery

This reduces the effort for database administration. Note, for example, that all systems use the same DBMS release.

- ▶ Greater use of high availability techniques

With multiple, critical components installed into one database, it is easier to implement high-availability solutions for this environment. This can include consideration of DB2 data sharing and use of a z/OS sysplex environment. In the case of DB2 data sharing, different members of a DB2 data sharing group can be configured with differing installation parameters (DSNZPARMs) appropriate for the SAP component having application servers utilizing those DB2 group members.

- ▶ Additive sizing approach

The total sizing requirements of an MCOD installation (CPU, storage, and so on) are easily determined by adding up the requirement for each single component.

- ▶ Complete independence of all SAP components

Some of these advantages are explored in more detail in the following sections.

1.1.3 Drawbacks

Remember, however, that there are also some drawbacks:

- ▶ If the DBMS instance is unavailable, all the SAP components resident in that DBMS instance are also unavailable. Therefore, MCOD may not be suitable for large installations or mission-critical applications if no failover strategy is in place. For more information, refer to *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836.

- ▶ The use of the MCOD feature can, in some circumstances, reduce flexibility. In a non-data sharing environment, the SAP components residing in one DB2 subsystem may only have their database server workload running in one z/OS logical partition (LPAR). This may be considered an advantage in some situations, and a disadvantage in others.
- ▶ Cloning single components is slightly more complex. This topic is covered in Chapter 5, “Cloning one component out of an MCOD landscape” on page 69.
- ▶ Backup, recovery, updating database statistics, and other utilities may take longer, although the total effort and duration will be quicker than the sum of the parts.
- ▶ Point-in-time recovery for a single SAP component within the MCOD landscape is possible, but much more complicated. This topic is covered in Chapter 6, “PPT recovery of one system in an MCOD landscape” on page 115.

1.1.4 Availability

The following can be said with respect to the availability of the MCOD feature:

- ▶ It is generally available on all OS/DBMS platforms supported by SAP.
- ▶ It is supported for selected products based on SAP Basis Release 4.6C/D and all SAP components based on SAP Basis Release 6.10 and later. For a detailed release matrix, refer to the SAP Service Marketplace quick link MCOD at:
<http://service.sap.com/mcod>
- ▶ All future SAP components are planned to be enabled for MCOD installation.
- ▶ An upgrade to a supported release may be necessary before consolidating an installed system into an MCOD landscape.

1.2 Technical realization

To better understand the following description of technical details, it may be helpful to review Appendix A, “Database layout” on page 145 in advance.

1.2.1 General implementation

Each SAP component connects to the database server with its own database user. The defaults are SAPR3 (4.6D and earlier) and SAP<SID> (6.10 and later). Note that this user is independent of the user logged on to the SAP system. In the case of DB2, the database user is identical to the CURRENT SQLID.

Database servers support schema. Tables with the same name but different schema can coexist in a database server. In the case of DB2, the schema is also called creator.

Using a one-to-one relationship between the database user and the database schema, each SAP component uses its own schema by accessing the database server with a unique dedicated database user. By that, all objects (in the case of DB2: stogroups, database, tablespaces, tables, indexes, and triggers) are clearly associated with only one SAP component. For table T100, this relationship is depicted in Figure 1-1.

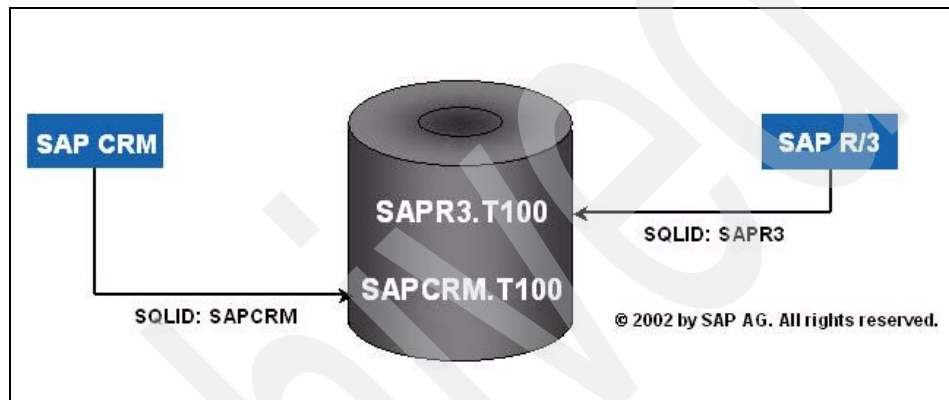


Figure 1-1 MCOD: Database user and schema

1.2.2 DB2-specific modifications

For the implementation of MCOD, SAP only requires a different schema for each component. In addition to that, some DB2-specific modifications are necessary (for example, to handle data storage). These enhancements are listed in the following sections.

Schema

Older SAP applications use the schema SAPR3 when accessing the database server. With the introduction of MCOD, the schema can now be specified during the installation procedure:

- ▶ For applications based on SAP Basis Release 4.6C/D, the default is still SAPR3. However, the default can be overwritten during the installation process (see 3.3.1, “SAP Basis Releases 4.6C and 4.6D” on page 33).
- ▶ For SAP Basis Releases 6.10 and later, the default value is SAP<SID>.

All DB2 objects belonging to one SAP component must have the same creator (the one specified during the installation). Besides tables, views, and indexes, this also includes stogroups, databases, and tablespaces.

Note that all SAP applications check whether the creator of a stogroup, tablespace, or database is correct before using it for the creation of a new database object.

Environment and profile setting

New variables on the application server are introduced to specify the schema:

- ▶ Most SAP applications and all stand-alone tools (for example, tp and R3trans) use the environment variable DBS_DB2_SCHEMA.
- ▶ In addition, some SAP applications retrieve the information from the instance profile, where it is specified using profile parameter dbs/db2/schema.

Stogroups

Each component has its own set of stogroups:

- ▶ If the creator associated with the SAP component is SAPR3, the traditional naming is applied for stogroups (for example, SAPBTD and SAPSTI).
- ▶ Otherwise, the first three characters are substituted with the SAP System ID. For example, if the SAP System ID is PRD, the stogroups will be named PRDBTD, PRDSTI, and so forth.

VCAT name

The VCAT name is specified during the installation procedure and subsequently used as an attribute when creating the stogroups. It should be chosen differently from the DB2 system data (catalog, directory, and so forth) and each other SAP component in the MCOD installation. Then, the VCAT can be used to easily identify SAP components on a data set level, because it forms the high-level qualifier (HLQ) of the data sets created by DB2.

Storage Management Subsystem (SMS) access control system (ACS) routines and storage groups can be adapted to allow each component to reside on its own pool of volumes to assist storage management. Alternatively, any combination of sharing or isolation is possible through SMS settings.

This is only possible by specifying unique VCATs when installing or merging into an MCOD environment and, therefore, is highly recommended by SAP and IBM (see 2.1.1, “Keep things as separate as possible” on page 16).

Databases and tablespaces

For databases and tablespaces, the naming convention is not changed to account for MCODE. Even for the merge in place procedure described in Chapter 4, “Merging SAP components without moving data” on page 41, the naming convention is preserved. Please note that there is no need to add a component-specific identifier to the database name, because of the following:

- ▶ On a data set level, the SAP components can be distinguished by the high-level qualifier (= VCAT name).
- ▶ In the DB2 catalog, the creator indicates the SAP component a database or tablespaces belongs to.

To account for the increased number of tablespaces in an MCODE system, there is a change in the 6.20 database layout. The maximum number of tablespaces per database has been raised from 1 to 100. For more details, refer to Appendix A., “Database layout” on page 145.

Plan name

Previous to MCODE, by default, SAP applications used the DB2 plan names FOME<REL>P for the Integrated Call Level Interface (ICLI) server and SAPR3<REL> for the z/OS application server (<REL> is the SAP Kernel Release). In an MCODE landscape, these names cannot be applied, because each application needs an individual plan name (ICLI connections are not shared between different SAP systems). A new default naming convention has to be introduced. For details, see Table 1-1.

Table 1-1 Naming convention for plan and package names

	ICLI	z/OS application server
Package	F<SID><REL>	O<SID><REL>
Plan	F<SID><REL>	O<SID><REL>

Here are two examples:

- ▶ FPRD46D is the plan name of the 4.6D ICLI server related to system PRD.
- ▶ For an z/OS application server of system DEV and Kernel Release 6.20, we use the plan name ODEV620.

Bufferpools

By default, all SAP components in an MCODE installation use the same bufferpools (BP2, BP3, BP8K0, BP16K0, BP32K). However, the bufferpool tuning can be adjusted individually for each SAP component by activating additional bufferpools (see 2.1.4, “Bufferpool tuning” on page 18).

Catalog

All SAP components share the DB2 catalog and directory as a common resource. Therefore, the only locking conflict between different SAP systems can occur in the catalog, and here, only the creation of databases is a likely candidate for deadlocks. Therefore, SAP recommends not to run upgrades, add-on installations, or database installations in parallel within an MCODE landscape.

1.2.3 Independence of components

As a consequence of the database layout and the technical implementation, the individual SAP components installed in one database are completely independent of each other. Therefore:

- ▶ It is possible to delete SAP components from an MCODE landscape. The SAP tools R3SETUP and SAPinst are able to exclusively delete all database objects of one SAP component from the subsystem.
- ▶ Upgrades can be performed for each system independently. However, during an upgrade, data definition language (DDL) operations take place that may incur wait times for those components that do DDL operations at the same time. In this case, there may be wait times or even dead locks on the DB2 catalog or dictionary, or both.
- ▶ The MCODE feature is a change in the SAP technology layer and, therefore, *neither* affects the business logic *nor* the business data of a component. The Implementation Guide (IMG), for example, will always use the correct table, that is, the one that belongs to the component.
- ▶ Data exchange always happens on an application level (see Figure 1-2 on page 9). Therefore:
 - The integrity of business data is ensured.
 - There is no special application coding for MCODE installations.
 - Locking conflicts on data between different components are not possible.

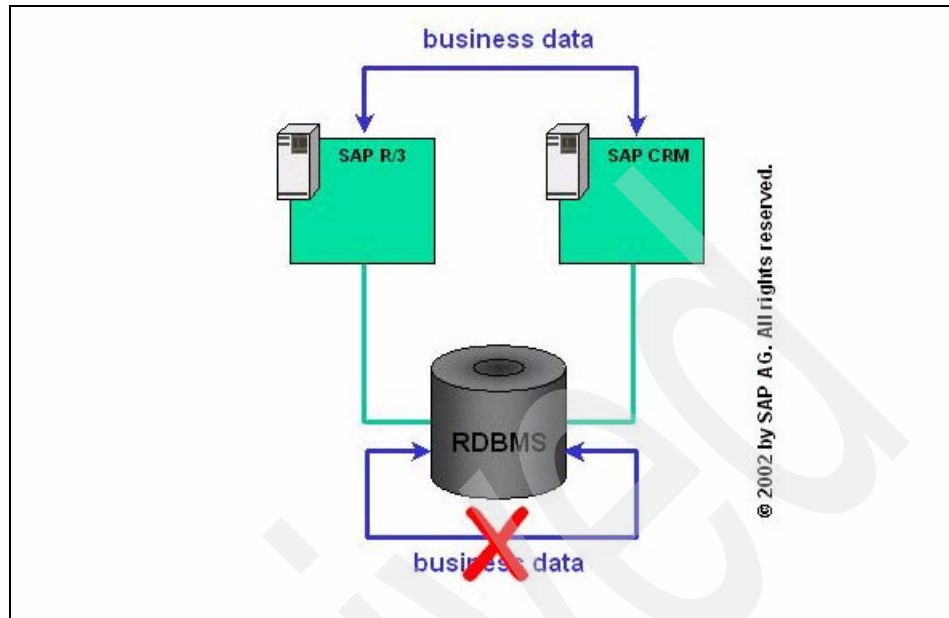


Figure 1-2 MCOD: No data exchange within the database

- ▶ Security

Because the SAP components are logically independent, they cannot influence each other. This applies to DDL and data manipulation language (DML) operations.
- ▶ <sid>adm users

There is no change in the number of <sid>adm users needed for administration (transports and so forth). The different DB schemas are reflected in the environments and profiles of the <sid>adm users.

1.2.4 Implementation

Using the MCOD technology is as straightforward as installing a new SAP component in a separate subsystem. Therefore, no extra effort is required. Keep in mind that SAP R/3 systems need to be upgraded to mySAP.com (including the latest SAP R/3 4.6C component) or SAP R/3 Enterprise to be enabled for MCOD.

For standard SAP (front-end) users, there is no difference compared to a non-MCOD installation. Even SAP Basis administrators who use stand-alone tools, such as tp, or perform upgrades are unlikely to notice any difference. Basically, MCOD is implemented entirely on a database (administration) level.

1.3 Setup options

There are several ways to establish an MCOD landscape with DB2. The main decision is whether to use DB2 data sharing or not. For this, the type of system and performance issues are relevant.

1.3.1 Basic setup considerations

This section describes basic setup considerations.

OLTP and OLAP systems

In general, it is not recommended to combine online transaction processing (OLTP) systems, for example, SAP R/3, SAP Customer Relationship Management (CRM), and SAP Workplace (WP), and online analytic processing (OLAP) systems, for example, SAP Business Information Warehouse (BW) and SAP Strategic Enterprise Management (SEM), in one database system. The reason is that these two system types require a different setting and tuning of the database system in order to maintain a high level of performance and good response times.

The situation is different in a DB2 data sharing system where individual tuning is possible (see 1.3.3, “Data sharing DB2” on page 12). Nevertheless, it has to be assessed carefully in each case whether the combination of OLTP and OLAP in one DB2 subsystem makes sense.

Performance

Certainly, MCOD installations need more resources for the database work than a single system, but fewer resources than the sum of all the systems within the database system. This is because the load can be better balanced when using this architecture by utilizing the facilities and features available.

Usually, SAP recommends to only combine low- to mid-sized systems in an productive MCOD installation. Otherwise, you may soon reach resource limitations. Again, DB2 data sharing is a powerful tool to tackle the larger workload and, therefore, allow the combination of large systems (see 1.3.3, “Data sharing DB2” on page 12).

System recommendations

Not every MCOD configuration makes sense. SAP recommends to only combine the following:

- ▶ Systems with semantically-related data, such as SAP R/3 and CRM
- ▶ Systems of the same type, for example, combinations of development or combinations of production systems

The combination of production and non-production systems is not recommended.

1.3.2 Non-data sharing DB2

Figure 1-3 on page 12 depicts an MCOB installation (and various setup options):

- ▶ The installation contains three different SAP components: SAP CRM, SAP R/3, and “New” application. The third one is called “New” application and denotes a future component that may employ Unicode and DRDA®. Note that ASCII and Unicode data can coexist in the same DB2 subsystem.
- ▶ The work processes connect to the database server by different means: z/OS application servers connect locally, and ICL1 is the software product currently used by all non-z/OS application servers (on z/Linux, hipersockets are used). In the future, SAP plans to implement Distributed Relational Database Architecture (DRDA) to support Unicode applications.
- ▶ It is possible that several SAP components use the same bufferpools. In Figure 1-3 on page 12, this is the case for SAP CRM and SAP R/3. However, you can also activate and use additional bufferpools for one or more applications, for example, to improve the performance for the “New” application. For details, see 2.1.4, “Bufferpool tuning” on page 18.

The MCOB landscape can be tailored to the specific needs of a customer.

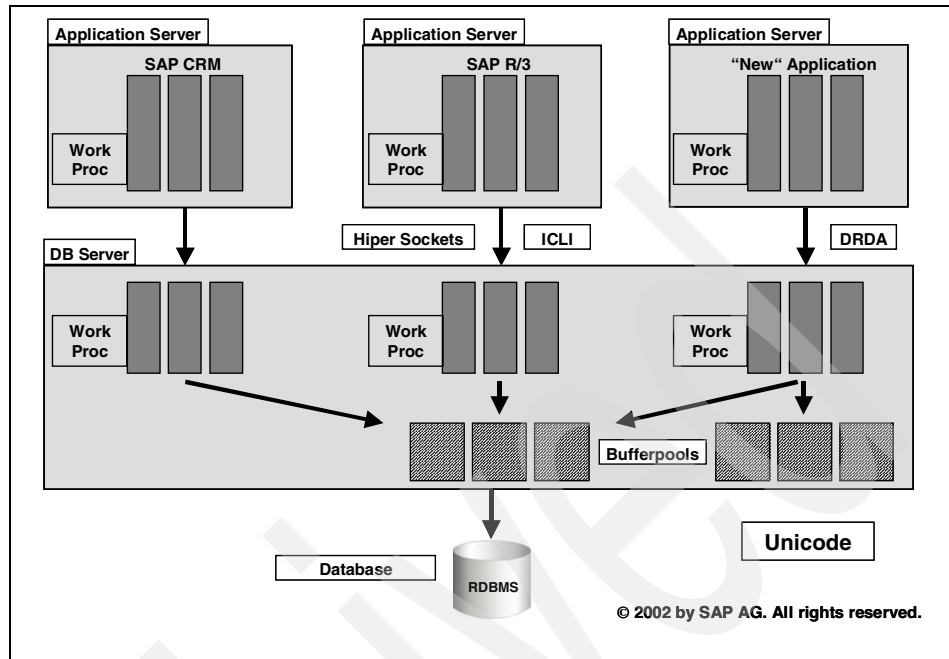


Figure 1-3 MCOD landscape with DB2

1.3.3 Data sharing DB2

Currently, DB2 is the only database platform that can be used in parallel mode for all SAP scenarios. It also very well suited for MCOD installation. Figure 1-4 on page 13 shows a possible configuration with three different SAP components. Each component is associated with a data sharing member of its own. Each data sharing member should be connected to the set of application servers dedicated to that particular SAP component.

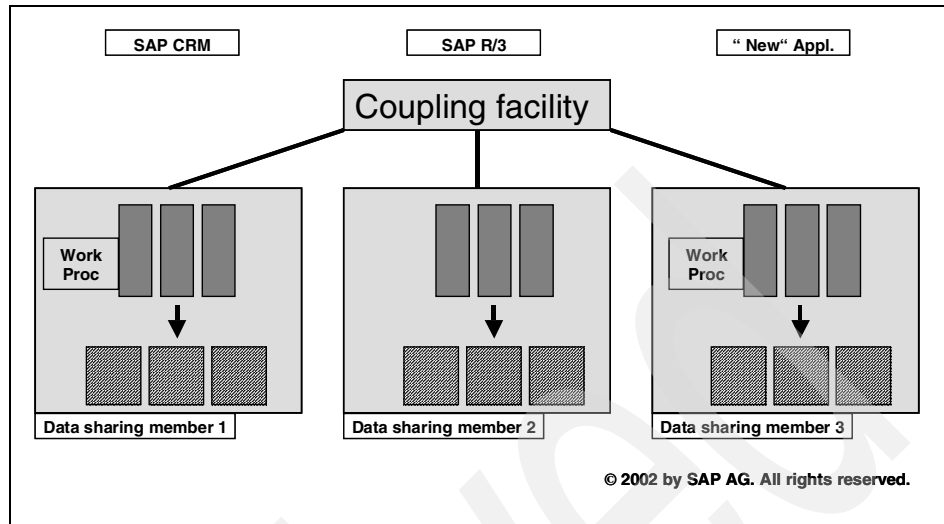


Figure 1-4 MCOD and data sharing

There are several advantages with this kind of setup:

- ▶ A very high degree of scalability can be achieved. Resource constraints can be overcome by adding additional CPUs, memory, or even LPARs. This caters to MCOD installations for large, productive SAP system landscapes.
- ▶ The SAP components are stored independently from each other. Therefore, there is no contention for data between the data sharing members.
- ▶ With DB2 data sharing, it is possible to combine OLTP and OLAP systems, because the data sharing members can be set up and tuned individually.

Archived

Planning for an MCODE landscape

This chapter discusses the following topics:

- ▶ Considerations when planning for an MCODE installation
- ▶ The initial configuration we used for our tests

For additional information, refer to:

- ▶ *SAP R/3 on DB2 for OS/390: Database Availability Considerations*, SG24-5690
- ▶ *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836
- ▶ *SAP R/3 on DB2 UDB for OS/390 and z/OS: Planning Guide, 2nd Edition, SAP R/3 Release 4.6D*, SC33-7966
- ▶ *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20*, SC33-7959
- ▶ *DB2 Universal Database for OS/390 Data Sharing: Planning and Administration Version 6*, SC26-9007

2.1 Planning for MCODE

In this section, we discuss what has to be considered when we want to run multiple SAP components in one DB2 subsystem.

2.1.1 Keep things as separate as possible

To insure maximum flexibility and ease of administration, we highly recommend to have the following:

- ▶ A separate alias (or VCAT) and integrated catalog facility (ICF) for the DB2 catalog and directory data sets

This allows you to easily differentiate between VSAM data sets belonging to the DB2 subsystem itself and those that comprise SAP application data.

- ▶ A separate alias (or VCAT) and ICF catalog for each SAP component

This enables the flexibility to move between systems the VSAM data sets corresponding to one SAP component by exporting the ICF catalog. For system cloning and other activities, it allows efficient mass processing of the objects for an SAP component by simply reconnecting the ICF catalog and reinitializing volumes.

- ▶ Separate SMS storage group(s) for each SAP component
- ▶ A separate pool of volumes for each SAP component

2.1.2 Size of the DB2 subsystem

If we start from scratch, we can get a sizing of the MCODE landscape as follows:

1. Size every SAP system that will be part of the MCODE landscape using the SAP quicksizer, available on SAP Service Marketplace quick link QUICKSIZER at:
<http://service.sap.com/quicksizer>
2. Ask IBM to map the results of the quicksizer to the required resources on an S/390® or a zSeries box.
3. Use the sum of all components as the sizing for processors, storage, DASD, and channels.

This also applies if we want to add only one new SAP system to an MCODE landscape.

However, with this *additive* approach, we may count common resources several times. We believe that SAP AG will work on additional sizing options once it has gained more experience with controlled installations.

This applies to the following:

- ▶ Storage
- ▶ CPU processors
- ▶ I/O bandwidth

We may even consider implementing a DB2 data sharing environment to relieve constraints in these areas.

By using the MCOD landscape to place critical SAP components into one DB2 subsystem, we may subsequently adopt some of the advantages of DB2 on S/390 or zSeries platforms to increase availability. These actions include implementing DB2 data sharing and the high availability option for the SAP Central Instance. These topics are covered in *SAP R/3 on DB2 for OS/390: Database Availability Considerations*, SG24-5690 and *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836.

2.1.3 Number of SAP systems in the MCOD landscape

The number of SAP components in a DB2 subsystem is limited by the maximum number of databases, which today is 65,271. However, you should plan to never approach this limit in an MCOD installation, because then, there is no space left for additional databases created during an upgrade or an add-on installation. Therefore, for the initial setup of an MCOD installation, the maximum number of database should not be more than 50,000.

Note that the number of database needed for an SAP component depends on the SAP Basis Release:

- ▶ SAP Basis Release 4.6D and earlier

The number of tablespaces (and hence databases) in an SAP R/3 4.6C system is approximately 10,000. The number is smaller for other applications based on SAP Basis Releases 4.6C and 4.6D.

Therefore, a pure 4.6x MCOD installation can contain up to five SAP systems.

- ▶ SAP Basis Release 6.10 and later

For SAP Basis Release 6.10 and later, the one-to-one relationship between tablespaces and databases has been removed. One database can now hold up to 100 tablespaces. Therefore, the number of databases in a newly installed SAP system based on SAP Basis Release 6.10 or later is approximately 1,000.

The database layout would allow an MCOD installation with 50 SAP systems. However, for practical reasons, we recommend limiting the number to 20 components.

2.1.4 Bufferpool tuning

Planning for an MCOB landscape requires planning strategies for bufferpool tuning. Methodologies for mapping DB2 objects into appropriately configured bufferpools to suit the normal activity on the objects is described in detail in *SAP R/3 on DB2 UDB for OS/390 and z/OS: Planning Guide, 2nd Edition, SAP R/3 Release 4.6D, SC33-7966*, and *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20, SC33-7959*. In an MCOB landscape, we also have additional considerations about the interaction of the SAP components installed in the DB2 subsystem.

There are two extreme approaches, one to replicate all SAP related bufferpools for every component, the other to retain one set of “categorized” bufferpools and place objects from all SAP components into these. There are then various approaches between these two extremes, with some objects in shared bufferpools, and some in bufferpools dedicated to objects belonging to one SAP component.

First, we recommend aiming placing DB2 objects into appropriately sized bufferpools shared across SAP components. Then, place any objects requiring special treatment in one SAP component into dedicated bufferpools. The sharing of bufferpool resources between components will allow the most efficient use of the storage resources. You may, however, want to take more control over the relative resource usage between the SAP components, and this may be achieved by appropriate bufferpool and Workload Manager (WLM) configurations.

It should also be noted that the additional storage resources required in a large MCOB installation may necessitate the usage of data space bufferpools, as well as normal virtual pools, to provide relief from virtual storage constraints along with the normal techniques used in tuning large SAP on DB2 for z/OS installations.

2.1.5 Recovery of the DB2 subsystem

Recovery falls into two different cases, driven by the type of MCOB landscape you have in mind.

Consistency MCOB

If you use MCOB for consistency reasons, you have to consider the complete DB2 subsystem as a unit of recovery. The normal recovery procedures apply. You have to be aware that you deal with n-times more tablespaces, as you run n SAP systems in one database. If you have to meet targets for the duration of the recovery, you have to adapt your recovery procedures to cope with that restriction.

Consolidation MCOD

If you use MCOD for consolidation, you want to have a possibility to recover every single SAP system to a certain point in time. This can be achieved. The process is explained in Chapter 6, “PPT recovery of one system in an MCOD landscape” on page 115. However, this requires the following:

1. A cloning of the DB2 system.
2. A recovery of the clone to the requested time. The recovery may only be done for the SAP system under discussion.
3. A deletion of the SAP system under discussion in the target system.
4. A merge of the SAP system out of the recovered clone to the target system.

You can already see that it requires all the techniques described in this book.

2.1.6 Resources for daily tasks

If you merge n SAP systems into one DB2 subsystem, you just increase the objects by a factor of n . The objects themselves may vary in size. The batch window may be not sufficient to run all the backup jobs. You have to review your backup procedures and may switch to less time consuming procedures.

2.1.7 License key

A valid license key is required for each component in the MCOD landscape.

2.1.8 Checklist

We summarize our considerations in Table 2-1.

Table 2-1 *Planning checklist for MCOD considerations*

Item	Considerations	
Type of MCOD landscape	Consolidation or consistency.	
	Number of SAP components.	
Naming conventions	Adopt rigorous conventions.	
Alias	One per SAP system and one for DB2 system.	
ICF catalog	One per used alias.	
SMS storage group	One or more per used alias.	
Volumes	Separate pool of volumes for each SAP component.	

Item	Considerations	
Sizing of the system	CPU.	
	Main storage.	
	I/O bandwidth.	
Bufferpool tuning	Redo bufferpool tuning.	
Recovery	Review recovery procedures and adjust.	
	Plan for recovery of a single component (consolidation MCOD).	
Daily tasks	Review daily jobs and adjust.	
Installing a new system into an existing one	“Install directly” on page 20.	
	“Installing a new component” on page 30.	
Merging two systems	“Merging two systems” on page 20.	
	Merging SAP components without moving data.	
License keys	Get them from SAP.	

2.2 Install directly

The simplest way to get a running MCOD landscape is to use an MCOD-enabled SAP system as a starting point and install another SAP system into this very DB2 subsystem. This approach is outlined in Chapter 3, “MCOD installation and merge using SAP tools” on page 27.

2.3 Merging two systems

Another option is to merge existing SAP systems.

First, you have to consider which of the two systems will be the receiving system. We call it the *target* system in the discussion. It is a system that may still use the schema SAPR3. Maintenance has to be applied to this system, as outlined in Chapter 3, “MCOD installation and merge using SAP tools” on page 27. We call the other system the *source* system.

Then, you have to decide which method you will use to merge the two systems. This decision will be based on affordable down time during the merge process and available DASD space.

2.3.1 Three methods of merging

There are three possible methods of merging.

Using tools provided by SAP

This is the preferred method, especially for small- and medium-sized systems when you have no constraints regarding down time or DASD space. The migration tools come in two flavors:

- ▶ The classical one explained in Chapter 3, “MCOD installation and merge using SAP tools” on page 27
- ▶ The incremental migration method, which is more complex, but considerably reduces down time

Using DB2 tools to move the data

We have to extract all the information out of the source DB2 system and create all the objects on the target system with a new schema name. In the context of SAP systems, the preferred schema name is SAP<SID>. However, this schema name relies heavily on the uniqueness of the SAP System ID in the landscape. This procedure might be faster than using SAP-provided tools. The process itself has two phases:

1. Extract the information out of the source DB2 catalog. This can be done with handwritten programs or a tool. If you decide to do it with handwritten programs, you need to have a deep understanding of where to find the information in the DB2 catalog tables to generate all the DDL needed. This is by no means a trivial task. We call this the online phase, because we do not need to have exclusive access to the systems.
2. Move the data with any tool which fits to the task, for example DB2 UNLOAD/LOAD utilities. This is the offline phase, because we do need to have exclusive access to the data.

Note: DB2 Administration Tool for z/OS is capable of extracting the object information out of the source DB2 catalog.

Unique database names requirement

We have to deal with another difficulty. The database names have to be unique in a DB2 system. We have to insure that this prerequisite is fulfilled. This requires a mapping table that gives us, for every database on the source system, a new database name on the target system. This itself is not trivial if we consider the number of databases involved.

Using DB2 tools without moving the data

This process is described in 4.7, “Merging existing DB2 data into the group,” in *DB2 Universal Database for OS/390 Data Sharing: Planning and Administration Version 6*, SC26-9007. It requires a careful understanding of what to do and how to do it. We explain this further in Chapter 4, “Merging SAP components without moving data” on page 41. This method is more complex than merging using DB2 tools to move the data. It is also a two-phase approach:

1. In phase one, we extract the object information on the source system and apply it to the target system. We have to preserve the object identifiers of the tables (OBID). This requires careful planning. This is an online phase, because we do not need to have exclusive access to the database. However, it must be insured that there is no change in the object definition.
2. The second phase deals with adjusting the data objects with the DB2 REPAIR utility and renaming the data sets using the VSAM utility IDCAMS ALTER. The number of objects that need this treatment is so large that we have to work with a semi-automated procedure. We describe this in 4.5, “Migration steps” on page 60.

Note: We used DB2 Administration Tool for z/OS for the object definition part and some REXX procedures to create the VSAM IDCAMS control statements and the REPAIR statements.

Highest log RBA requirement

This method has to adhere to another restriction imposed by the LOGAPPLY mechanism of DB2 during restart or recovery: The current relative byte address (RBA) of the target system has to be higher than that of the source system. This has to be checked by printing the BSDS of the source and target system with the DSNJU004 stand-alone utility. For more information, refer to *DB2 Universal Database for OS/390 and z/OS: Utility Guide and Reference Version 7*, SC26-9945.

This requirement might be met by providing an arbitrarily chosen RBA during a cold start of the target system or by doing tasks that generate log entries (a client copy, for example). A cold start of the DB2 subsystem will begin a new recovery cycle. This implies a down time of the target system.

Important: Having the higher RBA on the target system is crucial for the process explained in this book. We understand that there are means to insure this, but they may have drawbacks.

Unique database names requirement

We have to deal with the uniqueness of the database names in a DB2 system. We have to insure that this prerequisite is fulfilled. This requires a mapping table, that gives us for every database on the source system a new database name on the target system. This itself is not trivial if we consider the number of databases involved.

Note: We defined metadata tables to hold the mapping information and used REXX procedures to update them. See 4.4.5, “Source and Target: Create and populate metadata tables” on page 53.

2.3.2 Additional tasks

SAP application servers must be adjusted so that they do point to the new database server. This is accomplished, for example, by reinstalling the application servers again, as described in “Set up the target application server” on page 35.

Once merge is complete, a new license key is required for the target system. The license key of the source system is not valid for this system.

You must perform RUNSTATS on the target system for the newly added tablespaces.

2.3.3 Checklist for merging

We provide a short checklist in Table 2-2.

Table 2-2 Checklist for merging

Tasks	Reference	
Plan for the installation	“Planning for an MCO landscape” on page 15.	
Determine source and target	Consider “Highest log RBA requirement” on page 22.	
Direct installation	“MCO installation and merge using SAP tools” on page 27.	
Merge using SAP tools	“MCO installation and merge using SAP tools” on page 27.	
Merge using DB2 tools and moving the data	Not covered in this book.	

Tasks	Reference	
Merge using DB2 tools without moving the data	“Merging SAP components without moving data” on page 41.	
Additional tasks	“Set up the target application server” on page 35.	
	Insure that you get the new license keys.	
	Populate the statistics for the new objects in the target system.	

2.4 Our system configuration

We designed a configuration to demonstrate how MCOD can be implemented and managed. Our initial configuration included three SAP systems: an SAP Web Application Server based on SAP Basis Release 6.20 (RED), an SAP Basis Release 4.6C (BLU), and an SAP Basis Release 4.6D (WHT).

The database servers were installed on two LPARs running z/OS in a sysplex with three DB2 subsystems (non-data sharing). For the application servers, we used one LPAR with z/VM™ having two Linux guests. This is shown in Figure 2-1 on page 25.

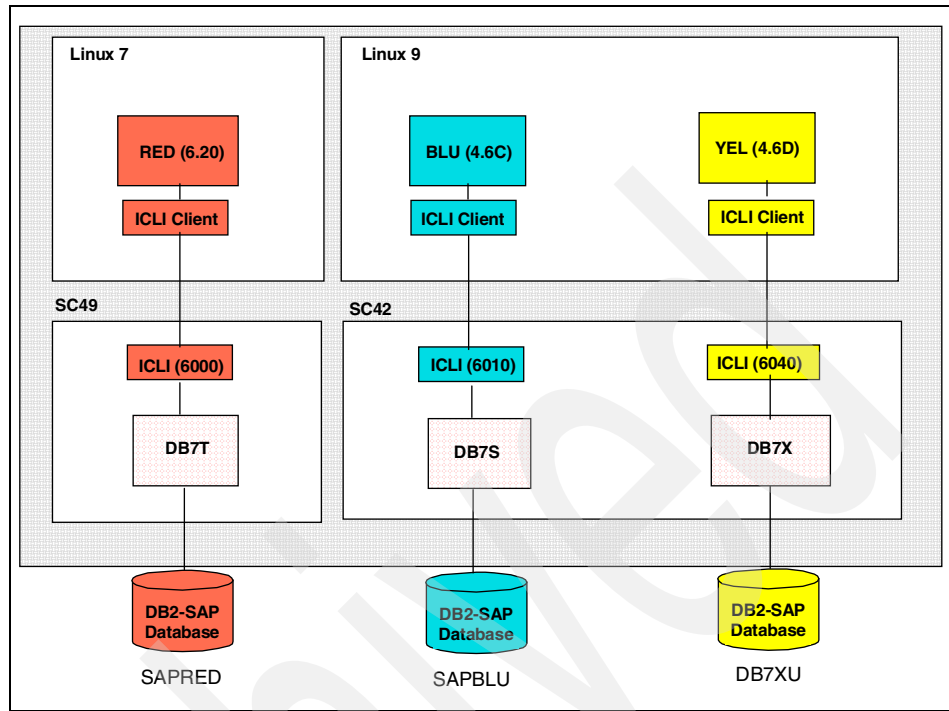


Figure 2-1 Our initial configuration

In the following chapters, we describe how we set up the two MCOD landscapes:

- ▶ One using SAP tools
- ▶ The other by merging two systems without moving the data sets

Archived

MCOD installation and merge using SAP tools

This chapter covers the following topics:

- ▶ We describe how to use SAP installation tools to set up an MCODE landscape. Please note that we distinguish between 4.6x- and 6.x-based products.
- ▶ We discuss how to minimize the merging down time.

For additional information, refer to:

- ▶ The following SAP Notes

You must be registered as an SAP Service Marketplace user to access the following resources. The registration requires an SAP installation or customer number. To register, go to:

<http://service.sap.com>

- *DB2/390: Incremental Migration to DB2/390*, SAP Note 353558
- *DB2/390: MCODE installation*, SAP Note 399419
- *DB2/390: APAR List*, SAP Note 081737
- *DB2/390: PTF check tool*, SAP Note 183311
- *DB2/390: DDIC corrections (Releases 4.6A, 4.6B, 4.6C, 4.6D)*, SAP Note 184399

- *DB2/390: DDIC corrections (6.10, 6.20), SAP Note 407663*
- *DB2/390: Newest version of the CCMS 4.6C, SAP Note 217468*
- *DB2/390: Latest version of CCMS 4.6D, SAP Note 337776*
- *DB2/390: CCMS corrections (6.10, 6.20), SAP Note 427748*
- *DB2/390: MCOD installation tools, SAP Note 580665*
- ▶ The SAP installation and migration guides available on the SAP Service Marketplace quick link INSTGUIDES at:
<http://service.sap.com/instguides>
- ▶ The following planning guides:
 - *SAP R/3 on DB2 UDB for OS/390 and z/OS: Planning Guide, 2nd Edition, SAP R/3 Release 4.6D, SC33-7966*
 - *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20, SC33-7959*

3.1 Introduction

There are several ways to set up an MCOD landscape:

- ▶ Install a new instance in a DB2 subsystem where another instance is already defined.
- ▶ Merge two existing components in the same DB2 subsystem.

In this chapter, we describe how both tasks can be done using the standard SAP installation tools. However, we need to distinguish between 4.6x and 6.x based products, because the installation tool has changed in between.

All references to an SAP Basis Release are also valid for all other SAP products based on this SAP Basis Release. For example, SAP Enterprise 4.7 is based on 6.20 technology and encapsulates all features of SAP Basis Release 6.20. Table 3-1 gives an overview.

Table 3-1 Relation between SAP Basis Releases and SAP products based on it

SAP Basis Release	SAP products
4.6C	Workplace 2.x, R/3 4.6C, BW 2.0B, BBP/CRM 2.0B
4.6D	Workplace 2.11, BW 2.1C, BBP/CRM 2.0C, APO 3.10
6.10	Web AS 6.10, BW 3.0A, EBP/CRM 3.0
6.20	Web AS 6.20, BW 3.0B, BW 3.10, EBP 3.5, CRM 3.1, Enterprise 4.7

3.1.1 SAP Basis Releases 4.6C and 4.6D

The installation and merge procedures outlined in the following sections are only possible for the 4.6x-based SAP products listed on the SAP Service Marketplace quick link MCOD at:

<http://service.sap.com/mcod>

Because MCOD was introduced when most of the SAP products based on Basis Release 4.6x (for example, SAP R/3 4.6C) were already shipped, most of the functionality needed for MCOD is not available immediately:

- ▶ In the context of this book, it is particularly important to understand that the installation tool R3SETUP and the templates shipped with the 4.6C/D installation CD-ROMs *cannot* be employed in general to install or migrate into an existing MCOD landscape. Usually, both R3SETUP and its templates have to be upgraded before starting the installation procedure. For details, see *DB2/390: MCOD installation tools*, SAP Note 580665.

- ▶ Additional actions are also necessary to adjust *all* SAP systems in the MCOD installation. They are described in 3.4, “Steps after installation or merge” on page 37.

3.1.2 SAP Basis Releases 6.10 and later

As of SAP Basis Release 6.10, MCOD is supported by all SAP tools and products. For example, if you install a Web Application Server 6.20, the installation tool SAPInst:

- ▶ Lets you specify a creator/schema other than SAPR3.
- ▶ Checks that there are no objects owned by the creator in the subsystem.
- ▶ Determines unique database names taking into account existing SAP components.

All other tools and transactions have also been adjusted for MCOD, for example, the upgrade, CCMS, or data dictionary (DDIC). Nevertheless, you should always perform the steps described in 3.4, “Steps after installation or merge” on page 37 after finishing the installation or merge.

3.2 Installing a new component

In this section, we describe how to install a new SAP component into an MCOD landscape.

3.2.1 SAP Basis Releases 4.6C and 4.6D

We did not perform this action. However, the procedure is very similar to the one outlined in 3.3, “Merging components into an MCOD landscape” on page 32. For details, refer to *DB2/390: MCOD installation tools*, SAP Note 580665 and *DB2/390: MCOD installation*, SAP Note 399419. Also, make sure that the steps described in 3.4, “Steps after installation or merge” on page 37 are performed after the installation.

3.2.2 SAP Basis Releases 6.10 and later

To show how simple it is to enter the MCOD world, we added one SAP component (WHT) based on SAP Basis Release 6.20 into DB2 subsystem DB7X, where another component (YEL) based on SAP Basis Release 4.6D was already installed (see Figure 3-1 on page 31).

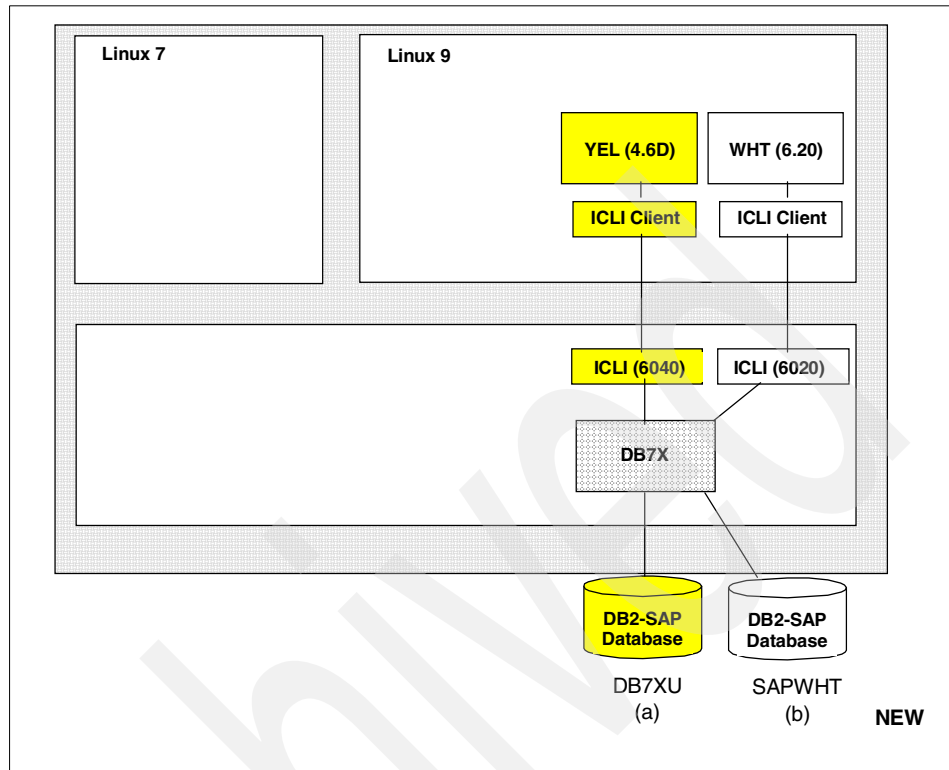


Figure 3-1 Installing a new dialog instance in an MCOD landscape

We use the installation tool SAPinst and follow the procedure described in the *Installation Guide – SAP Web Application Server 6.20 on UNIX: IBM DB2 Universal Database for OS/390 and z/OS*, available on the SAP Service Marketplace quick link INSTGUIDES at:

<http://service.sap.com/instguides>

The most important input screen with respect to MCOD is depicted in Figure 3-2 on page 32, where DB2 installation parameters and, in particular, the schema (in our case, SAPWHT) have to be specified.

Do not forget to perform the post-installation steps described in 3.4, “Steps after installation or merge” on page 37.

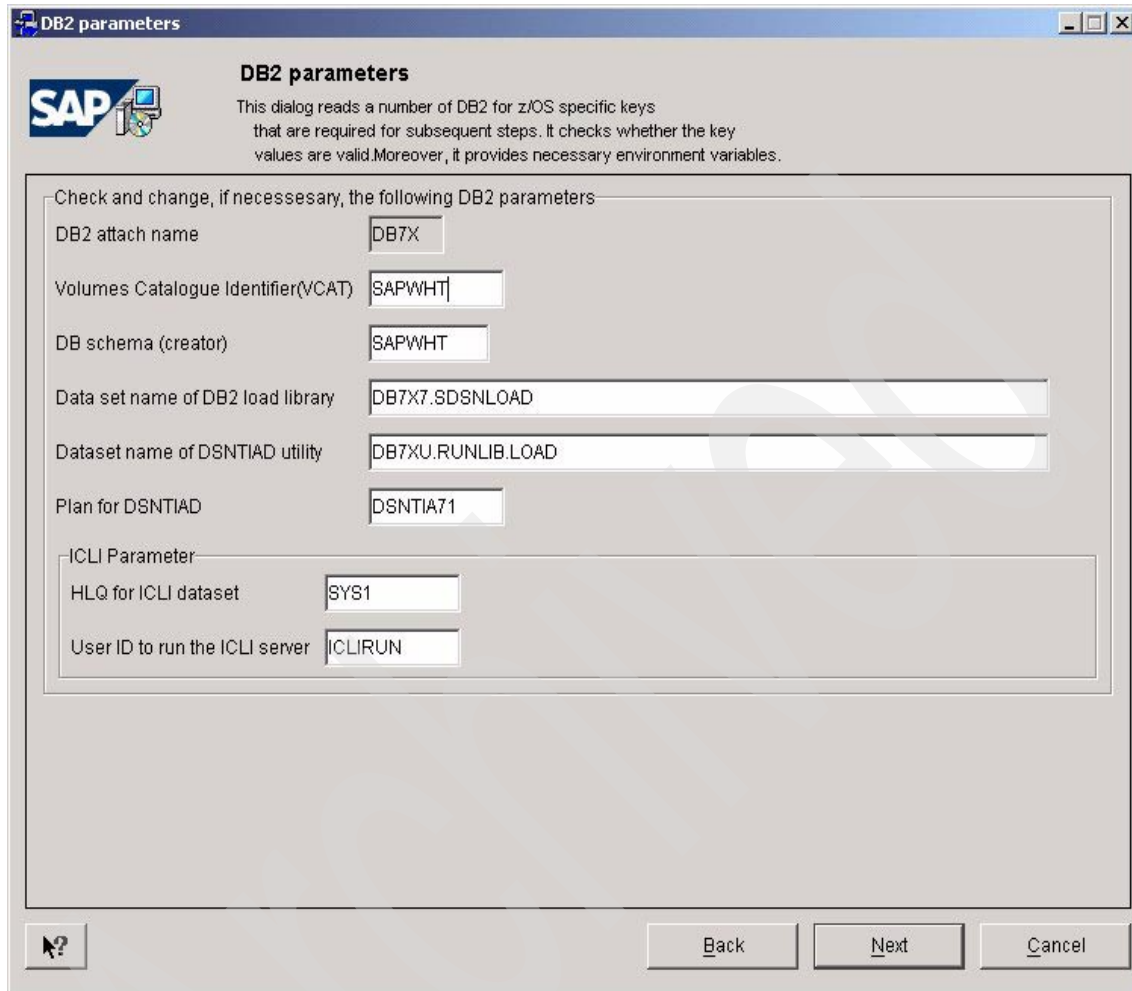


Figure 3-2 SAPinst: Specifying DB2-specific installation parameters

3.3 Merging components into an MCODE landscape

In this section, we describe how to merge components into an MCODE landscape using SAP tools.

3.3.1 SAP Basis Releases 4.6C and 4.6D

We describe here how to merge an SAP component based on SAP Basis Release 4.6x into an MCO landscape. In particular, we show how to upgrade both R3SETUP and its templates.

Description of the scenario

In this scenario, we have the MCO landscape created in the previous phase with two components: YEL and WHT. Now, we want to merge into that MCO landscape the SAP component BLU based on Basis Release 4.6C. See Figure 3-4 on page 34.

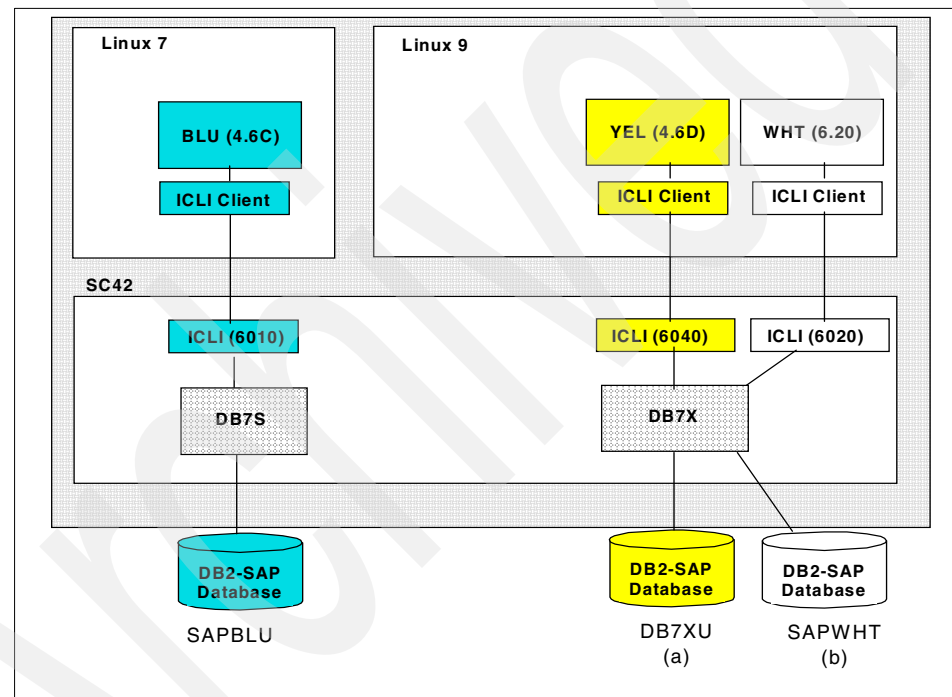


Figure 3-3 Merging components with SAP tools: Initial configuration

The merge consists of the following steps:

1. Export the source system (in our case, BLU).
2. Set up the target application server (in our case, GRE).
3. Load the exported data into the target database.
4. Post-installation steps.

Figure 3-4 shows the final configuration after the merge. We now have one MCOD landscape consisting of three SAP components: YEL, WHT, and GRE. Note that the three components are based on different SAP Basis Releases.

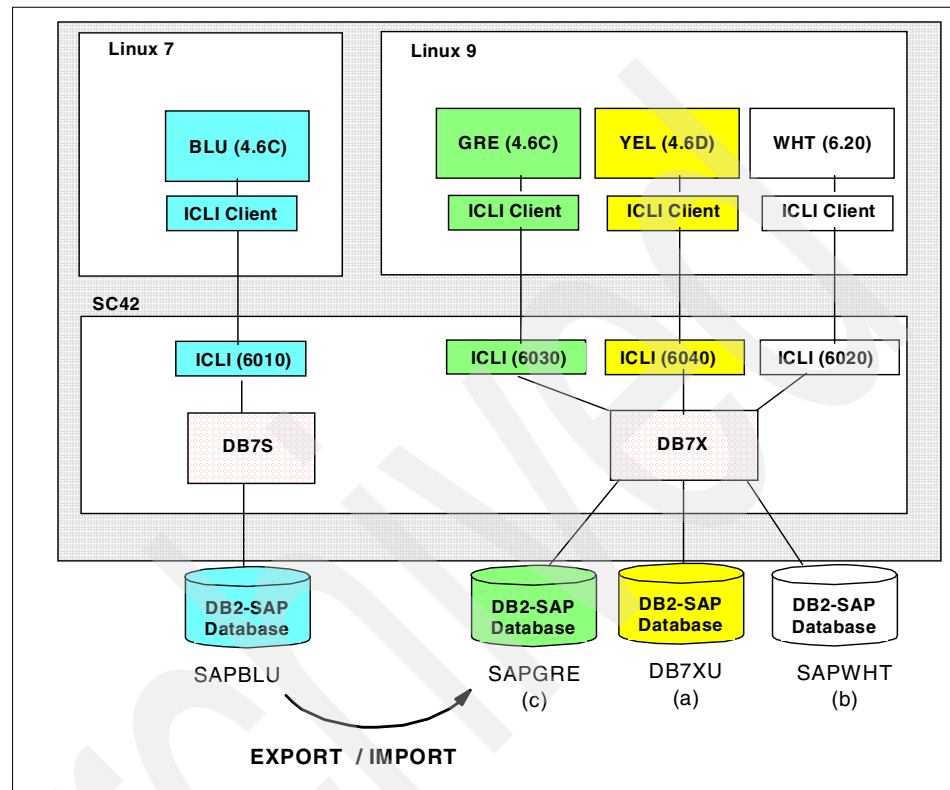


Figure 3-4 Merging components with SAP tools: Final configuration

Export the source system

The export has to be performed using standard 4.6D migration tools. For details, refer to the guide *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678, available on the SAP Service Marketplace quick link INSTGUIDES at:

<http://service.sap.com/instguides>

The steps are as follows:

1. Set up the export and work directory.
2. Go to the work directory and call:

```
/<CDMOUNT>/UNIX/INSTTOOL.SH
```

3. Only Linux for zSeries:
 - a. Retrieve the file DBEXPORT.R3S from sapservX.
 - b. Obtain latest R3SETUP patch from SAP Service Marketplace quick link PATCHES at:
<http://service.sap.com/patches>
4. Shut down the source SAP system:
`stopsap r3`
5. Start R3SETUP:
`./R3SETUP -f DBEXPORT.R3S`
6. Specify all parameters as usual.
7. Continue and finish the export as usual.

Set up the target application server

The following list describes how to set up the target application server.

Important: It is highly recommended when migrating with SAP tools that you reinstall the SAP instance or instances using R3SETUP, even if you are reusing the same application server hardware with SAP instances already installed. This ensures that all required MCOD settings are correctly implemented in the environment and instance profile variables, as well as plans bound and grants executed.

To preserve any changes you have made in SAP profiles, take a copy of the files from `/usr/sap/<SID>/system/profile`, as well as the `.dbenv_XXXX` files from the `$HOME` directory of `<sid>adm` user, prior to starting the install. Upon completion of the migration, you will need to integrate the parameters in the new instance and those in the files you have saved. A complete backup of the application server is highly recommended prior to reinstalling to allow a point of return in case of unforeseen error.

The target application server is installed as follows:

1. Set up the installation directory.
2. Go to the installation directory and execute:
`/<CDMOUNT>/UNIX/INSTTOOL.SH`
3. Copy MCOD-specific R3SETUP templates from sapservX to the installation directory. For details, see *DB2/390: MCOD installation tools*, SAP Note 580665.

4. Retrieve the latest 4.6D R3SETUP patch from SAP Service Marketplace quick link PATCHES, copy it to the installation directory, and execute:

```
./R3SETUP -v
```

Check that the output is VERSION: 20020504 or later.

5. Modify MCODECE_UNIX_db2.BASE.R3S:
 - a. Specify/add CREATOR=<SCHEMA> in section [DBCCOMMONPARAMETERS_IND_xxx].
 - b. Specify as 1_LABEL (in most cases located in section [CDSERVERKERNELBASE_IND_xxx]) the content of LABEL.ASC of the Kernel CD, for example:

```
DB2forOS/390:BASIS:46D:KERNEL:Kernel:CD51017309
```

6. Start R3SETUP:

```
./R3SETUP -f MCODECE_UNIX_db2.BASE.R3S
```

7. Specify the input as usual.
8. Check BIND and GRANT JCLs. The plan name should be F<SID>46D.
9. Continue and finish the installation.
10. Modify the plan name (from step 8) and, optionally, the port number in the ICL procedure or shell script. Start the ICL server on the host.
11. Apply the latest 4.6D Kernel patches from the SAP Service Marketplace quick link PATCHES, in particular for the DBSL library.
12. Log on as <sid>adm and check whether the connection works using:

```
R3trans -x
```
13. Check that DBS_DB2_SCHEMA is set to <SCHEMA>, using the **echo** command.
14. Specify dbs/db2/schema=<SCHEMA> in the profile DEFAULT.PFL.

Load data into the target database

To load the exported SAP component into the target database, proceed as follows:

1. Go to the installation directory.
2. Modify MCODEMIG_UNIX_db2.BASE.R3S:

Specify CREATOR=<SCHEMA> in one of the following sections:

 - [DBCCOMMONPARAMETERS_ADD_IND_DB2] (non-z/OS)
 - [DBCCOMMONPARAMETERS_ADD_IBM390_DB2] (z/OS)

3. Start R3SETUP:

```
./R3SETUP -f MCO DMIG_UNIX_db2.BASE.R3S
```
4. Continue and finish the installation as usual.

3.3.2 SAP Basis Release 6.10 and later

We did not perform this action. The procedure follows standard migration techniques. For details refer, to the SAP documentation. Also, perform the steps listed in 3.4, “Steps after installation or merge” on page 37.

3.4 Steps after installation or merge

Independent of the SAP Basis Release, after installation or merge, the following procedures are very important to ensure flawless operation of *all* SAP components in the MCO D installation. Complete the following steps:

1. SAP Kernel patches
Apply the latest SAP Kernel patches available on the SAP Service Marketplace quick link PATCHES at:
<http://service.sap.com/patches>
2. DDIC corrections
Apply the latest DDIC corrections and Basis Support Packages. For details, refer to *DB2/390: DDIC corrections (Releases 4.6A, 4.6B, 4.6C, 4.6D)*, SAP Note 184399 and *DB2/390: DDIC corrections (6.10, 6.20)*, SAP Note 407663.
3. CCMS transports
Import the latest CCMS transport available. For details, refer to *DB2/390: Newest version of the CCMS 4.6C*, SAP Note 217468, *DB2/390: Latest version of CCMS 4.6D*, SAP Note 337776, and *DB2/390: CCMS corrections (6.10, 6.20)*, SAP Note 427748.
4. DB2 maintenance
Ensure that the DB2 subsystem on the target system is at the correct preventive maintenance level. For details, refer to *DB2/390: APAR List*, SAP Note 081737 and *DB2/390: PTF check tool*, SAP Note 183311.

Important: Please note that the fixes listed need to be applied to every SAP component in an MCO D installation. That also includes the one that is there before other SAP systems are merged into the subsystem.

For more details about the fixes that we apply to our systems, refer to 7.1.1, “Required patches” on page 128.

3.5 Minimizing migration down time

The main drawback of the SAP standard migration procedure is that the down time needed to export and import the structure and data can be very, very long. If the system to be migrated is large, these processes may last several days or even weeks. In most cases, this is not acceptable, in particular when dealing with a production system.

There are two ways to minimize the down time:

- ▶ Incremental migration
- ▶ Merging without moving the data

3.5.1 Incremental migration

The basic idea of incremental migration (IMIG) is to completely migrate the largest tables while the SAP component is still productive. This is achieved by the following procedure:

1. Install a temporary SAP Basis system running on the target database.
2. Establish an SAP remote connection between the source system and the temporary SAP Basis system.
3. Create triggers and log tables on the critical tables to monitor all changes to these tables.
4. Export all critical tables while the source system is still productive.
5. Import the critical tables' data into the target database (MCOD system). Note that the data is inconsistent. We are importing an arbitrary state of the table at some point of time *after* the triggers and log tables were created.
6. Adjust the data in the target system using the remote connection and the changes monitored in the log tables.
7. Shut down the source system.
8. Perform a standard SAP migration of the remaining tables.

The advantage of this procedure is that the source system is up and running while the critical tables are migrated. Usually, the largest 50 tables make up for more than 80% of the total data in an SAP component. Once this step is finished, the migration of the remaining tables (step 8) can be easily done within one or two days. Only this last step contributes to the down time.

The complete procedure has been automated and can also be used for cross-database migrations. It is available as an add-on functionality to the SAP standard migration procedure. See *DB2/390: Incremental Migration to DB2/390*, SAP Note 353558 for details.

3.5.2 Merging without moving the data

It is also possible to merge SAP components without moving the data. The procedure can only be applied if the source and target system are running on the same DB2 version. This is explained in detail in Chapter 4., “Merging SAP components without moving data” on page 41.

Archived

Archived

Merging SAP components without moving data

This chapter describes the procedure for merging one or more SAP components without moving the underlying DB2 data. This is done by defining the source DB2 objects in the target DB2 subsystem. The source DB2 VSAM data sets are then updated using the DB2 REPAIR utility and renamed so that they can be accessed from the target DB2 subsystem.

In this chapter, we discuss the following:

- ▶ Considerations for using this procedure
- ▶ Planning steps
- ▶ Preparation steps
- ▶ Migration steps
- ▶ Post-migration steps

For additional information, refer to:

- ▶ Appendix A, “Database layout” on page 145
- ▶ *DB2/390: RSDB2MAS – new version*, SAP Note 330289
- ▶ *DB2 Universal Database for OS/390 Data Sharing: Planning and Administration Version 6*, SC26-9007

- ▶ *DB2 Universal Database for OS/390 and z/OS Data Sharing: Planning and Administration Version 7*, SC26-9935
- ▶ *DB2 UDB for OS/390 and z/OS Diagnosis Guide and Reference Version 7*, LY37-3740
- ▶ *DB2 Universal Database for OS/390 and z/OS: Utility Guide and Reference Version 7*, SC26-9945
- ▶ *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, SAP document available on the SAP Service Marketplace quick link INSTGUIDES at:
<http://service.sap.com/instguides>
- ▶ *IBM DB2 Administration Tool for z/OS User's Guide Version 4 Release 1*, SC27-1601

4.1 Considerations for using this procedure

This section discusses issues to consider when deciding to merge SAP components without moving the underlying DB2 data. We also refer to this as the *merge in place* procedure.

Attention: The procedure described here is *not trivial* and data loss may occur if you do not follow the instructions closely. Read the instructions completely before beginning this procedure. Do not proceed if you do not have the required skills to prevent potentially irrecoverable loss of data.

4.1.1 Reasons for choosing the merge in place procedure

The merge in place procedure is technically more complex than merging SAP components using SAP tools, as discussed in Chapter 3, “MCOD installation and merge using SAP tools” on page 27. However, a merge in place approach is recommended when:

- ▶ The volume of data to be merged and system availability requirements make the time required to use the SAP tools to export and import the data unfeasible.
- ▶ The volume of data and the availability of disk space during the merge process will not support simultaneous copies of the data in the source system, the target system, as well as an exported copy of the data.

4.1.2 Limitations

The following issues are not fully automated in our merge in place procedure. The handling of these issues may affect whether you decide to use the merge in place procedure or whether you decide to merge SAP components using SAP tools.

- ▶ The systems we merge are non-data sharing. However, the majority of the steps for the merge in place procedure is the same for data sharing or non-data sharing systems.
- ▶ The systems we merge contain no large objects (LOBs). This supports systems prior to and including SAP Basis Release 4.6C; however, this needs to be considered for systems later than this.
- ▶ The systems we merge contain a small number of partitioned tablespaces. Migration of partitioned tablespaces is addressed in our procedure although it is not fully automated.
- ▶ We do not automatically migrate any multi-VSAM tablespaces, that is, those with suffixes of A002, A003, and so forth.

- ▶ SAP creates several indexes on the DB2 catalog. These indexes are not moved as part of this procedure, because we assume that the MCOB landscape that we are merging into already contains those indexes. Additionally, SAP creates tables, for example, PLAN_TABLE, DSN_STATEMENT_TABLE, and DSN_FUNCTION, that do not use the SAP storage groups and may not use the same catalog alias as other SAP objects. These tables are not migrated because they are created automatically when first using the explain function within transaction DB2.

4.1.3 Tools we used

All of the steps for this procedure could be generated manually. However, the number of objects to be processed and the number of steps required make the use of REXX procedures beneficial. REXX is used to generate some of the DDL for defining the source objects in the target DB2 subsystem, as well as to perform many of the subsequent steps required in the merge procedure.

The DB2 Administration Tool for z/OS Version 4 Release 1 is also used for generating DDL (in the rest of this book, the DB2 Administration Tool for z/OS is referred to by its short name of DB2 Admin). The output from DB2 Admin is then tailored using REXX to meet our requirements.

4.2 Planning considerations

This section discusses planning considerations for the merge in place procedure.

4.2.1 Decide on source and target DB2 subsystem

When two SAP components are to be merged, a decision on which DB2 subsystem will be the source and which subsystem will be the target must be made. The following factors influence this decision:

- ▶ System size
If the system with fewer DB2 objects is selected as the source system, the merge procedure will take less time.
- ▶ DB2 log RBA values (see “Highest log RBA requirement” on page 22)
If the system chosen as the target system has RBA values lower than that of the source system, then once the source objects are merged into the target system:
 - All applications running on the target system must be stopped, and the target system must be cold started.

- All objects in the target system must be image copied. This is required for all objects in that DB2 subsystem, not just those that were merged from the source system.
- ▶ System availability requirements
During some steps of the merge in place procedure, the source system will be unavailable. However, the target system is available most of the time throughout the entire process.

4.2.2 Decide on naming conventions for merged system

There is a potential for naming conflicts to occur when SAP components are merged into an MCO landscape. We rename the following source system items to prevent conflicts:

- ▶ Storage group: The storage group names used in the source system are renamed so that are unique when they are merged into the target system. We alter the first three characters of each STOGROUP name to achieve this.
- ▶ VCAT name: We chose to rename the high-level qualifier for the tablespaces and indexspaces to uniquely identify each component within the MCO landscape. Additionally, renaming the VCAT name ensures unique VSAM data set names. Note that the aliases for the old and new VCATs relate to the same ICF user catalog.
- ▶ Schema: The schema is used as the owner of all of the DB2 objects within the SAP system, so it must have a unique name for each component within an MCO landscape.
- ▶ Database name: We regenerate the name of any source databases that already exist in the target system because database names must be unique within a DB2 subsystem. This is further discussed in 4.4.5, “Source and Target: Create and populate metadata tables” on page 53.

4.2.3 System availability issues

Our intention is to maximize system availability during the merge in place procedure. While some preparation and post-migration steps can be performed without exclusive access to the source DB2 objects, the following steps must be done with the source SAP system shutdown:

- ▶ Execute the REPAIR TABLESPACE utility on every source tablespace.
- ▶ Execute the REPAIR INDEX utility on every source indexspace.
- ▶ Use IDCAMS ALTER to rename all of the VSAM data sets used by every source tablespace and indexspace.
- ▶ Execute the REPAIR LEVELID utility on every source tablespace.

We perform the merge in place procedure on an SAP Basis Release 4.6C system, which contains approximately 3,300 tablespaces and 7,000 indexspaces, initially created with DEFINE NO. However, around 1,100 tablespaces and 2,700 indexspaces are not empty and have data sets defined for them. Table 4-1 shows the execution times for these steps.

Table 4-1 Execution times for merge in place steps for SAP 4.6C Basis only

Step	Execution time (approximate)
REPAIR TABLESPACE	3 min.
IDCAMS ALTER TABLESPACE	8 min.
REPAIR LEVELID TABLESPACE	3 min.
REPAIR INDEX	8 min.
IDCAMS ALTER TABLESPACE	20 min.
REPAIR LEVELID INDEXSPACE	8 min.

Note: The execution times are only given as an indication based on our experience. These times may not be the same in your environment.

4.2.4 System backup and recovery issues

Important: Backup and recovery planning is essential when using the merge in place procedure. The procedure is complex with many steps, including the execution of the DB2 REPAIR utility against the DB2 tablespaces and indexspaces containing SAP data.

As a minimum, we recommend performing a full backup of the source SAP system and a full backup of the target DB2 catalog and directory before commencing 4.4, “Preparation steps” on page 51. These steps involve the creation and population of metadata tables and the definition of source objects in the target catalog.

In addition, it is beneficial to perform another full backup of the source SAP system before commencing 4.5, “Migration steps” on page 60, because these steps modify the source tablespaces and indexspaces using the DB2 REPAIR utility and the IDCAMS ALTER utility. If a recovery is needed during the migration steps, and the preparation steps are done over an extended period of time, then this backup will provide a more current recovery point than a backup performed prior to the preparation steps.

4.2.5 Hardware-based backup options

Most customers implement their SAP systems on disk technology that includes mirroring technology and the ability to “break” mirrors rapidly. The use of this technology is widely referred to as *split mirror backup* techniques. For detailed descriptions of these techniques in a DB2 for z/OS environment, see *Storage Management for SAP and DB2 UDB: Split Mirror Backup/Recovery with IBM’s Enterprise Storage Server (ESS)*, white paper by Sanjoy Das, Siegfried Schmidt, Jens Claussen, and BalaSanni Godavari, available at:

<http://www.storage.ibm.com/hardsoft/products/sap/smdb2.pdf>

The flexible use of this technology is particularly useful in the case of major system changes, such as SAP updates and migrations.

The different storage vendors vary in their actual implementation, and so do the tools and commands to establish, break, and re-establish mirrors. However, all share the same principles for the usage suggested in the merge in place procedure. There are two main types of split mirror backups. In the first type, the broken mirror remains only in the storage subsystem with the ability to restore the system quickly to these volume level copies. The second variation is that these copies are externalized to some other media, for example, cartridge or tape. The first type we will call a *soft copy* and the second, externalized copies, a *hard copy*. Depending on the storage subsystem type, it may be possible to establish and maintain one or more soft copy images within the subsystem depending on limits of disk storage and device address ability.

For the purpose of the following exercise, the use of soft copy and hard copy backups at various points are suggested. Soft copies have the advantage that they can be restored quickly, and the disadvantages that the subsystem may only be capable of maintaining one copy or may require additional disk storage per copy, or both. Hard copies have the advantages that you can maintain as many copies as required, and that they may be kept for a long time. Hard copies have the disadvantage that they take time to externalize the mirror copy to disk, and they consume tape media resources.

4.2.6 Merge in place procedure checklist

The following tables provide checklists that can be used during the procedure.

Preparation steps

The steps in Table 4-2 on page 48 do not require exclusive use of the source or target systems.

Table 4-2 Preparation steps for merge in place procedure

Preparation steps	
4.4.1, "Source: Create full backup of source system" on page 51	
4.4.2, "Target: Create full backup of catalog and directory of target" on page 52	
4.4.3, "Source: Redefine empty tablespaces as DEFINE NO in source" on page 52	
4.4.4, "Source: Reorganize tablespaces" on page 52	
4.4.5, "Source and Target: Create and populate metadata tables" on page 53	
4.4.6, "Source and Target: Define source objects in target system" on page 55	
4.4.7, "Target: Update metadata tables" on page 59	
4.4.8, "Target: Stop newly created databases" on page 59	
4.4.9, "Target: Prepare RUNSTATS JCL" on page 60	

Migration steps

The steps in Table 4-3 require exclusive use of the source system.

Table 4-3 Migration steps for the merge in place procedure

Migration steps	
4.5.2, "Source: Perform full backup of source DB2 subsystem" on page 61	
4.5.3, "Source: Start source databases for UT access" on page 61	
4.5.4, "Source: Execute REPAIR on page set header pages" on page 61	
4.5.5, "Source: Stop source DB2 subsystem" on page 62	
4.5.6, "Target: Delete target data sets and rename source data sets" on page 62	
4.5.7, "Target: Start target databases in target system" on page 63	
4.5.8, "Target: Execute REPAIR LEVELID on tablespaces and indexes" on page 63	
4.5.10, "Target: Execute RUNSTATS" on page 65	
4.5.11, "Target: Perform IMAGE COPY" on page 65	
4.5.12, "Configure SAP application server" on page 65	

Post-migration steps

The steps in Table 4-4 do not require exclusive use of the source or target system.

Table 4-4 Post-migration steps for the merge in place procedure

Post-migration steps	
4.6.1, "Target: Verify access to objects using RSDB2MAS (optional)" on page 68	
4.6.2, "Target: Alter the space allocations for TS and IX (optional)" on page 68	

4.3 Merge in place scenario

In this section, we describe how to merge two SAP systems without moving the underlying DB2 data. To demonstrate this procedure, we use the SAP systems shown in Figure 4-1 on page 50.

We chose the SAP system BLU as the source system. BLU is based on SAP Basis Release 4.6C and uses the DB2 V7 subsystem DB7S. We chose to merge the BLU system with the target DB2 V7 subsystem DB7T, where the SAP system RED, based on SAP Basis Release 6.20, is already installed.

At a high level, the merge in place procedure consists of the following steps:

1. Definition of the source BLU objects in the target DB7T system.
2. Stop update activity on BLU objects from DB7S.
3. Use the DB2 REPAIR utility to update the BLU page set header pages.
4. Rename the DB2 VSAM data sets used by BLU.
5. Use the REPAIR LEVELID utility against the BLU objects so that they can be accessed from DB7T.

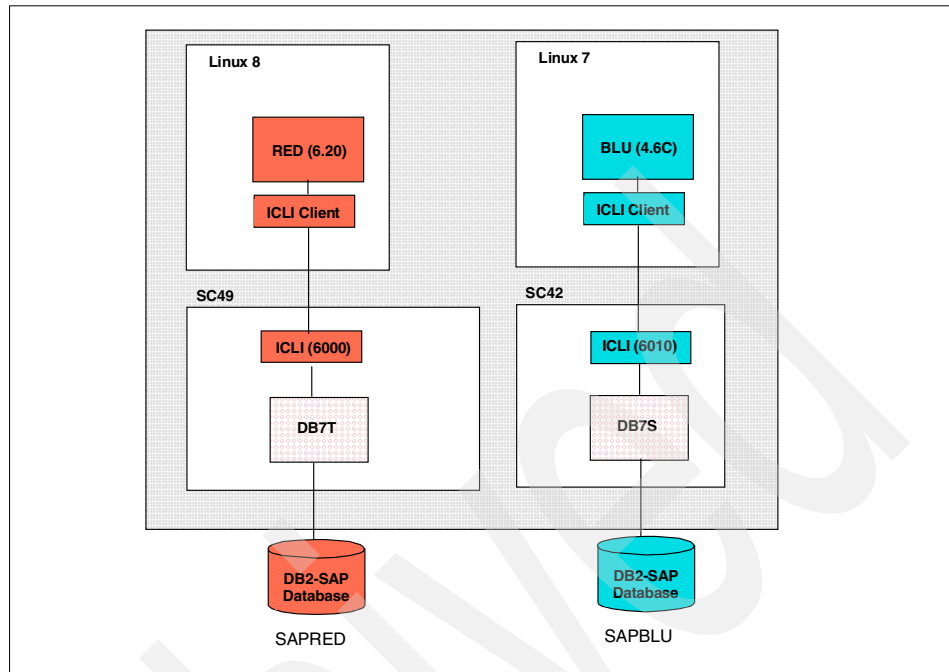


Figure 4-1 Merge in place scenario: Initial configuration

Once this procedure completes, the BLU objects that were in DB7S are now accessed from DB7T. DB7T now contains the BLU and RED components. The final configuration is shown in Figure 4-2 on page 51.

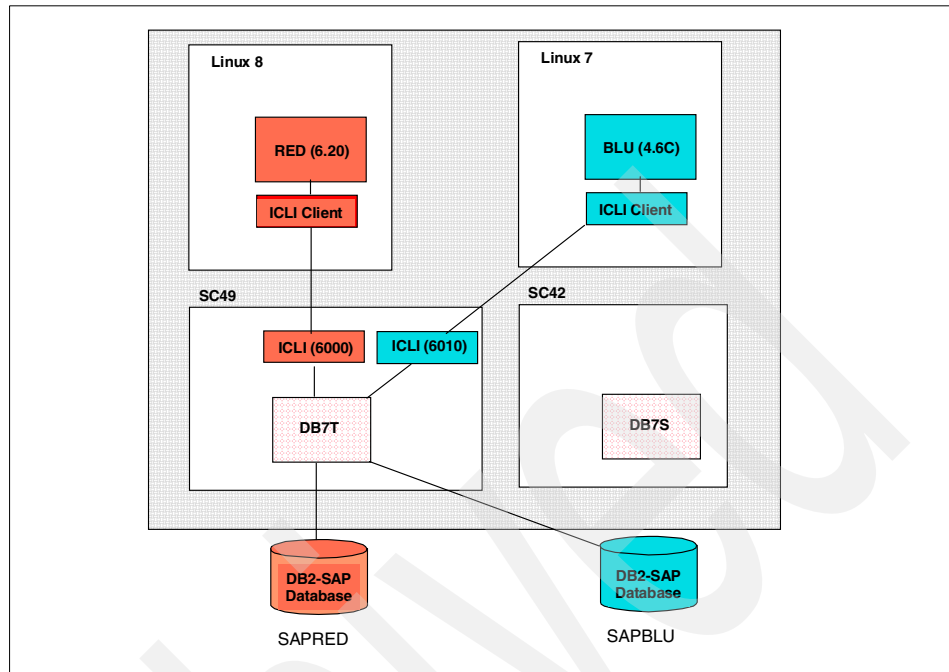


Figure 4-2 Merge in place scenario: Final configuration

4.4 Preparation steps

These steps are performed in preparation for the migration. They do not require exclusive use of the source or target DB2 subsystems.

For each step, an indication of whether the step is performed on the source or target DB2 subsystem is given.

4.4.1 Source: Create full backup of source system

As discussed in 4.2.4, “System backup and recovery issues” on page 46, it is highly recommended that a full backup of the source SAP system is performed before beginning the migration process. A backup at this time would be used if a full system recovery was needed during the preparation steps.

4.4.2 Target: Create full backup of catalog and directory of target

During the merge process, the source DB2 objects will be recreated in the target system. This updates the target DB2 catalog and directory. If a backout of the process is required, these objects can be removed using DB2 DROP statements. However, it is highly recommended to perform a full backup of the target DB2 catalog and directory before beginning the following steps.

4.4.3 Source: Redefine empty tablespaces as DEFINE NO in source

The number of tablespaces to be merged from the source system into the target system will impact the time taken to perform the migration.

Even though a source tablespace may be empty, if it was created with the storage attribute DEFINE YES, then the underlying VSAM data set exists. During the merge process, the REPAIR utility will be executed against it and it will be renamed, despite having no data that needs to be moved.

To reduce the number of empty tablespaces, it is recommended that the SAP utility program RSDB2MAS is executed on the source system prior to the migration.

RSDB2MAS determines which tablespaces are empty, but have underlying VSAM data sets (that is, they were created as DEFINE YES but contain no data). RSDB2MAS can then redefine these tablespaces as DEFINE NO, which removes the underlying VSAM data set.

Refer to *DB2/390: RSDB2MAS – new version*, SAP Note 330289 for additional information about using RSDB2MAS.

4.4.4 Source: Reorganize tablespaces

The REORG utility must be run against any tablespace that contains a table that meets all of the following criteria:

- ▶ The table contains columns that are all fixed length.
- ▶ The table has a column added using ALTER TABLE... ADD COLUMN....
- ▶ The tablespace has not had the REORG utility run against it since the ALTER was done.

This step is needed to ensure that the information in the source DB2 catalog accurately reflects the layout of the underlying VSAM data set.

It is possible to determine if any tablespaces have had an ALTER performed on them since they were created with the following query, as shown in Example 4-1 on page 53.

Example 4-1 Sample SELECT statement

```
SELECT DBNAME, TSNAME, MAX(ALTEREDTS)
FROM SYSIBM.SYSTABLES
WHERE CREATOR = 'SAPR3'
      AND TYPE = 'T'
      AND ALTEREDTS <> CREATEDTS
GROUP BY DBNAME, TSNAME
ORDER BY DBNAME, TSNAME;
```

This shows the most recent ALTER TABLE within each tablespace. However, there is no way to tell from the DB2 catalog whether the ALTER consisted of adding a column.

4.4.5 Source and Target: Create and populate metadata tables

We find it beneficial to store metadata about the source and target objects that we are merging.

This information is extracted from the source and target DB2 catalogs and stored in DB2 tables. Additionally, the tables are populated with naming conventions we want to use for existing and new objects. These metadata tables are then used throughout the rest of the merge in place procedure for generating and tailoring job control language (JCL) and DDL.

Initially, the tables are loaded with information from the source system. They are then moved to the target system to obtain additional information to be used for the DDL generation steps. After the DDL is created, they are loaded with information about the created objects.

Note: Because metadata is extracted and used throughout the merge procedure, it becomes important to ensure that the database layout remains constant in both systems, from a certain point in the migration through to the completion of the migration. Because SAP remains available for users during most of this time, the control of this rests with ceasing DB2 administrative activities, SAP Transport Management System (TMS) activities, and any SAP transactions that can cause DDL execution. This means all normal SAP usage that results in changes to the content of tables can continue. You may choose to set SAP System Change options to prevent changes and lock the TMS system to prevent any transport activity (bear in mind that transports not related to DDIC objects can still be performed).

Source and Target: Create the metadata tables

The metadata tables we use are as follows:

- ▶ ZMCOB_STOGROUP
- ▶ ZMCOB_DATABASE
- ▶ ZMCOB_TABLESPACE
- ▶ ZMCOB_TABLES
- ▶ ZMCOB_INDEXES

These tables can be created by SAP transport, which can be downloaded from the Internet as described in Appendix E, “Additional material” on page 265. Alternatively, the DDL for creating these tables is included in Appendix B, “Merge in place: Working with metadata tables” on page 153.

Source: Populate the metadata tables

Once the tables are defined in the source DB2 subsystem, the following REXX procedures are used to populate the tables with metadata:

- ▶ SETUPSTG
- ▶ SETUPDB
- ▶ SETUPTS
- ▶ SETUPTAB
- ▶ SETUPIX

These REXX procedures and the JCL to execute them are included in “Populating metadata tables” on page 159.

Source and Target: Move metadata tables across systems

Once the metadata tables are loaded with information from the source system, they are then moved to the target system to obtain additional information to be used for the DDL generation steps. The JCL to unload the metadata tables on the source system, reload them into the target system, and reset copy pending status on tablespaces are included in “Moving metadata tables across systems” on page 178.

Target: Work with metadata tables

After the metadata extracted from the source system has been transferred to the target system with UNLOAD/LOAD, it is then necessary to run a REXX procedure to detect and correct any clashes that exist in database names between the two systems. This is achieved by running the REXX procedure DUPLICDB.

The procedure DUPLICDB finds any clashes and tries to resolve them by regenerating the last three characters of the database name to new random values. It tries this up to 20 times for each clash, and if it cannot resolve the clash, it reports this. If this error occurs, then you will need to change the value of the target database name in the migration tables to a unique value. This action is very unlikely, and all clashes should be resolved.

The JCL to move the metadata from the source to the target, the procedure DUPLICDB, as well as the JCL to submit it, are included in “Working with metadata tables” on page 182.

4.4.6 Source and Target: Define source objects in target system

In order to merge the tables from the source to the target system, the definitions of the source objects need to be added into the target DB2 catalog.

Important: A fundamental requirement of the merge in place procedure is to retain the OBIDs of the tables when they are merged from the source to target systems. When the OBID is retained, only the header pages of the tablespace must be REPAIred. If the OBID of the table is changed, then every single page of the table must be REPAIred, and this is not feasible.

We use a mix of methods for generating the DDL to create the source objects in the target system:

- ▶ For generating the DDL for creating stogroups and views, we extract the data out of the DB2 catalog tables directly using SQL statements and REXX procedures.
- ▶ For generating the DDL for databases, tablespaces, tables, and indexes, we use the reverse engineering functionality of DB2 Admin. Output from DB2 Admin is tailored using REXX procedures.

We assume that the only objects in the system are those in a typical SAP installation. Therefore, we only generate DDL for stogroups, databases, tablespaces, tables, indexes, and views. Objects, such as synonyms, aliases, comments, relationships, and triggers, are not expected to exist in the system, so these are not included in our generation procedures.

We chose to write SQL queries and REXX procedures for creating stogroups and views rather than using DB2 Admin. DB2 Admin generates objects for a given database, and, as views and stogroups may be associated with more than one database, duplicate stogroups and views would have been created.

Before executing DB2 Admin, we add extra indexes to the DB2 catalog to improve the performance of the DDL generation program ADB2GEN. Details of these indexes are in Chapter 2, “Customizing DB2 Admin,” in *IBM DB2 Administration Tool for z/OS User’s Guide Version 4 Release 1, SC27-1601*.

Source: Generate DDL for storage groups

The DDL for creating storage groups can easily be generated by running simple SQL (see Example 4-2) against the source DB2 catalog. Make sure that the naming convention follows the requirements of your MCO landscape and add the correct SET CURRENT SQLID when you run the generated DDL on the target system (for example, in our case, SAPBLU).

Example 4-2 Generating DDL for storage groups

```
SELECT 'CREATE STOGROUP BLU'!!  
      SUBSTR(NAME,4,3)!!  
      ' VOLUMES (''*'' ) VCAT MCDBLU ;'  
FROM SYSIBM.SYSSTOGROUP  
WHERE CREATOR='SAPR3' ;
```

Source and Target: Generate DDL for DB, TS, TB, and IX

DB2 Admin is used to generate the DDL for databases, tablespaces, tables and indexes. We use a two-step process for this generation:

1. Target: Generate JCL to invoke DB2 Admin

The REXX procedure JCLGEN and its subroutines JCLGENDB, JCLGENTS, JCLGENTB, and JCLGENIX produce JCL for invoking DB2 Admin. These REXX procedures use the metadata tables in the target system as parameters to generate the JCL.

2. Source: Execute JCL to invoke DB2 Admin

The size of the generated JCL prevents us from using TSO SUBMIT in our system. So, once the JCL is generated, it is copied directly to the JES2 internal reader for submission. The JCL job cards contain TYPRUN=HOLD, so release the jobs (using the “a” action character against each job in SDSF) when you are ready to run them. The order in which these jobs are run is not important. These jobs run on the source system to extract information from the source DB2 catalog.

The REXX procedure JCLGEN (and its subroutines) and the JCL to execute it and submit the generated JCL are included in “Generating DDL using DB2 Admin” on page 198.

Target: Tailor DDL for DB, TS, TB, and IX

Once DDL is produced by DB2 Admin, the REXX procedures TAILORTS, TAILORTB, and TAILORIX are employed to tailor the generated DDL as follows:

- ▶ Reduce the primary quantity allocations on tablespaces and indexes (optional).

The VSAM data sets being created by the CREATE TABLESPACE and CREATE INDEX statements in the target system will not be used, because the source VSAM data sets will be “used in place” in the target system. As such, these VSAM data sets being created can be allocated at minimum quantities to conserve disk space during the merge process. After the migration is completed, the correct primary quantity must be restored using ALTER statements.

- ▶ Include the OBID clause on the CREATE TABLE statements so that the source table OBIDs are retained.

The REXX procedures TAILORTS, TAILORTB, and TAILORIX, and the JCL to execute them, are included in “Tailoring the output from DB2 Admin” on page 217.

Source: Generate DDL for views

The REXX procedure PCREATVI generates DDL for views. More information about this REXX procedure is provided in “Generating DDL for views” on page 255.

Source and Target: DDL for altering object sizes (optional)

The REXX procedures PTEPTSAL and PTEPINAL generate DDL for altering object sizes after the migration has completed. More information about these REXX procedures is provided in “Generating DDL for altering object sizes” on page 259.

Target: Generate DDL for filler tablespaces (optional)

We need to create our tables using specific OBIDs. However, we create tablespaces before tables, and we cannot specify what OBIDs these tablespaces will use. The database layout determines whether this step is required:

- ▶ When there is only one tablespace per database, then this step is not required, because we know that each table will contain OBIDs larger than its tablespace.
- ▶ When there are multiple tablespaces in a database, if all tablespaces are created before tables, then it is possible that a tablespace will use an OBID we require for a table.

Therefore, this step is only required for systems that contain multiple tablespaces per database (SAP Basis Release 6.10 and later).

When this step is required, we influence which OBIDs our tablespaces use by:

1. Creating databases.
2. Creating filler tablespaces to reserve the OBIDs we want for tables within each database.
3. Creating our real tablespaces.
4. Dropping the filler tablespaces, freeing up the OBIDs we want for tables.
5. Creating our tables and other objects.

The REXX procedure FILLERTS generates the DDL for creating and dropping filler tablespaces. This generated DDL must be run against the metadata tables on the target system.

More information about this REXX procedure is provided in “Using filler tablespaces to reserve OBIDs” on page 230.

Target: Execute the generated DDL in the target subsystem

Once the DDL is generated and tailored, you should review it *briefly* to verify that it adheres to the naming conventions you requested, such as schema name, CURRENT SQLID, and stogroup names. If these names are not correct, you may need to update the data in the metadata tables and regenerate the DDL.

After you have reviewed the DDL, it can be executed against the target DB2 subsystem using DSNTEP2. Ensure that the DDL is run in the following order:

1. Create storage groups.
2. Create databases.
3. Create filler tablespaces (if required).
4. Create tablespaces.
5. Drop filler tablespaces (if required).
6. Create tables.
7. Create indexes.
8. Create views.

Samples of the DDL that we use are provided in “Sample DDL for creating objects” on page 227.

Before defining these objects, you should ensure that:

- ▶ The VCAT/alias you are using for the stogroups has been defined in the *same* ICF catalog as the one used for the source VCAT (to be able to rename the VSAM data sets).
- ▶ That SMS ACS routines are updated if stogroups are SMS managed.
- ▶ The creator you are using for the new objects has the authority to create objects in the target DB2 subsystem.

4.4.7 Target: Update metadata tables

After the objects are defined in the target DB2 catalog, additional information can be updated in the metadata tables. This information includes:

- ▶ DBIDs for new databases
- ▶ PSIDs for new tablespaces
- ▶ Names of the indexspaces
- ▶ ISOBIDs for the new indexes

This is achieved by running the REXX procedures POSTDDL and POSTDDL2. These REXX procedures and the JCL to execute them are included in “Updating metadata tables” on page 189.

4.4.8 Target: Stop newly created databases

The following steps require exclusive access to the newly created tablespaces and indexspaces in the target DB2 subsystem. This is achieved by issuing the DB2 STOP DATABASE command against each newly created database in the target system.

This does not impact the availability of either the source or the target system because the associated VSAM data sets are not currently being used by any SAP system.

The REXX procedure PDSNDBST can be used to generate the JCL for running the DB2 STOP DATABASE command against each database. More information about this REXX procedure is provided in “Issuing DB2 START and STOP DATABASE commands” on page 241.

4.4.9 Target: Prepare RUNSTATS JCL

After the source objects have been merged into the target system, the RUNSTATS utility needs to be executed to update catalog information about the merged objects. Because this JCL is run while the source SAP system is still unavailable, it is advisable to prepare it in advance if that is possible.

Existing RUNSTATS utility JCL needs to be updated before being executed against the merged objects with the following information:

- ▶ New DB2 subsystem
- ▶ New schema
- ▶ New database names

The metadata tables can be used for reference when updating the RUNSTATS utility JCL.

4.5 Migration steps

The following steps require exclusive use of the DB2 objects in the source SAP system. This marks the start of the down time of the source SAP system. For each step, an indication of whether the step is performed on the source or target system is given.

4.5.1 Source: Stop all update activity on source system

In this step, all update activity to the source DB2 subsystem is stopped. After the source objects are moved into the target system, recovery information will not be available until after an image copy is done. Therefore, it is essential that all work is completed on the source system before the migration. Perform at least the following steps:

- ▶ Shut down the SAP application servers.
- ▶ Stop the ICL servers.
- ▶ Verify that there is no outstanding activity on the source subsystem using the following commands:

```
DIS THREAD(*)  
DISPLAY UTILITY(*)  
DISPLAY DATABASE(*) SPACE(*) RESTRICT
```

Continue only if no threads, no utilities, and no databases with spaces in restricted states are returned. Resolve stopped utilities and restricted space states before proceeding.

You may want to stop the DB2 subsystem and start it in restricted mode to ensure you have exclusive access to the system. Use the START DB2 ACCESS(MAINT) command for this.

4.5.2 Source: Perform full backup of source DB2 subsystem

A full backup of the source system may have been done prior to the pre-migration steps in 4.4.1, “Source: Create full backup of source system” on page 51. It would be beneficial to perform another backup at this point in the migration, because the following steps will execute the DB2 REPAIR utility against the source tablespaces and indexspaces. A backup at this stage of the migration would provide a more recent recovery point, particularly if the pre-migration steps have been performed over an extended period of time.

4.5.3 Source: Start source databases for UT access

The source tablespaces and indexspaces must be put into utility access mode so that the DB2 REPAIR utility can be run against them. This is achieved by issuing the DB2 START DATABASE ACCESS(UT) command.

The REXX procedure PDSNDBST can be used to generate the JCL for running the DB2 START DATABASE ACCESS(UT) command against each database. More information about this procedure is provided in “Issuing DB2 START and STOP DATABASE commands” on page 241.

4.5.4 Source: Execute REPAIR on page set header pages

When creating databases and tablespaces, it is not possible to specify a DBID or PSID, respectively. For a source tablespace to be accessed by the target DB2 subsystem, the DBID and PSID on the header page of each tablespace must be changed to match the DBID and PSID that are in the target DB2 catalog. This is achieved by using the DB2 REPAIR utility.

In addition, the following information is updated by the REPAIR utility:

- ▶ Source DB2 subsystem name to that of the target DB2 subsystem
- ▶ VCAT and storage group name to those of the merged objects

The DB2 REPAIR utility is also executed against each index, because they require similar changes.

The REXX procedures PREPTSOP and PREPINOB are used to generate the REPAIR statements. More information about these procedures is provided in “Executing REPAIR on page set header pages” on page 244.

Important: These procedures create samples for partitioned tablespaces and the partitioning index. However, these samples have to be adapted manually. These samples assume a partition size of 4 GB, thus the page number of the first page of portion 2 will be X'00200000'. This value is different if the partition size is different. This value can be derived from the DSSIZE field and PGSIZE field in SYSIBM.SYSTABLESPACE.

In our case, the same values we use for the corresponding tablespace identifies the corresponding header pages in the respective part of the partition.

The resulting jobs must run against the source DB2 subsystem!

Important: It is crucial that these jobs run successfully. In case REPAIR fails on objects, you may still be able to access the data, but other tasks may fail.

4.5.5 Source: Stop source DB2 subsystem

The following step renames the source tablespaces and indexspaces. Because the source DB2 subsystem is no longer able to access these objects once they are renamed and the rename step requires exclusive access to the source objects, stop the source DB2 subsystem.

You may want to disable the source DB2 subsystem so that it is not started. Perhaps introducing a JCL error into the DB2 startup procedure would achieve this aim.

4.5.6 Target: Delete target data sets and rename source data sets

When the source tablespaces and indexes are defined in the target catalog (see “Source and Target: Define source objects in target system” on page 55), a new VCAT name and, possibly, a new database name are used. Similarly, the index may be created with a new indexspace name.

For DB2 to access the source VSAM data sets from the target DB2 subsystem, the VSAM data sets must be renamed to match the names in the target DB2 catalog.

When the source tablespaces and indexspaces are defined in the target catalog, a VSAM data set may have been created. We distinguish two cases:

- ▶ If the source tablespace or index was created with DEFINE NO, and it contains no data, no VSAM data set exists in the source system. In this case, no VSAM data set is created when the object is defined in the target system.
- ▶ If the source tablespace or index was created with DEFINE YES, or it contains data, the VSAM data set exists in the source system. When it is defined to the target system, a VSAM data set is created. These newly created VSAM data sets will not be used, because the existing source VSAM data sets will be reused in the target DB2 subsystem. So, it is these VSAM data sets that must be deleted

The REXX procedures PVSATSDE and PVSAINDE are used to generate the DELETE statements. The REXX procedures PVSATSAL and PVSAINAL are used to generate the ALTER statements. More information about these procedures is provided in “Deleting created target data sets and renaming source data sets” on page 248.

Important: Our procedures do not handle the multi-data set tablespaces and the multi-data set indexspaces. A simple check to see if your installation has such types is a ISPF 3.4 list datasets <svcat>.DSNDBD.*.*.A002.

If you get more than the expected ones for partitions and partitioning indexes, note them and create the IDCAMS rename jobs yourself.

4.5.7 Target: Start target databases in target system

The merged databases can now be started in the target system. Issue the DB2 START DATABASE command against each of the newly created databases in the target system.

The REXX procedure PDSNDBST can be used to generate the JCL for running the DB2 START DATABASE command against each database. More information about this REXX procedure is provided in “Issuing DB2 START and STOP DATABASE commands” on page 241.

4.5.8 Target: Execute REPAIR LEVELID on tablespaces and indexes

Now that the source VSAM data sets have been renamed and repaired for use in the target system, the target DB2 subsystem will recognize the VSAM data sets. However, the LEVELID may not be compatible. This can be solved by executing the DB2 REPAIR LEVELID utility on each tablespace and index.

The REXX procedures PREPTSLV and PREPINLV are used to generate and tailor the REPAIR. More information about these procedures is provided in “Executing REPAIR LEVELID on tablespaces and indexes” on page 253.

4.5.9 Cold start target system (optional)

This step is not required if the system with the higher RBA values was chosen as the target.

If the system with the lower RBA values was chosen as the target, a cold start must be done to introduce higher RBA values into the target system. To perform a cold start, use the DSNJU003 utility to insert a conditional restart record into the bootstrap data set (BSDS), as shown in Example 4-3. The DB2 subsystem can then be restarted, and the higher RBA values will be used.

Example 4-3 Using DSNJU003 to create a conditional restart record

```
//RESET EXEC PGM=DSNJU003
//STEPLIB DD DSN=DB7T7.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=DB7TU.BSDS01,DISP=SHR
//SYSUT2 DD DSN=DB7TU.BSDS02,DISP=SHR
//SYSIN DD *
CRESTART CREATE,STARTRBA=000900000000,ENRBA=000900000000,BACKOUT=NO
```

Important: Before performing the cold start, you must do the following:

- ▶ Shut down all SAP application servers.
- ▶ Stop all ICLI servers.
- ▶ Verify that there is no outstanding activity on the target subsystem using the following commands:

```
DIS THREAD(*)
DISPLAY UTILITY(*)
DISPLAY DATABASE(*) SPACE(*) RESTRICT
```

Continue only if no threads, no utilities, and no databases with spaces in restricted states are returned. Resolve stopped utilities and restricted space states before stopping the target subsystem.

After performing the cold start, you must perform an image copy on *all objects* in the target DB2 subsystem, not only the objects that are merged.

4.5.10 Target: Execute RUNSTATS

Important: This step is important. Access to all SAP data will be done by a tablespace scan until this step is performed.

Execute the DB2 RUNSTATS utility against the migrated tablespaces and indexes in the target system. This step is essential for the performance of the merged system. To reduce the time required to execute the RUNSTATS utility, it may be possible to initially run with a low sample size (for example, SAMPLE(010)) to populate the tables and, subsequently, perform a more thorough update of statistics once user activity has commenced.

All of the normal careful steps are required in updating statistics, especially regarding volatile tables, and the use of NPGTHRSH ZPARM, as documented in the *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, SAP document available on the SAP Service Marketplace quick link INSTGUIDES at:

<http://service.sap.com/instguides>

4.5.11 Target: Perform IMAGE COPY

Perform an image copy of the newly created objects in the target DB2 subsystem. Because the objects have been moved outside of DB2 control, until an image copy is performed, no DB2 controlled recoveries of individual objects can be made without the full image copy.

In addition, perform an image copy of the target DB2 catalog and directory, because they contain the new definitions for these objects.

4.5.12 Configure SAP application server

The following section details the required changes to SAP parameters to allow the SAP instances to connect to the new location of the SAP database in the new DB2 subsystem. These steps should be repeated for each application server that is part of the SAP instance being merged. Configuring the SAP application server may be done any time after stopping the source SAP system in 4.5.1, “Source: Stop all update activity on source system” on page 60.

Attention: We strongly recommend reinstalling the application server software, as described in “Set up the target application server” on page 35. It is the simplest and safest means of adapting the SAP instance to the new environment.

The following steps were sufficient to allow our testing to be conducted successfully, but it is much safer to use SAP installation tools, because they will perform all required actions, some of which may be missing here.

Adapt SAP instance profiles after migration

If you choose not to reinstall the SAP application server software, you need to perform at least the following actions for components based on Basis Release 4.6x:

- ▶ Change the DEFAULT.PFL profile to contain the following lines:

```
db2/db2/schema=<new schema>
```

- ▶ Change the instance profile or profiles to contain the following lines:

```
db2/db2/ssid=<new DB2 SSID>  
db2/db2/hosttcp=<DB2 hostname>
```

- ▶ Change the .dbenv_<hostname>.sh file in \$HOME for <sid>adm to contain the lines:

```
DBS_DB2_SCHEMA=<new schema>  
R3_DB2_SSID=<new DB2 SSID>  
DBS_DB2_SSID=<new DB2 SSID>  
SAPDBHOST=<DB2 hostname>
```

If needed, manually export the environment variables DBS_DB2_SCHEMA, R3_DB2_SSID, DBS_DB2_SSID, and SAPDBHOST.

- ▶ Change the .dbenv_<hostname>.csh file in \$HOME for <sid>adm to contain the lines:

```
setenv DBS_DB2_SCHEMA <new schema>  
setenv R3_DB2_SSID <new DB2 SSID>  
setenv DBS_DB2_SSID <new DB2 SSID>  
setenv SAPDBHOST <DB2 hostname>
```

- ▶ In SAP transaction STMS, change the System Transport Tool data in the Domain Controller for this system, and distribute the configuration to all members. The following values must be changed:

```
R3_DB2_SSID  
DBS_DB2_SSID  
DBHOST
```

- ▶ After the system is started, and you can log on to the SAP system, in transaction DB2J under the Profile option, the following parameters need to be updated:
 - DB2 load library (general tab)
 - DB2 run library (general tab)
 - High Level Qualifier (storage tab)
 - Upload data set (upload tab) if required due to host change
 - SAPOSCOL directory and DEST (saposcol tab) if required due to host change
 - SDSF HASPINDEX (console tab) if required due to host change

For SAP Web Application Server Release 6.10 and later, the configuration file connect.ini has been introduced, allowing the specification of up to 10 database servers. Please refer to *DB2/390: New failover support from Release 6.10*, SAP Note 402078 for information to adapt this file during an MCODE merge.

In the case of DB2 data sharing, the Group Attach Name should be used instead of the DB2 SSID, as documented in *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20*, SC33-7959.

Run BIND and GRANT steps

If manually adapting the application server, you need to manually run the DB2 BIND and GRANT jobs that are normally performed by the SAP installation tools. Either copy the FOMEBIND.jcl and FOMEGRNT.jcl jobs from the SAP installation directory, or adapt the samples supplied in <hlq>.SAMPLIB, as documented in *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20*, SC33-7959.

Review and update the following values where required:

- ▶ DB2 subsystem (previously the source, this is now the target DB2 subsystem).
- ▶ Database schema (update to use the new owner of the DB2 objects).
- ▶ Plan name.

Start ICLI server if required

If you are changing the z/OS host name where the database resides, you will need to start an ICLI server on that host for communication. Follow the normal steps for running the ICLI server as a started task, job, or USS process, and if you chose to run the server with a changed port number (-PORT parameter to FOME<SAP rel>S), change the entry in the /etc/services file to the corresponding value.

4.6 Post-migration steps

At this point in the migration process, the source data is available in the target DB2 subsystem and the migrated SAP system can be used. The following steps are necessary to complete the migration. However, they do not require exclusive use of the migrated system.

4.6.1 Target: Verify access to objects using RSDB2MAS (optional)

This step is optional; however, it is recommended because it verifies that all SAP objects have been moved successfully and are accessible from the target system.

4.6.2 Target: Alter the space allocations for TS and IX (optional)

In order to save space during the migration, the source tablespaces and indexspaces can be defined to the target system with minimum space allocations. The underlying tablespaces to these definitions are deleted, and the source VSAM data sets are renamed for use by the target catalog, so these values are never even used.

After migration, the primary quantity and secondary values registered in the target DB2 catalog must be updated to the actual values that were in the source DB2 catalog. These values will be used if the DB2 REORG utility is run against this tablespace or indexspace or new extents are taken. This is achieved by running the DDL that is prepared in step “Source and Target: DDL for altering object sizes (optional)” on page 57.

Cloning one component out of an MCOD landscape

This chapter discusses the cloning of one component out of an MCOD landscape. We refer to it as MCOD cloning.

For additional information, refer to:

- ▶ *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287
- ▶ *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678
- ▶ *SAP Web Application Server 6.20: Homogeneous and Heterogeneous System Copy*, SAP document available on the SAP Service Marketplace quick link INSTGUIDES at:
<http://service.sap.com/instguides>
- ▶ Program directories:
 - *Program Directory for DB2 Management Tools Package (JDB661D)*, GI10-8193
 - *Program Directory for IBM DB2 UDB Server for OS/390 and z/OS: DB2 Management Clients Package (JDB771D)*, GI10-8218
 - *Program Directory for DB2 for OS/390 and z/OS: DB2 Administration Server for z/OS (HDAS810)*, GI10-8472

- ▶ PSP buckets:
 - 390 Enablement V6 (Upgrade DB2610, subset JDB661D)
 - 390 Enablement V7 (Upgrade DB2710, subset JDB771D)
 - DAS (for DB2/390 V7) (Upgrade DB2710, subset HDAS810)

5.1 MCOD cloning: Just another homogeneous system copy

In this section, we discuss the differences between homogeneous system copy and MCOD cloning.

5.1.1 Homogeneous system copy (HSC)

HSC is a process where an entire SAP system is copied to a target system. The target SAP database server has the same operating system (such as OS/390 or z/OS) and the same database system (such as DB2 UDB for OS/390 and z/OS) as the source SAP system. The DB2 version/release of the source and target subsystems must also be the same.

The copy (or clone) can be done using:

- ▶ SAP R/3 export/import tools
- ▶ Database-specific tools

In this section, we summarize the HSC options that are specific to DB2 for z/OS.

HSC methods

The method used for HSC is determined by your operational business requirements for both the source and the target systems:

- ▶ If high availability is required for the source system, it is necessary to make the copy without stopping the SAP application and impacting its availability to the end users. This method is known as *online* copy.

The definition of an online copy implies the following:

- Source subsystem active log activity is suspended (using DB2 SET LOG SUSPEND).
- Target restart method is conditional restart (skips portion of log processing).
- Copy is done using hardware-assisted volume level copying (ESS FlashCopy®).

- Logs need to be copied to the target and applied during restart.
- ▶ If high availability is not an issue for the source system, you might be able to afford stopping the source system prior to making the copy. This means the end users have accepted that the source SAP application will be unavailable in order to make a copy. This method is known as *offline* copy.

The definition of an offline copy implies the following:

- The source system is stopped and quiesced prior to copying. Data is consistent at the time it is copied.
- Target restart method is cold start (do not process any log records).
- DFDSS copy/rename (to disk or tape) or ESS FlashCopy is used for copying.

Cloning system configurations

When planning for HSC in a DB2 for z/OS environment, you have to consider one of the following configurations:

- ▶ Data sharing to data sharing
- ▶ Data sharing to non-data sharing
- ▶ Non-data sharing to non-data sharing
- ▶ Non-data sharing to data sharing

Cloning process

The cloning system configuration, along with the method used for HSC, has an impact on the cloning procedure itself. However, we can define six phases for the DB2 subsystem cloning process that are common to all methods:

1. Prepare for cloning.
2. Prepare the target subsystem.
3. Check the source environment and copy the source subsystem.
4. Restore the target system and restart it.
5. Alter the target subsystem.
6. Perform post cloning activities.

Additional information

For more detailed information about HSC, refer to the following documents:

- ▶ *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287
- ▶ *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678

- ▶ *SAP Web Application Server 6.20: Homogeneous and Heterogeneous System Copy*, available at:
<http://service.sap.com/instguides>

5.1.2 MCOD cloning

We use the terminology MCOD cloning to refer to the cloning of one component out of an MCOD landscape.

The specifics of MCOD cloning are as follows:

- ▶ VCATs of the other SAP components must be excluded.
- ▶ It is quite common that the VCAT used by the DB2 catalog (that is, the primary VCAT) is also used by the SAP component that was first installed in the MCOD landscape. If you choose to clone another SAP component out of the MCOD landscape, you need to exclude all the application data that is located in the primary VCAT.
- ▶ After cloning, the DB2 catalog of the target system must be cleaned up: All objects that were excluded from cloning must be dropped.

MCOD cloning is a *selective* HSC that must be followed by a cleanup of the DB2 catalog. Therefore, the cloning methods described in the HSC documentation apply, with a few modifications that take MCOD cloning into account. In the following sections, we describe an alternative technique using the Control Center.

5.1.3 Control Center support for HSC and MCOD cloning

DB2 Universal Database Control Center Version 8 provides complete database administration functionality in one easy to use tool. It is available free of charge as part of the DB2 Management Tools Package (JDB661D) for DB2 for OS/390 V6 and the DB2 Management Clients Package (JDB771D) for DB2 for OS/390 and z/OS V7.

Control Center helps you generate quickly and easily the JCL jobs¹ required to do an HSC of an entire subsystem. Using Control Center, you can perform an offline HSC using DFDSS copy/restore to DASD between two non-data sharing DB2 Version 6 or Version 7 subsystems.

This implies the following:

- ▶ The DASD for the dump data sets must be shared between the source and target systems (cloning between sysplexes is not supported).

¹ This JCL can be reused as often as you want as long as no major change occurs (see 5.3.7, “When do you have to regenerate JCL” on page 98).

- ▶ Faster ESS FlashCopy methods are not supported. Cloning of large DB2 production systems may take several hours. However, the source subsystem will be down for, at most, half of the time. Furthermore, the attention of the DBA is required for only a fraction of this time to control the results of the JCL jobs. And last but not least, no special hardware feature (for example, ESS FlashCopy) is required, as opposed to the online copy method.
- ▶ Almost every large SAP implementation uses data sharing (which is not supported).

Important: Support for data sharing and cloning to tape will be added to Control Center in an upcoming release.

However, cloning *to tape* and cloning *between data sharing systems* can be accomplished even with the current Control Center support with some modifications of the JCL and the cloning instructions. An experienced DB2 system administrator will be able to do that. In 5.3.9, “Additional considerations” on page 108, we demonstrate with a few examples how to modify the JCL generated by the Control Center in order to perform dump and restore to and from tape and to clone in a data sharing environment.

Control Center supported HSC offers a number of compelling features that make the solution very attractive over manual procedures and perfectly suited for MCOB cloning:

- ▶ Selective cloning (VCATs can be excluded).
- ▶ Application databases in the primary VCAT can be excluded.
- ▶ Catalog cleanup (Control Center provides a cleanup job that drops all objects from the target catalog that were excluded from cloning).

5.2 Cloning scenario

Our starting point is the MCOB landscape created in Chapter 3, “MCOB installation and merge using SAP tools” on page 27, as shown in Figure 5-1 on page 74.

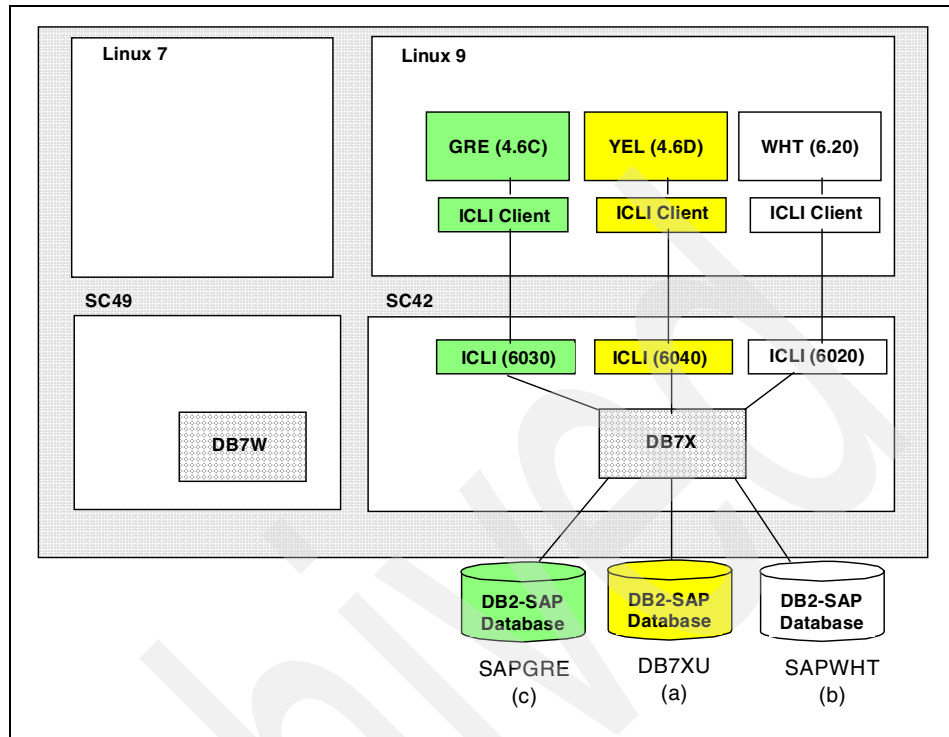


Figure 5-1 Cloning scenario: Initial configuration

That MCOD landscape is made up of three SAP systems, YEL, WHT, and GRE, running in a non-data sharing DB2 subsystem. The first system in the MCOD uses the same VCAT as the DB2 catalog (DB7XU).

The objective of our scenario is to clone GRE and create ORA on the other LPAR, where an empty non-data sharing DB2 subsystem (DB7W) has been installed, as shown in Figure 5-2 on page 75. For the purpose of our test, we use the Control Center cloning wizard to generate the JCL.

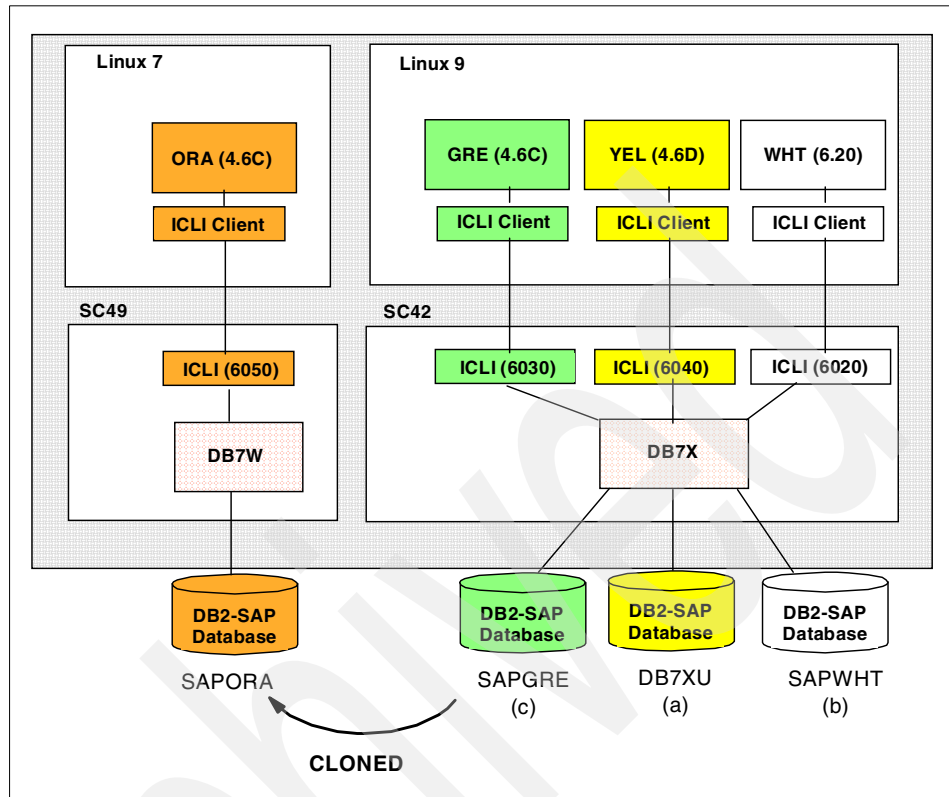


Figure 5-2 Cloning scenario: Final configuration

5.3 Using the Control Center cloning wizard

The Control Center cloning wizard helps you generate the JCL jobs required to do an HSC of an entire subsystem. The wizard will not perform the actual cloning process itself, rather it generates the JCL jobs required for this task. You must then submit the cloning jobs manually for execution. Refer to 5.3.8, “Running the generated JCL” on page 98 for executing the generated cloning JCL to actually clone the selected subsystem.

5.3.1 System requirements

The following lists the system requirements:

- ▶ The target DB2 subsystem has been created, including bootstrap data sets (BSDS), active logs, and associated RACF® authorizations, and is operational.

- ▶ The Database Administration Server (DAS) for OS/390 and z/OS V8.1 (a no-charge feature of DB2 for OS/390 and z/OS V7) is installed and active on the source and target LPARs. It is possible to clone a DB2 for OS/390 and z/OS V6 subsystem provided that the DAS is installed and configured on that LPAR.
- ▶ The 390 Enablement (JDB661D or JDB771D) is installed and activated on both the source and target subsystem, with the most recent maintenance including the PTF for PQ68659 applied to it (see PSP bucket).
- ▶ Both source and target subsystem must be cataloged in the Control Center and have the utilities bound against them.
- ▶ The source and target subsystems do *not* use data sharing.
- ▶ The source and target subsystems are running in the same sysplex. The generated JCL supports only dumping to and restoring from DASD, not magnetic tape.
- ▶ All data sets are located in system managed space (SMS).
- ▶ Sufficient DASD space must exist to hold the dump data sets and the copied DB2 objects on the target subsystem.
- ▶ The cloning of the source subsystem is performed only after a normal shutdown of the source subsystem.
- ▶ The target subsystem will be at the same version, release, and maintenance level for DB2 and its applications as the source subsystem after the cloning process is complete.
- ▶ Any additional work needed to copy applications that run against the cloned subsystem, such as QMF™ and DB2AUG, must be performed independently. All active and valid application packages are bound on the target subsystem as part of the cloning process.
- ▶ The user ID that submits the batch jobs must have an OMVS segment defined. You can verify this by issuing the following TSO command:

```
TSO LU <userid> OMVS
```
- ▶ The user ID that submits the batch jobs needs to have read access to the DAS LIBPATH directories and libraries.

5.3.2 Skill requirements

The following lists the skill requirements:

- ▶ It is assumed that you are familiar with listing and editing data sets in TSO, submitting jobs, and reviewing job output in SDSF.

- ▶ You should also understand the physical representation of a DB2 subsystem. You should be able to identify:
 - BSDS and know their use
 - Tablespace and indexspace VSAM data sets (consisting of cluster data sets and the data set naming conventions)
 - Active logs data sets
 - Understand where the data sets corresponding to the DB2 directory, DB2 catalog, and the DB2 work file database reside for a subsystem
- ▶ You should also have a good understanding of DFSMSdss™ storage administration, in particular the use of the DUMP and RESTORE commands.
- ▶ Part of the process requires you to compile a new ZPARM and DSNHDECP member for the target subsystem. Please make sure you have the JCL required at hand. The job DSNTIJUZ, which was generated during installation, generates the DSNZPARM (or the job whose name you specified for PARAMETER MODULE on the installation panel DSNTIPO) and the DSNHDECP. For detailed information, see the *DB2 Universal Database for OS/390 and z/OS V7: Installation Guide*, GC26-9936.
- ▶ Make sure you have access to the DB2 and z/OS or OS/390 online documentation in case you run into unexpected system problems.

Attention: Subsystem cloning is *not trivial* and data loss may occur if you do not follow the JCL instructions closely. Make sure you plan for *enough time*, have your system programmer available for questions, and if possible, try cloning a non-critical test system first.

5.3.3 Prepare to create a cloning session

Before you can create a cloning session using the Control Center cloning wizard, you need to allocate two libraries: one for JCL, the other for job cards. They must be large enough and have enough directory blocks, as shown in Example 5-1.

Example 5-1 Allocate a JCL library

```

Data Set Name . . . : SAPRES1.CLONE.JCL.CNTL

Management class . . . . . (Blank for default management class)
Storage class . . . . . (Blank for default storage class)
Volume serial . . . . . (Blank for system default volume) **
Device type . . . . . (Generic unit or device address) **
Data class . . . . . (Blank for default data class)
Space units . . . . . TRKS (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
  
```

Average record unit	(M, K, or U)
Primary quantity . . 10	(In above units)
Secondary quantity 5	(In above units)
Directory blocks . . 50	(Zero for sequential data set) *
Record format FB	
Record length 80	
Block size 27920	
Data set name type : PDS	(LIBRARY, HFS, PDS, or blank) *

Before you update JCL or write new JCL into an existing library, make sure you compress it. Otherwise, an error may occur while saving the JCL.

Both the source and target subsystem must be up and running in order to generate cloning JCL, because the wizard will query their catalogs for VCAT and space information.

It is recommended to run STOSPACE on all source subsystem VCATs in preparation for your cloning session. Having accurate space allocation information will help you choosing a primary and secondary space for your dump data sets. To run STOSPACE in Control Center, select the Storage Groups folder, select all storage groups, and select **STOSPACE** from the pop-up menu.

5.3.4 Running the Control Center cloning wizard

In the following section, we describe how to generate the JCL required to clone a DB2 subsystem using the Control Center wizard.

Create a new session

Connect to the source subsystem and select **Clone** -> **Create Session**, as shown in Figure 5-3, from the source subsystem pop-up menu.

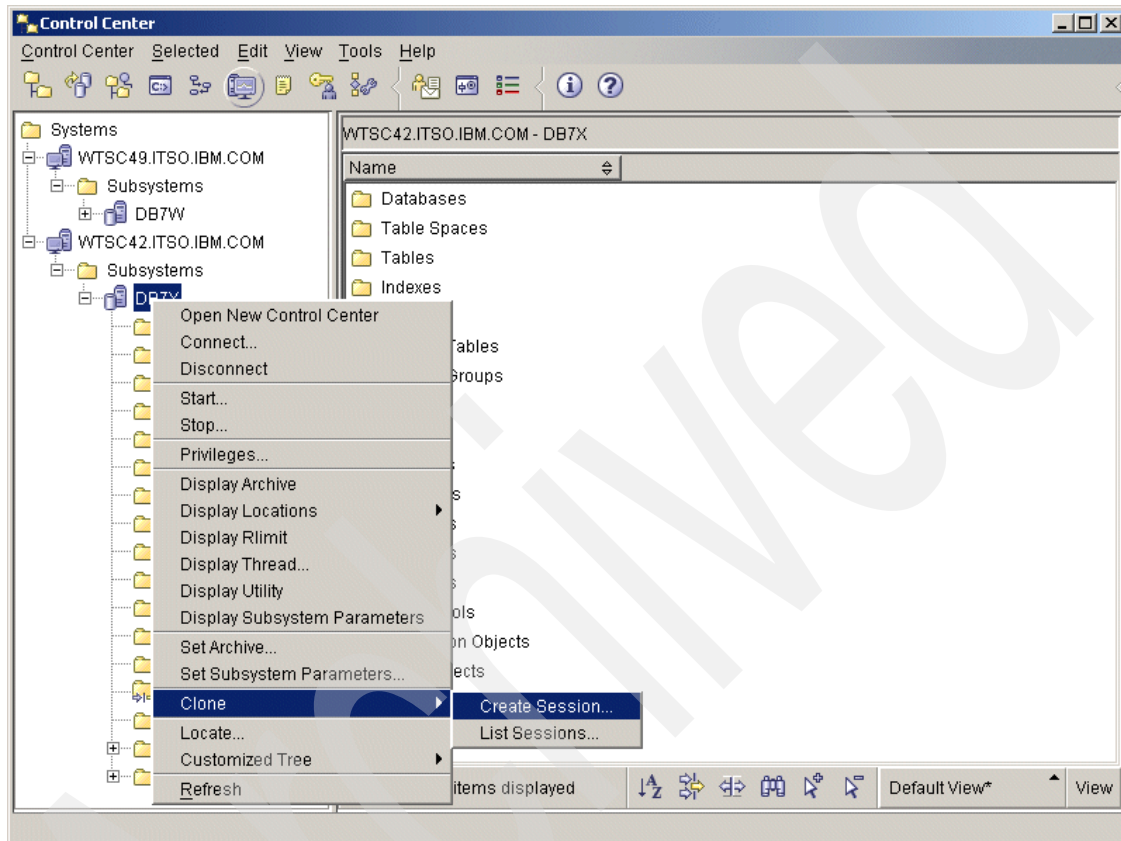


Figure 5-3 Create a new session in the cloning wizard

Introduction

This panel, as shown in Figure 5-4, shows some introductory comments about the Create Cloning Session or Edit Cloning Session wizard. Continue to the next panel and begin to enter the information required for generating the subsystem cloning JCL.

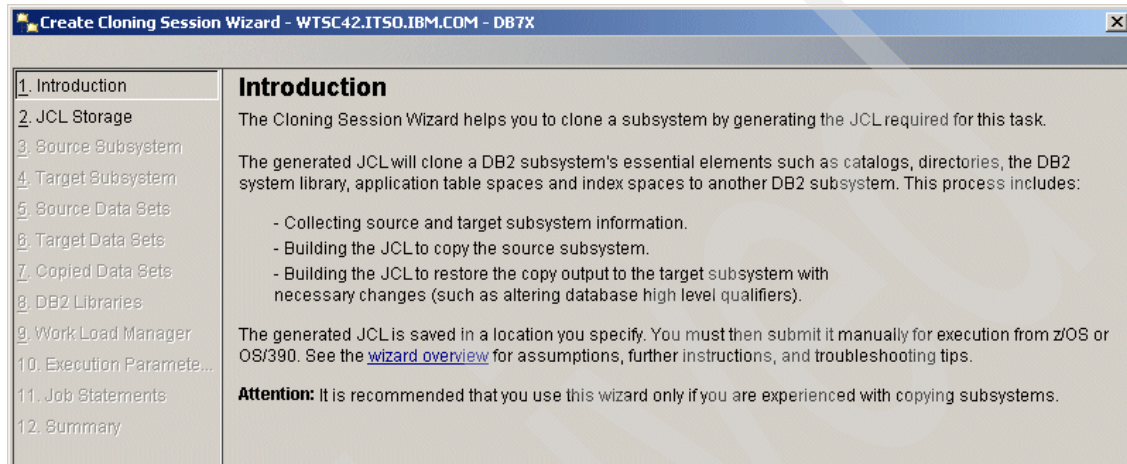


Figure 5-4 Create cloning session wizard: Introduction

JCL storage

On this panel, as shown in Figure 5-5, you need to specify the partitioned data set library where the cloning JCL jobs generated by the wizard will be stored. Existing members will be displayed in the members list. It is strongly recommended to manually empty the library so that you don't submit jobs from old sessions by mistake (for example, jobs that would delete a target VCAT that no longer exists on the target subsystem).

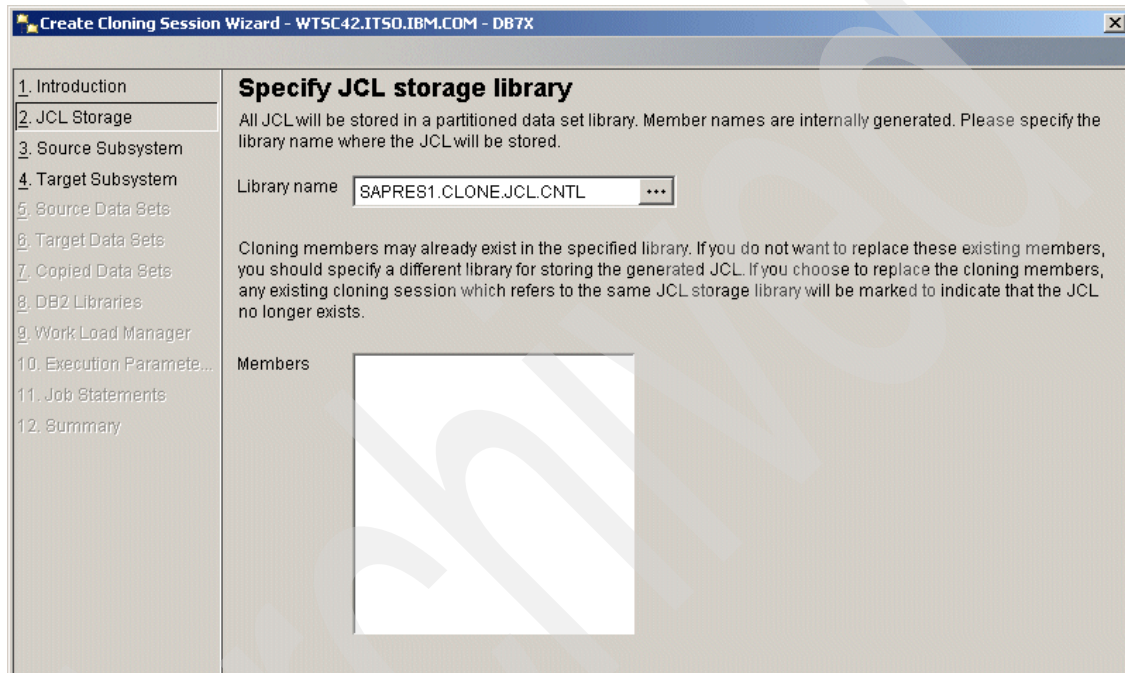


Figure 5-5 Create cloning session wizard: JCL storage

Source subsystem

On this panel, as shown in Figure 5-6, you enter the BSDS clusters for the source subsystem you want to clone. Specify both BSDSs. If your system only uses one BSDS, specify BSDS1 only.

1. Introduction
2. JCL Storage
3. Source Subsystem
4. Target Subsystem
5. Source Data Sets
6. Target Data Sets
7. Copied Data Sets
8. DB2 Libraries
9. Work Load Manager
10. Execution Paramete...
11. Job Statements
12. Summary

Specify source subsystem bootstrap data set

The following shows the source system and subsystem to be copied during the cloning process. Please note that the cloning function is supported only for source and target systems that run on the same system complex (SYSPLEX).

System name

Subsystem name

Specify the bootstrap data set (BSDS). If the source subsystem is configured with dual bootstrap data sets, you need to specify both data sets.

BSDS 1

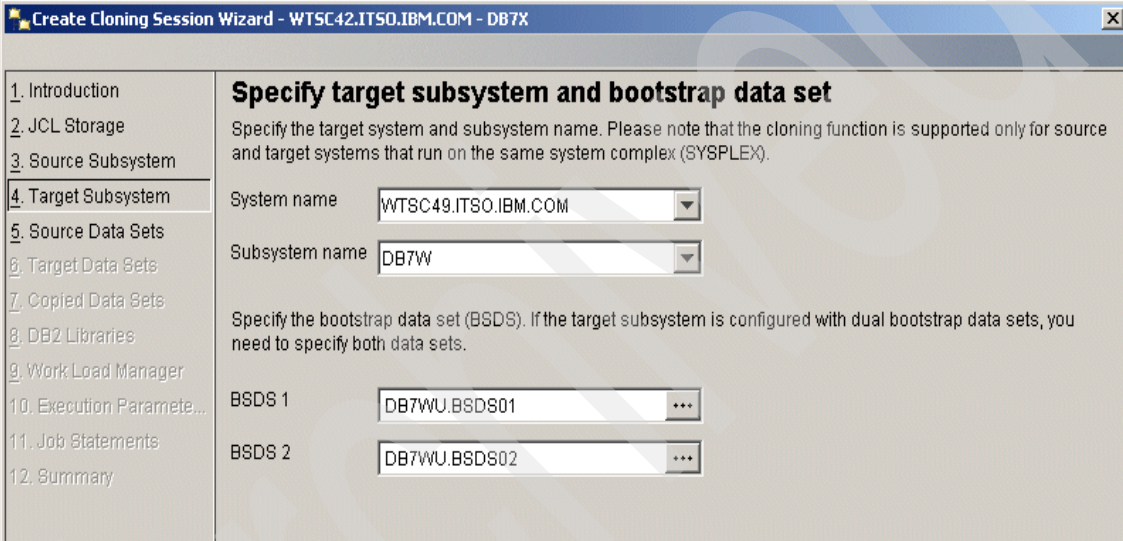
BSDS 2

Figure 5-6 Create cloning session wizard: Source subsystem

Target subsystem

On this panel, as shown in Figure 5-7, you select the target subsystem for the cloning process. You must also specify the BSDS clusters of the target subsystem.

You can specify any DB2 subsystem as the target subsystem that is cataloged on the client. If you specify a target subsystem in an LPAR with which the source subsystem does not share DASD or in a different sysplex, you have to modify the generated JCL to dump the source subsystem to tape. Control Center only checks that the source and target subsystem are different and on the same DB2 version.



The screenshot shows a window titled "Create Cloning Session Wizard - WTSC42.ITSO.IBM.COM - DB7X". The left sidebar contains a list of steps: 1. Introduction, 2. JCL Storage, 3. Source Subsystem, 4. Target Subsystem (highlighted), 5. Source Data Sets, 6. Target Data Sets, 7. Copied Data Sets, 8. DB2 Libraries, 9. Work Load Manager, 10. Execution Paramete..., 11. Job Statements, and 12. Summary.

The main panel is titled "Specify target subsystem and bootstrap data set". It contains the following text and fields:

Specify the target system and subsystem name. Please note that the cloning function is supported only for source and target systems that run on the same system complex (SYSPLEX).

System name: WTSC49.ITSO.IBM.COM

Subsystem name: DB7W

Specify the bootstrap data set (BSDS). If the target subsystem is configured with dual bootstrap data sets, you need to specify both data sets.

BSDS 1: DB7WU.BSDS01

BSDS 2: DB7WU.BSDS02

Figure 5-7 Create cloning session wizard: Target subsystem

Source data sets

On this panel, as shown in Figure 5-8, you specify which VCATs of the source subsystem you want to copy. The primary VCAT (the ICF catalog where your DB2 directory and catalog data sets are cataloged) always has to be copied. You can deselect any secondary VCAT to exclude it from copying. However, if a database consists of DB2 objects in an included and excluded VCAT, the excluded objects will be dropped from the database on the target.

Note: In our scenario, we want to clone the SAP system with VCAT SAPGRE out of the MCODE landscape. The primary VCAT DB7XU has to be copied (it cannot be deselected). Unfortunately, DB7XU is also the VCAT of the SAP system that was first installed in the MCODE landscape. To enable selective primary VCAT cloning, we have to manually update the XMAP member, as described in 5.3.6, “XMAP member and MCODE cloning” on page 96.

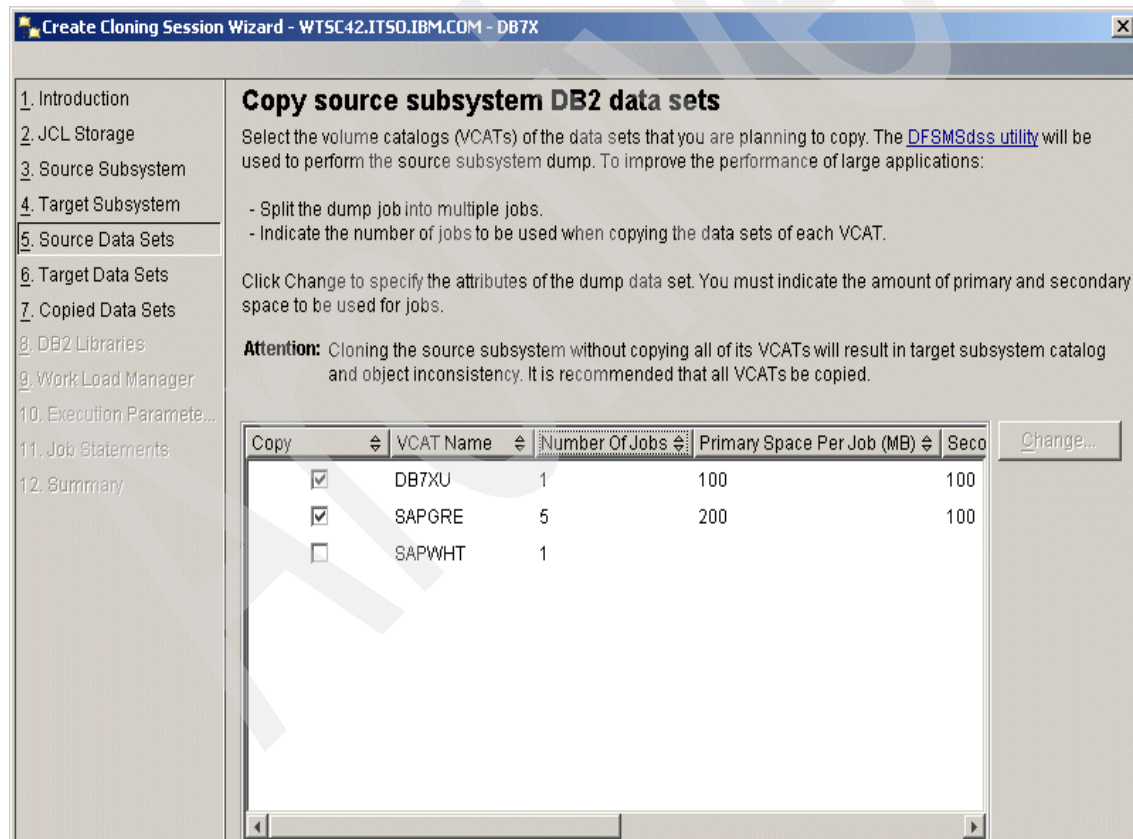


Figure 5-8 Create cloning session wizard: Source data sets

For each VCAT, you can specify the number of jobs (which is also the number of dump data sets for that VCAT). For a large application, such as SAP, 5 to 8 jobs is a good choice. It allows you to run the dump jobs in parallel and shortens your source system down time. For an almost empty subsystem with just a test database, you can also specify 1.

If you have run STOSPACE, the Change dialog box tells you the entire space requirements of that VCAT. Choose primary, secondary spaces, and unitcount accordingly. For example, for a VCAT with 100 MB, you may choose five jobs, leaving roughly 20 MB for each dump data sets. Then, you can use 10 MB as the primary space and 5 MB as the secondary space and a unit count of 10.

The dump data sets are allocated with RLSE so that allocated but unused space will be released.

Make sure that the system programmer is aware of the naming conventions for the cloning data sets. All work data sets allocated by the batch programs will have the following prefix: SVCAT.CLONE, where SVCAT is the high-level qualifier (HLQ) of the primary VCAT of the target DB2 subsystem.

You should double check that there are no ACS routines set up that would force, for example, VSAM only allocation under that qualifier, or anything that may interfere with the cloning process. It is a good idea to either set up an ACS routine for SVCAT.CLONE.** to direct the cloning workfiles to a set of volumes or to use a data storage class for the same purpose. Enough space should be provided for the cloning work and dump data sets.

Target data sets

On this panel, as shown in Figure 5-9, you specify the DB2 volume catalogs (VCATs) that will be deleted on the target subsystem in preparation for restoring the source subsystem data sets to the target subsystem.

For larger systems (such as SAP), it is also recommended to split the delete action into separate parallel jobs. You have to delete the DB2 data sets under the primary target VCAT. You may keep other target VCATs, but then you should not specify them as target VCATs on the following panel. It is recommended to delete the entire target subsystem, because data sets that are excluded from deleting will not be in the target catalog anymore after the cloning process.

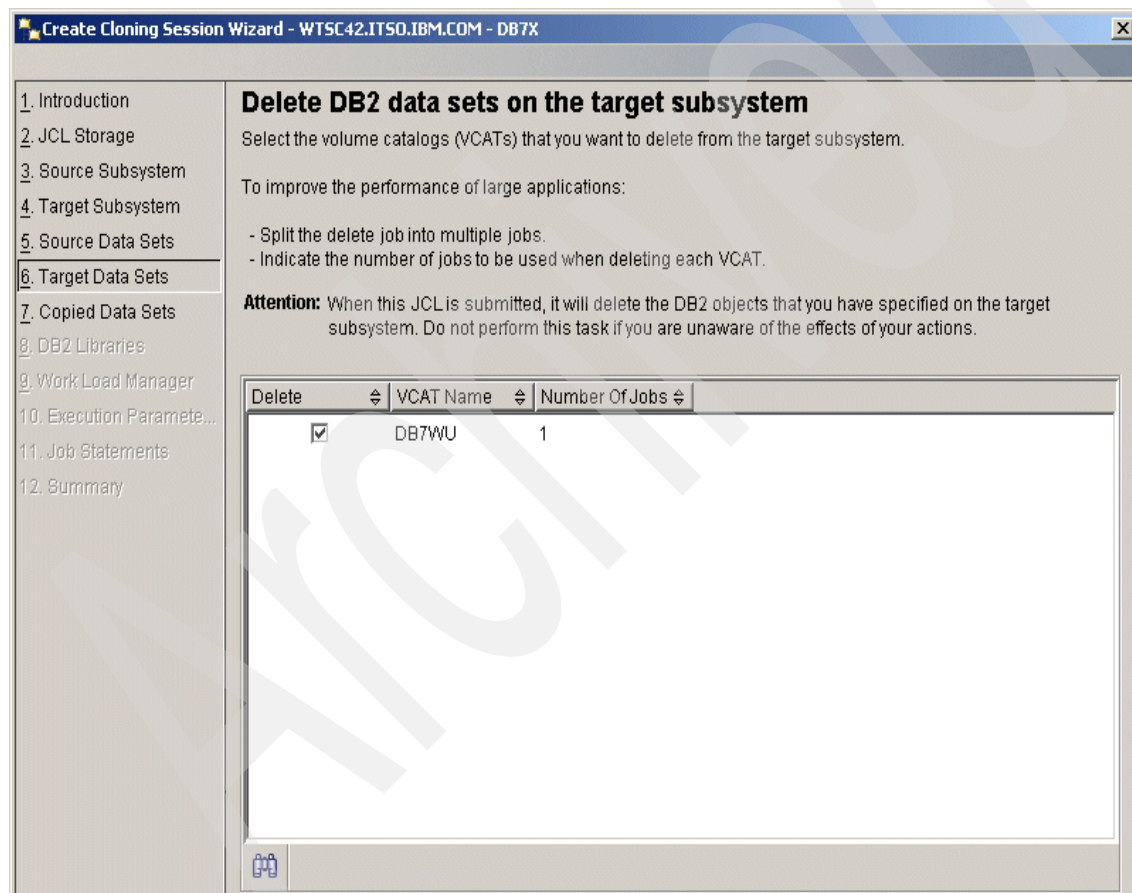


Figure 5-9 Create cloning session wizard: Target data sets

Copied data sets

On this panel, as shown in Figure 5-10 on page 88, you map source VCATs to target VCATs. Typically, you will map the primary source VCAT to the primary target VCAT by selecting it from the pull-down list. Only VCATs that exist on the target are on the pull-down list. For new VCATs, you have to type them.

Note: It is your responsibility to define any non-existing VCATs before submitting the jobs.

You can optionally specify a storage or management class that will control the placement of your restored target DB2 data sets. Typically, this will be controlled by an ACS routine for that HLQ. If you enter a target VCAT name that is not on the list, go into 3.4 and check that there are no data sets under this target VCAT name. Otherwise, you could run into problems restoring the dump data sets because of duplicate data set names.

In our scenario, the target subsystem has only one VCAT (DB7W). We have chosen to introduce a new VCAT (SAPORA) so that the DB2 catalog and the SAP application data will not share the same VCAT anymore. To fully comply with the recommendations we give in 2.1.1, “Keep things as separate as possible” on page 16, we define the associated alias in a separate ICF catalog. We also update the SMS definitions (new SMS storage class and storage group) to ensure that the application data sets reside on a separate pool of volumes.

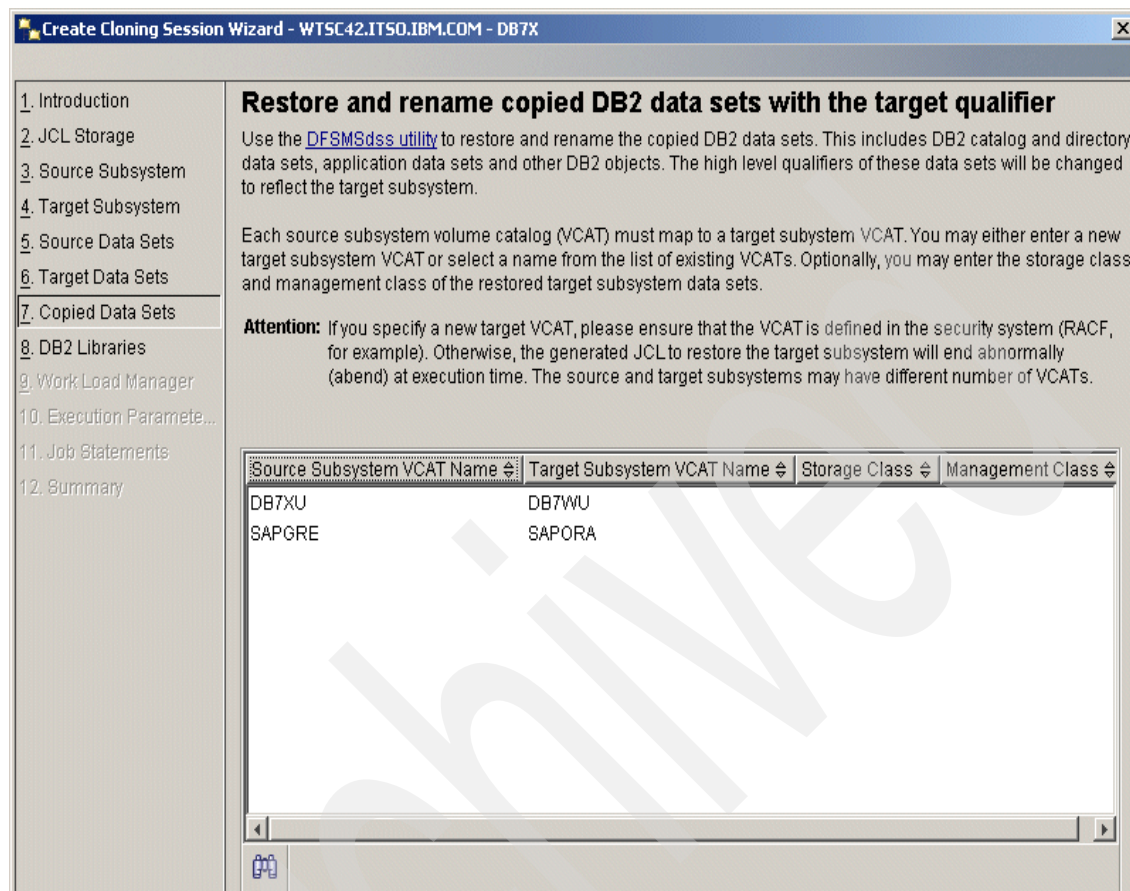


Figure 5-10 Create cloning session wizard: Copied data sets

DB2 libraries

On this panel, as shown in Figure 5-11, you specify all subsystem-specific DB2 libraries that should be copied from source to target, including SDSNEXIT (where the ZPARAMs, EDITPROCs, FIELDPROCs, and compression exits are stored), RUNLIB.LOAD (where the sample programs including DSNTIAD and DSNHDECP are stored), and DBRMLIB.DATA (where the sample program DBRMs are stored). The SDSNLOAD library may also be listed (unless it is shared and under a different HLQ). You enter the source and target subsystem HLQs, click **Show Libraries**, and all libraries under the source HLQ will show up in the list selected for copy by default.

Note: In our environment, SDSNEXIT is under HLQ DB7X7, whereas RUNLIB.LOAD and DBRMLIB.DATA are under DB7XU. The DB2 libraries panel can handle only one HLQ, so we select DB7XU, and we will manually add SDSNEXIT to the generated JCL, as described in “Step 3: Check source environment and copy subsystem” on page 101.

Copy and rename DB2 system libraries

Part of the cloning process involves copying the DB2 subsystem libraries to the target system. To accomplish this, the high level qualifier of the DB2 subsystem libraries has to be renamed with a target subsystem high level qualifier. The number of levels in the high qualifiers must be the same.

Specify the source subsystem high level qualifier. To list the data sets associated with a specified high level qualifier, click Show Libraries beside the table below.

Source subsystem high level qualifier

Specify the target subsystem high level qualifier. The selected source subsystem libraries will be copied and renamed with the target high level qualifier.

Target subsystem high level qualifier

Select the source subsystem libraries to be copied and renamed.

Copy	Source Subsystem Library
<input checked="" type="checkbox"/>	DB7XU.DBRMLIB.DATA
<input type="checkbox"/>	DB7XU.NEW.SDSNCLST
<input type="checkbox"/>	DB7XU.NEW.SDSNSAMP
<input type="checkbox"/>	DB7XU.PROCLIB
<input checked="" type="checkbox"/>	DB7XU.RUNLIB.LOAD
<input checked="" type="checkbox"/>	DB7XU.SRCLIB.DATA

Figure 5-11 Create cloning session wizard: DB2 libraries

Work Load Manager

On this panel, as shown in Figure 5-12, you map the Work Load Manager (WLM) application environment for every user-defined function and stored procedure running in a WLM-managed address space. Use copy and paste to speed up this tedious process. It is common to have separated WLM application environments per subsystem for stored procedures. Make sure you map the ones with compatible NUMTCBs, for example, DSNUTILS must have NUMTCB=1 and DSNACCMO must have NUMTCB=100.

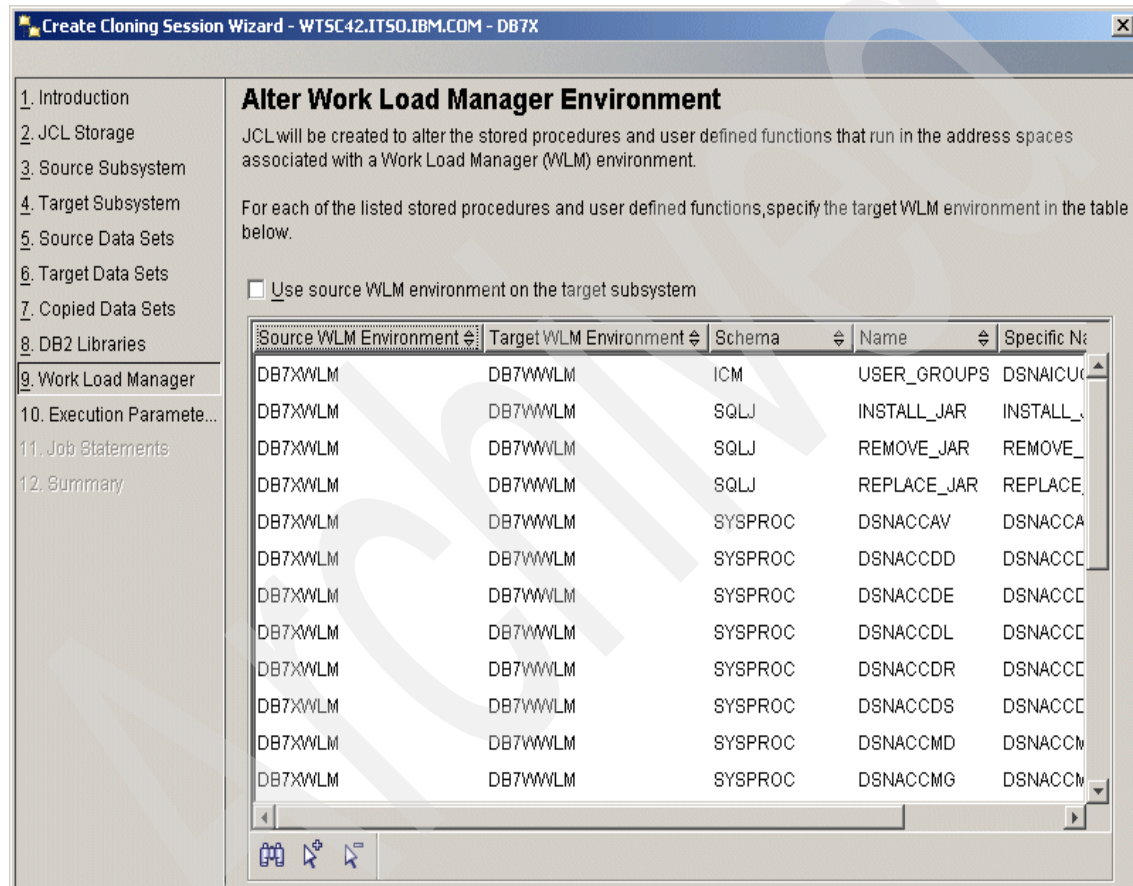


Figure 5-12 Create cloning session wizard: Work Load Manager

Execution parameters

On this panel, as shown in Figure 5-13, you must enter the system libraries and plan names for the source and target subsystems.

The cloning JCL executes Control Center batch programs DSNACCC1 to DSNACCC8 (which are installed in SDSNLOAD), DSNTIAD (which is installed in RUNLIB.LOAD), DSNTDP2 (which is installed in RUNLIB.LOAD), DFSMDSS utilities, such as IDCAMS and ADRDSSU that are assumed to be in the LINKLIST, and DB2 offline utilities that are installed in SDSNLOAD. What you enter here will be used to build JOBLIB and STEPLIB statements for the cloning JCL. If you have a DB2 installation that does not follow these conventions, you have to manually edit the JOBLIB and STEPLIB statements in the generated JCL.

10. Execution Parameters...

Specify JCL execution parameters

Specify the system libraries and plan names used by the source and target subsystems. The specified libraries and plan names are used as the default load libraries and plan names to generate JCL.

Source subsystem

SDSNLOAD library: DB7X7.SDSNLOAD

RUNLIB.LOAD library: DB7XU.RUNLIB.LOAD

DSNTIAD plan name: DSNTIA71

DSNTEP2 plan name: DSNTEP71

Target subsystem

SDSNLOAD library: DB7W7.SDSNLOAD

RUNLIB.LOAD library: DB7WU.RUNLIB.LOAD

DSNTIAD plan name: DSNTIA71

DSNTEP2 plan name: DSNTEP71

Figure 5-13 Create cloning session wizard: Execution parameters

Job statements

On this panel, as shown in Figure 5-14, you specify a job card statement for each job that will be created by the generated cloning JCL. These job cards will be saved to the JOBCARD library that you created before you started the new cloning session.

You do not have to edit the job card for each job. Edit the job card for the very first job PADELET1 so that it is a valid job card in your environment, as shown in Example 5-2. Select **Use this jobcard for all jobs**, and select **OK**.

Example 5-2 Create a job card

```
//PADELET1 JOB (999,POK), 'SAPRES1', CLASS=A, MSGLEVEL=(1,1),  
//          MSGCLASS=T, NOTIFY=&SYSUID, REGION=OM
```

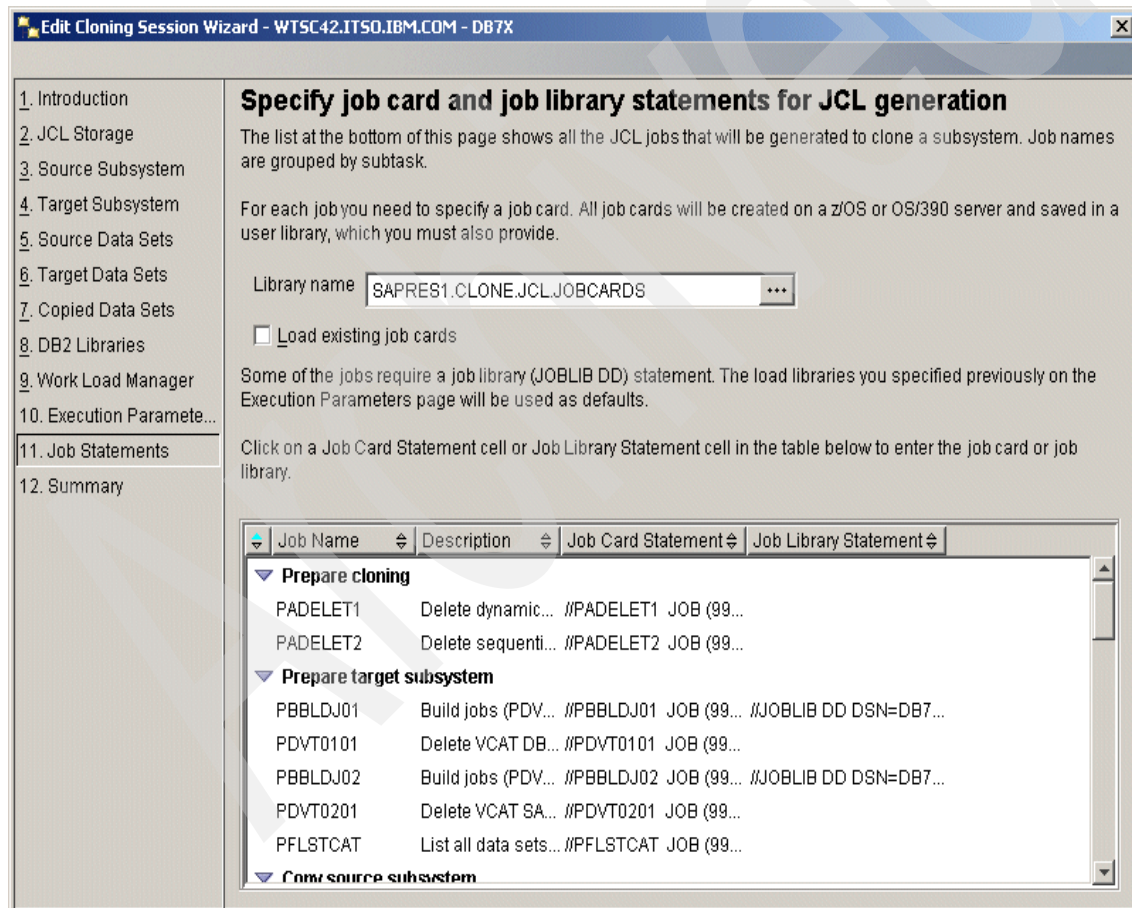


Figure 5-14 Create cloning session wizard: Job statements

Summary

After clicking **Next** on the job statements panel, the JCL jobs will be generated so that you can review them before saving them. Generating the jobs may take a few minutes. You will receive a message saying, “All JCL jobs were generated successfully” as shown in Figure 5-15.

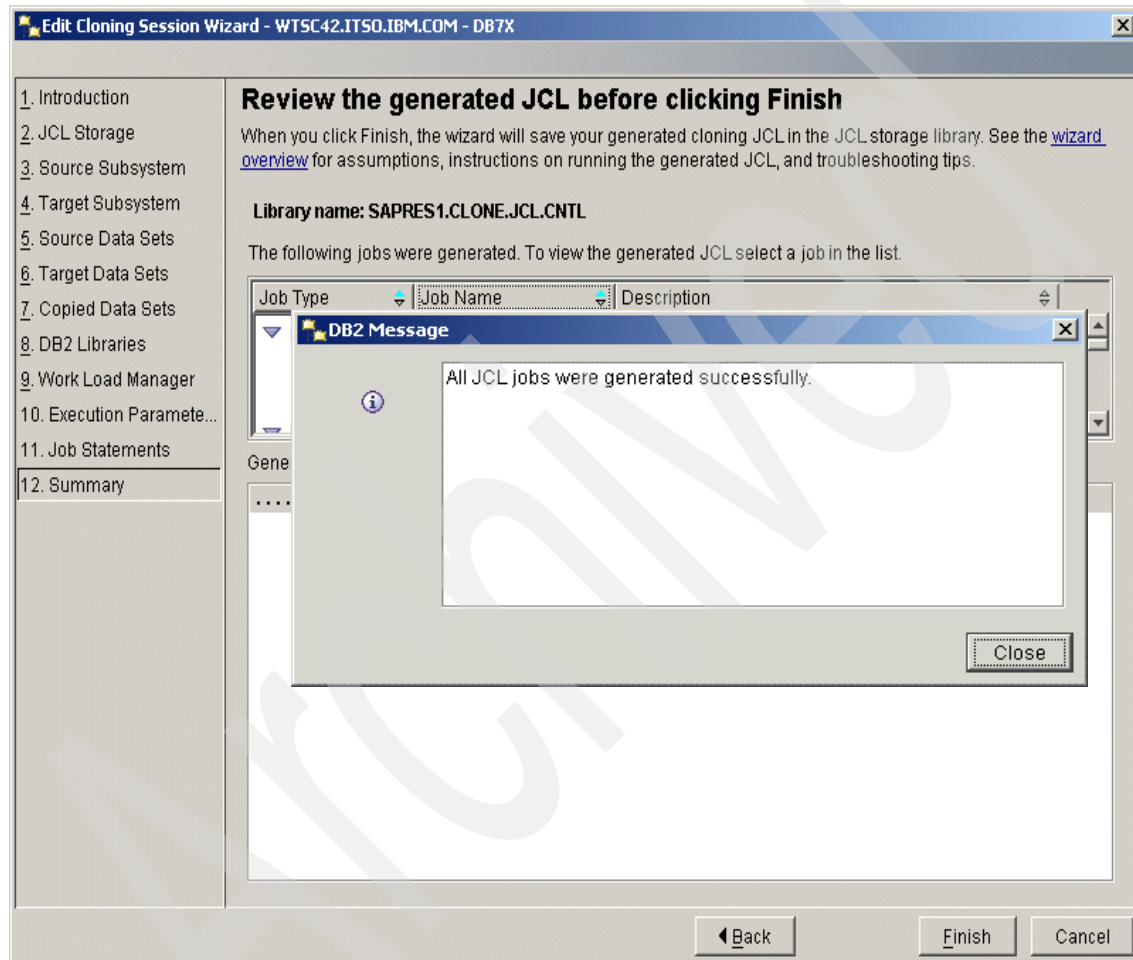


Figure 5-15 Create cloning session wizard: Summary (1/2)

Select **Finish** to actually save the JCL jobs and job cards to the specified libraries and to save the cloning session. If you cancel the dialog here, you will have to start all over again. Saving the job cards to the library will take a few minutes. Make sure that before you select **Finish**, the two libraries (for JCL and job cards) are not allocated (for example, you are not browsing them in a TSO session). Otherwise, the JCL save will fail.

Your JCL has been generated successfully only if you see a message like the one shown in Figure 5-16. Do not use the JCL if you do not get this message. The JCL may be incomplete.

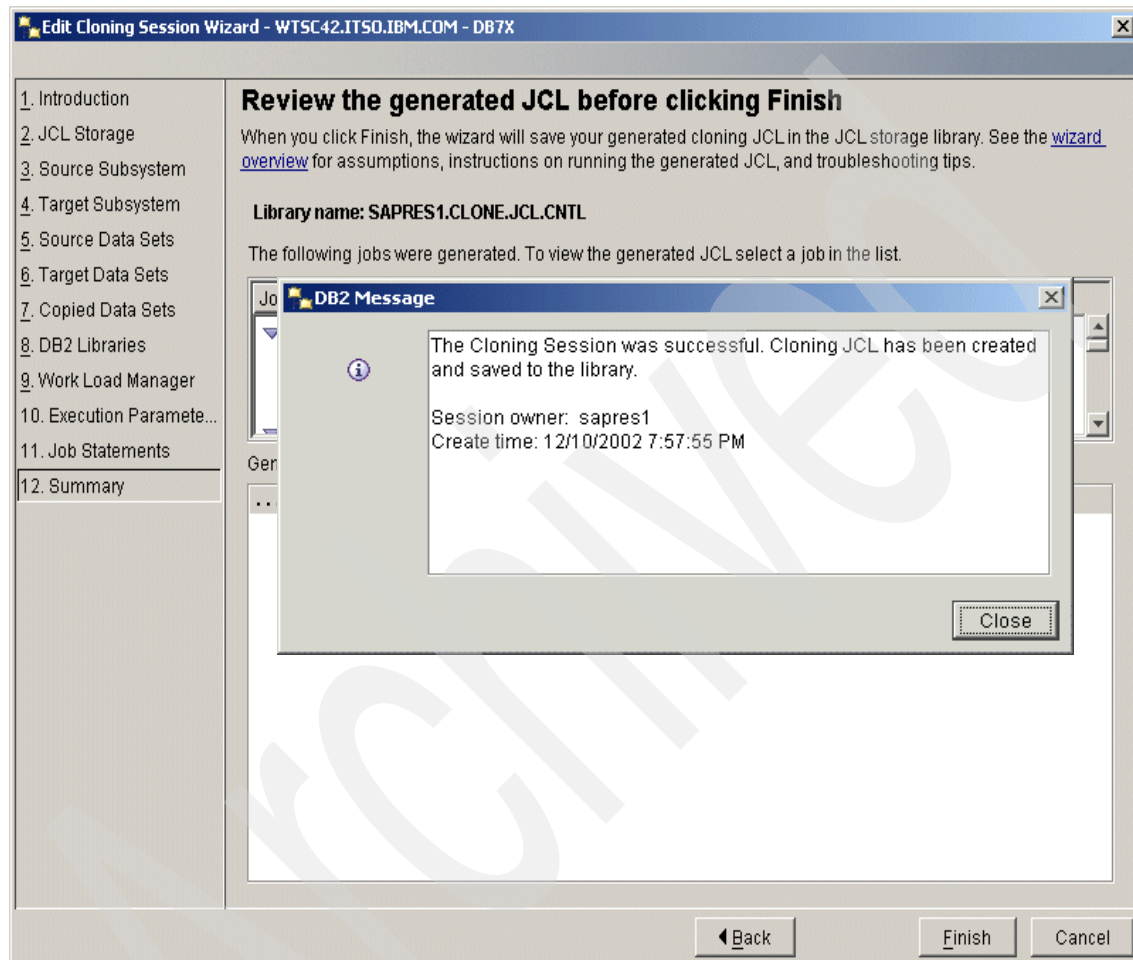


Figure 5-16 Create cloning session wizard: Summary (2/2)

5.3.6 XMAP member and MCODE cloning

The XMAP member contains the VCAT and WLM mapping information for this cloning session. The format is simple. There is always the number of lines followed by the actual information: first, the VCAT mapping information, followed by the routine mapping information. When a VCAT is excluded from copying, you will see the token `_EXCLUDED_` instead of the target VCAT.

Attention: It is usually not recommended to edit the XMAP file. However, MCODE cloning may require that you manually add some lines to the file. This must be done with extra care and be limited to what is described in this book.

MCODE cloning may require that not the entire primary VCAT of the source subsystem is copied, but only the system databases (databases that start with DSN), the Control Center database (CC390), and one or more SAP installations in secondary VCATs.

This option is not yet supported in the Control Center cloning wizard GUI but it is supported by the back-end programs if you have all maintenance levels including PQ68650 installed. To enable selective primary VCAT cloning, you have to manually add the following two lines in bold at the end of the XMAP file, as shown in Figure 5-18 on page 97.

2
MYDB1
MYDB2

Selectively cloning the primary VCAT will reduce the space requirements for the dump and target DB2 data sets, as well as shorten the cloning time. The cleanup job will make sure that all objects that were excluded from cloning will be dropped from the target catalog.

5.3.7 When do you have to regenerate JCL

The following describes when you have to regenerate JCL:

- ▶ When the number of VCATs changes on either your source subsystem or your target subsystem (for example, you create a new storage group with a new VCAT). Creating new storage groups with existing VCATs (but different volume counts or volumes) is fine.
- ▶ When you create a stored procedure or a user-defined function.
- ▶ When your WLM application environment names for stored procedures or user-defined functions change.
- ▶ When the space requirements for the dump data sets change significantly.

It is important that once you start generating your cloning JCL, nothing of the above happens. It will lead to your cloning JCL not working properly.

5.3.8 Running the generated JCL

This section describes running the generated JCL.

Attention: Do not proceed if you do not have the required skills to prevent potentially irrecoverable loss of data.

Important rules for submitting the cloning JCL

The following lists important rules for submitting the cloning JCL:

- ▶ Read the JCL instructions completely before your first cloning session. If you do not understand something, do not proceed and research the topics. Once you start cloning, you cannot undo certain tasks and you have to be able to deal with error situations that are very specific to your environment.

- ▶ The job names have been chosen so that their alphabetical order reflects the order in which they have to be submitted. Make sure you submit all the jobs in the right sequence and not in parallel (unless stated otherwise). It is important that you precisely follow the cloning instructions.
- ▶ Do not ignore non-zero return codes, unless it is specifically documented that they are acceptable. For troubleshooting, refer to the online help of the Control Center cloning wizard.

Naming convention

When examples are given, the naming convention shown in Table 5-1 is used.

Table 5-1 Naming convention

Abbreviation	Our value	Description
SSID	DB7X	Subsystem name for the source DB2 subsystem
TSID	DB7W	Subsystem name for the target DB2 subsystem
SVCAT	DB7XU	HLQ of the catalog data sets for the source DB2 subsystem
TVCAT	DB7WU	HLQ of the catalog data sets for the target DB2 subsystem
SSAPSID	GRE	Source SAP system name
TSAPSID	ORA	Target SAP system name
SCHEMA	SAPGRE	Creator/schema of the SAP objects

Step 1: Prepare for cloning

Depending on your applications, you may have to perform preparation activities. You may also have to delete any work files that were dynamically created on the target subsystem during a previous cloning session. Proceed as follows:

1. Refer to Chapter 1, “Introduction,” in *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678, or Chapter 1, “Planning” and Chapter 2, “Preparations,” in *SAP Web Application Server 6.20: Homogeneous and Heterogeneous System Copy* for planning and preparation activities specific to SAP applications.
2. The cloning process creates temporary work files used for cloning under SVCAT.CLONE.* and temporary JCL members with the prefix PDVTjinn, SHVTjinn, and TEVTjinn. If you have previously successfully completed a cloning section, submit the following two jobs to clean up these temporary data sets and members. You can also delete them manually.

Skip these two jobs if you are cloning for the first time. Otherwise, you will receive a JCL error on PADELET1 and an RC=4 on PADELET2.

- a. Submit PADELET1 to delete JCL library members called PDVTjnn, SHVTjnn, and TEVTjnn that have been dynamically created by the PBBLDJnn and SFBLDJnn jobs in any previous cloning session. Verify that RC=0.
- b. Submit PADELET2 to delete work files under the TVCAT.CLONE high-level qualifier that were created in any previous cloning session. Verify that RC=0.

Step 2: Prepare the target subsystem

During this step, you delete all DB indexspace and tablespace data sets associated with the target subsystem:

1. Stop all target subsystem applications (this means Application Servers including Central Instance and ICLI servers). Verify that all target subsystem applications are stopped.
2. Issue DISPLAY THREAD(*) on the target subsystem. Verify that no connections are returned.
3. Issue STOP DB2 to stop the target subsystem. Verify that the target subsystem is stopped by checking that the following message is printed to the console:

```
DSN3100I ssid DSN3ECOX - SUBSYSTEM ssid READY FOR START COMMAND
```
4. Submit PBBLDJjj jobs² to dynamically create the delete jobs PDVTjnn. Verify that RC=0 or RC=4. Verify that any VCAT names you have typed as target VCATs (and not selected from the list) do not have any data sets associated with them. It is your responsibility to delete any data sets under a target VCAT name you typed on the Copied Data Sets panel³. Refresh your member list to see the generated jobs.

² You have one PBBLDJjj job for every target VCAT to be deleted. PBBLDJjj uses a DLL in HFS, so you need to make sure that the user ID under which you submit the job has an OMVS segment defined and read access to all the DLLs that are in the libpath of the user under which the DAS is running (for example, dasuser), as well as read access to /var/db2/global.reg. If one of these prerequisites is not met, the job will abend. If you did not create links to the DAS DLLs in /usr/lib during the DAS activation process, you have to replace any occurrence of /usr/lib in the JCL job with the direct path (for example, /u/dasuser/das/lib).

³ As mentioned in “Copied data sets” on page 87, it is also your responsibility to define any non-existing VCATs before submitting the jobs.

5. Submit PDVTjnn jobs to delete DB2 indexspace and tablespace data sets associated with the target subsystem. These jobs can be submitted in parallel. Verify that RC=0. If RC=8, and some data sets failed serialization, make sure that the target subsystem is completely stopped and resubmit the job. Verify that the resubmitted job returns RC=0 or RC=4. Submit them as often as required until you get RC=0 or RC=4 (typically, because all data sets have been deleted).
6. Submit PFLSTCAT to list all data sets under the target subsystem VCAT name and all VCAT names that were selected for deletion. Verify that RC=0 or RC=4 and that no DB2 indexspace or tablespace data sets are listed (with DSNDBC as their second-level qualifier). The only VSAM cluster data sets that should remain under the target VCAT name are the BSDS and the log data sets.

If you accidentally deleted a set of BSDS on the target system, do not proceed. You have to recover them to proceed.

Step 3: Check source environment and copy subsystem

During this step, you create the commands that are required to alter the target subsystem after the cloning process and dump all the DB2 indexspace and tablespace data sets, as well as the user-specified library data sets associated with the source subsystem. Then, you take the dump after a normal shutdown of the source subsystem. Proceed as follows:

1. Stop all source subsystem applications (this means all Application Servers including Central Instance and all ICLL servers). Verify that all source subsystem applications are stopped.
2. Submit SAADSPY to display activity on the source subsystem. Verify that there is no outstanding activity on the source subsystem. Continue only if the only thread displayed in the SAADSPY report is SAADSPY, and no utilities or databases with spaces in restricted states were returned. Resolve stopped utilities and restricted space states before proceeding.
3. Submit SAALTIDX, SAALTQRY, and SAALTUSR in parallel to create commands that are required to alter and clean up the target subsystem after the cloning process. Verify SAALTIDX RC=0 or RC=4. If SAALTIDX returns RC=4, the job WCALTUIX that is part of Step 5 must not be run. Verify that SAALTQRY STEP01 returns RC=0 or RC=4. Verify that SAALTQRY STEP02 returns RC=0. Verify that SAALTUSR returns RC=0 or RC=4. If SAALTUSR returns RC=4, the job WLALTUSR that is part of Step 5 must not be run.

Additional customizing information: The last parameter of the program DSNACCC1 that is run in SAALTQRY STEP01, which is passed on the last line of SYSIN, is MMCGRTS (mimic grants). By default, this is set to N. This parameter determines under which user ID storage groups are created. If MMCGRTS is N (default), all storage groups are created under the user ID of

the system administrator who performs the cloning. If MMCGRTS is Y, the SQL ID will be set to the actual creator of the storage group on the source subsystem, before the storage group is created on the target subsystem. It is recommended to keep the default setting, because the original creator of a storage group on the source subsystem may not be set up as a user or have the required authority on the target system.

Tip: All the DB2 objects, including storage groups, that belong to an SAP component must have the same creator. Therefore, we recommend that MMCGRTS is set to Y when cloning an SAP system. This will create SET CURRENT SQLID statements in the data set TVCAT.CLONE.SAALTQRY.OUTDB26. It is your responsibility to check that the original creator of a storage group on the source subsystem is set up as a user and has the required authority on the target system. In particular, a group ID <SCHEMA> (for example, SAPGRE in our scenario) must exist on the target system.

If any of the SAALTIDX, SAALTQRY, or SAALTUSR jobs abends, or there is a problem (for example, DBRMs not bound), make sure you manually delete the corresponding cloning data sets before you resubmit them. The third level is always the job name. For example: TVCAT.CLONE.SAALTQRY.**

4. Issue DISPLAY THREAD(*). Verify that no connections are returned. If connections are found, manually delete the data sets created by the SAALTIDX, SAALTQRY, and SAALTUSR jobs. Make sure all activity ceases and start Step 3: Check source environment and copy subsystem again.
5. Issue STOP DB2 MODE(QUIESCE) to stop the source subsystem in a consistent mode. Verify that the source subsystem is stopped by checking that the following message is printed to the console:

```
DSN3100I ssid DSN3ECOX - SUBSYSTEM ssid READY FOR START COMMAND
```

6. Submit SDPRTLOG to:
 - Run the Print Log Map utility (DSNJU004) against the quiesced BSDS of the source DB2 subsystem to identify the last checkpoint RBA (and save it for later use in the cloning process).
 - Ensure that the source DB2 subsystem was properly quiesced using the DSN1LOGP utility with the options SUMMARY(YES) and the STARTRBA and ENDRBA of the last DB2 checkpoint taken from the output of the Print Log Map utility from the previous step.

Verify that RC=0. If SDPRTLOG returns RC=12, you have submitted the job too early, and the BSDS was still allocated by the master address space.

7. Submit SFBLDJjj jobs to dynamically create the SHVTjjnn (dump) and TEVTjjnn (restore) jobs. Verify that RC=0 or RC=4. If a SFBLDJjj job returns RC=4, non-DB2 data sets have been found under the corresponding DSNDBC second-level qualifier, and their VSAM cluster data set names have been written to an exception filter card named TVCAT.CLONE.SFBLDJCL.SHVTjjEX.FILTDD. In most cases, the system administrator will review them and delete them by manually creating a delete job for the exception filter card. These data sets are typically temporary DB2 data sets that were not cleaned up after an abnormal utility termination. Refresh your member list to see the generated jobs.
8. Submit SHVTjjnn jobs in parallel to dump the DB2 indexspace and tablespace data sets of the selected source subsystem VCAT names to sequential data sets. These sequential files are later used in the cloning process to restore the DB2 tablespace and indexspace data sets to the target subsystem under a different VCAT name. Verify that RC=0. If you run out of space here, delete the SVCAT.CLONE.**.DUMP data set and start over again. You may have to adjust the space allocation or other DD parms in the JCL. The DUMP may fail for a number of reasons. Make sure you are comfortable resolving these problems or involve your system programmer.

Tip: If you do not have enough space to hold all dump data sets and the restored DB2 data sets, you may want to submit a SFVTjjnn job, submit the corresponding TEVTjjnn job to restore the DB2 data sets, and then delete the dump data set to free allocated space. In this scenario, you can submit TBSYSLIB when you have restored all DB2 data sets on the target subsystem.

9. Submit TBSYSLIB to copy user-specified source subsystem library data sets to the target subsystem. STEP01 attempts to delete these data sets under the target subsystem VCAT name first. If you want to save certain members or the entire data set, you must do so before you run this job. Verify the return code of STEP02 and STEP03 is RC=0. STEP01 may return RC=8 if a data set to be deleted from the target subsystem did not exist. STEP03 may return RC=4 if a copied library was empty.

If you need to copy other data sets that are not included in this job, you can add them manually to this job. In our environment, for example, we had to manually add the data set DB7X7.SDSNEXIT to TBSYSLIB, as shown in Example 5-4 on page 104.

```
...
//STEP01.SYSIN DD *
  DELETE (DB7WU.DBRMLIB.DATA)
  DELETE (DB7WU.RUNLIB.LOAD)
  DELETE (DB7WU.SRCLIB.DATA)
  DELETE (DB7W7.SDSNEXIT)
//STEP02.SYSIN DD *
  DUMP -
  OUTDD(OUTDD1) -
  DATASET( -
  INCLUDE( -
    DB7XU.DBRMLIB.DATA, -
    DB7XU.RUNLIB.LOAD, -
    DB7XU.SRCLIB.DATA, -
    DB7X7.SDSNEXIT -
  ) -
  ) OPTIMIZE(4) COMPRESS -
  ALLDATA(*) -
  ALLEXCP
//STEP03.SYSIN DD *
  RESTORE -
  INDD(DUMP) -
  DATASET( INCLUDE (**) ) -
  RENUNC((DB7XU.** , DB7WU.**), (DB7X7.** , DB7W7.**)) -
  NSC -
  CATALOG
...
```

10.Restart the source DB2 subsystem.

Step 4: Restore the target subsystem and restart it

During this step, you restore the DB2 tablespace and indexspace data sets under their corresponding target subsystem VCAT name and cold start the target subsystem in maintenance mode to prepare it for the last cloning step:

1. Submit TEVTjjnn jobs in parallel to restore the dumped DB2 indexspace and tablespace data sets using the DFSMSdss RESTORE command under their corresponding target subsystem VCAT name. Verify that RC=0. RESTORE can fail for many reasons, you need to be comfortable using RESTORE and resolving problems that occur.
2. Use the TSO Data Set List Utility (3.4) to list all VSAM data sets under the source subsystem VCAT names and under the target subsystem VCAT names (such as SVCAT.DSNDBC.** and TVCAT.DSNDBC.**). The record count must match.

Note: This is not relevant in the case of MCODE cloning.

3. Submit TLCHGLOG to update the STARTRBA and ENDRBA values of the BSDS with the highest RBA at the time the source subsystem was stopped (rounded up to the nearest 4 K). Verify that RC=0.
4. Create a temporary ZPARM member that will be used for the initial start of the target DB2 subsystem. Use the original DSNTIJUZ job that compiles (step DSNTIZA) and link edits (step DSNTIZL) the DSNZPARM member of the target DB2 subsystem and change the SYSADM and SYSADM2 values to the user ID of the system administrator performing the cloning. Compile (step DSNTIZP) and link edit (step DSNTIZQ) the DSNHDECP member. Verify that the new DSNZPARM and DSNHDECP members have been compiled and link edited with RC=0 into the SDSNEXIT library of the target subsystem.

Important: The source and target DSHDECP members should be identical (apart from SSID). In particular, the source and target values for CCSIDs, SQLDELI, and DECIMAL must be identical.

5. Issue START DB2 PARM(DSNZTEMP) ACCESS(MAINT) to start the target DB2 subsystem in maintenance mode. DSNZTEMP stands for the name of the temporary ZPARM member you created in the previous step. Starting DB2 in maintenance mode prohibits the connection of any authorization IDs other than Install SYSADM and Install SYSOPR. Because DB2 will detect that a cold start is requested, you will be asked to reply Y to continue the DB2 start process. Verify that the target subsystem was started by checking that the following message is printed to the console:

```
DSN9022I  -ssid DSNYASCP 'START DB2' NORMAL COMPLETION
```

Step 5: Alter the target subsystem

During this step, you alter the target subsystem DB2 catalog to the target subsystems VCAT names and drop DB2 objects that were temporarily used during the cloning process or excluded from the cloning process.

Note: If you experience any SQL problems here (for example, unavailability of a catalog index), that problem existed in the source subsystem. It is the best if you start the JCL process all over again, manually clean up the temporary cloning data sets, and start from PBBLDJjj stops, of course, after you have solved the problem in the source subsystem (rebuilt the index and so forth).

Proceed as follows:

1. Submit WBALTS GP to alter the buffer pools on the target subsystem according to the source subsystem and to create a temporary storage group for the cloning process. Verify that RC=0.
2. Submit WCALTU IX (only if SAALTID X in step 3 returned RC=0) to alter application-defined indexes on the catalog. Verify that RC=0.
3. Submit WEWORKFL to drop and recreate the work file database. Verify that RC=0.
4. Submit WGALTD B2 to alter all DB2 managed tablespaces and indexspaces to use the temporary storage group. Verify that RC=0.
5. Submit WIALTD B2 to drop and create storage groups on the target subsystem using the corresponding target subsystem VCAT names and to grant use of the storage groups to the same user IDs as on the source subsystem. Verify that RC=0.

Note: In the case of M COD cloning, the names of the storage groups created on the target subsystem must match the following naming convention <TSAPSID>XXX (for example, ORABTD in our scenario). Therefore, you must manually update the CREATE statements in the file TVCAT.CLONE.SAALTQRY.OUTDB26 to reflect this requirement.

6. Submit WKALTD B2 to alter all DB2-managed tablespaces and indexspaces to use the original storage group. Verify that RC=0.

Note: In the case of M COD cloning, you must manually update the ALTER statements in the file TVCAT.CLONE.SAALTQRY.OUTDB28 to use the correct storage groups (that is, <TSAPSID>XXX instead of <SSAPSID>XXX).

7. Submit WLALTUSR (only if SAALTUSR in step 3 returned RC=0) to alter user-managed objects to their corresponding target VCAT name. Verify that RC=0.

Note: All objects in the catalog, even the ones that were excluded from the cloning process, are altered to a target system VCAT so that when they are dropped by the cleanup job, the data sets in the primary subsystem are not deleted. At this point in the procedure, there must be no source subsystem VCAT in the SYSIBM.SYSTABLEPART, SYSIBM.SYSINDEXPART, and SYSIBM.SYSSTOGROUP catalog tables.

8. Submit WNALTWLM to alter the WLM application environment of stored procedures and user-defined functions on the target subsystem. Verify that RC=0.
9. Submit WSCLNUPT to drop DB2 objects that were temporarily used or excluded from the cloning process. Verify that RC=0.

Note: WSCLNUPT only drops the storage groups that refer to an excluded VCAT. Therefore, the storage groups for the SAP system using the primary VCAT are not dropped automatically. It is your responsibility to do so.

10. Submit WTVRFALT to check the DB2 catalog tables for any references to VCAT names from the source subsystem. Verify that no rows are returned.
11. Stop and start the DB2 subsystem with the original ZPARM member.

From a DB2 point of view, the system is now successfully copied.

Step 6: Perform post cloning activities

Depending on your applications, you may have to perform the following post-cloning activities:

1. Install the ICLI server on the target subsystem.
In the case of MCODE cloning, the plan name used on the target subsystem should be F<TSAPSID><REL> (for example, FORA46D in our scenario).
2. Install the target application server.
In the case of MCODE cloning, follow the procedure described in “Set up the target application server” on page 35. Whenever the creator/schema is specified (for example, in the file MCODE_UNIX_db2.base.R3S, or in DEFAULT.PFL), it must refer to SCHEMA (for example, SAPGRE in our scenario).
3. Perform the steps as described in Chapter 10, “General subsequent actions,” in *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678, or Chapter 4, “Final activities,” in *SAP Web Application Server 6.20: Homogeneous and Heterogeneous System Copy*.
4. After all the verifications have been performed, back up the DB2 subsystem and the application data. If you have hardware-assisted volume level copy available, use it to get a recovery copy and return the target system to the end users. However, get images copies as soon as you can. If you do not have hardware assisted volume level copy, execute image copy jobs prior to returning the target system to the end users.

5.3.9 Additional considerations

In this section, we demonstrate with a few examples how to modify the JCL generated by the Control Center in order to perform dump and restore to and from tape and to clone in a data sharing environment.

Dump and restore to and from tape

When you have to copy a DB2 subsystem to tape to restore it on a different sysplex, you can still use the JCL that is generated by the Control Center cloning wizard. However, you have to slightly adapt the sequence of the cloning steps and edit some of the JCL as described in the following. Before cloning to tape, make sure that sufficient volumes and drives are available.

Estimate the number of tapes required for cloning

If you have run STOSPACE on the source subsystem, the Change dialog box on the Source Data Sets panel in the cloning wizard displays the space in MB under each VCAT. Using that information, you can calculate how many tapes are required for cloning (based on the type of tape: 3480, 3490, or 3590). In addition to the dump data sets, you have to put the DB2 libraries, the JCL library, and the cloning work data sets on tape.

Perform the following steps on the source system

Follow the instructions for “Step 1: Prepare for cloning” on page 99 as described when cloning to DASD. Skip “Step 2: Prepare the target subsystem” on page 100 if you clone to tape.

During “Step 3: Check source environment and copy subsystem” on page 101, submit SAADSPY, SAALTIDX, SAALTQRY, SAALTUSR, and SDRPTLOG as described when cloning to DASD. Then, create a job SEDMPWRK to dump the following data sets on tape (see Example 5-5), because they will be required on the target subsystem.

Example 5-5 Cloning work data sets to be dumped and restored on target system

```
CLONE.JCLLIB  
TVCAT.CLONE.SAALTIDX.OUTIX1  
TVCAT.CLONE.SAALTIDX.OUTIX2  
TVCAT.CLONE.SAALTQRY.CLEANUP1  
TVCAT.CLONE.SAALTQRY.CLEANUP2  
TVCAT.CLONE.SAALTQRY.OUTBP  
TVCAT.CLONE.SAALTQRY.OUTDB22  
TVCAT.CLONE.SAALTQRY.OUTDB24  
TVCAT.CLONE.SAALTQRY.OUTDB26  
TVCAT.CLONE.SAALTQRY.OUTDB28  
TVCAT.CLONE.SAALTQRY.OUTSGP1  
TVCAT.CLONE.SAALTQRY.OUTSGP2
```

```
TVCAT.CLONE.SAALTQRY.OUTWLM
TVCAT.CLONE.SAALTQRY.OUTWRKF1
TVCAT.CLONE.SAALTQRY.OUTWRKF2
TVCAT.CLONE.SAALTQRY.OUTWRKF3
TVCAT.CLONE.SAALTUSR.OUTUSR1
TVCAT.CLONE.SAALTUSR.OUTUSR2
TVCAT.CLONE.SDPRTLOG.STEP01.SYSPRINT
TVCAT.CLONE.SFBLDJCL.JOBNAMEF
```

Submit SFBLDJjj jobs to generate the SHVTjjnn jobs.

Before you submit the dump jobs, make sure that you edit OUTDD1 DDDEF so that the data set is put on tape. Submit the SHVTjjnn jobs in the same sequence as they have been placed into the library. Do not submit them in parallel.

Before you run TBSYSLIB, remove the delete step and the restore step. Change the DDDEF for OUTDD1 so that the TVCAT.CLONE.TBSYSLIB.STEP02.DUMP is dumped on tape.

Perform the following steps on the target system

If you have previously restored a target system from tape on your target system, manually delete the previously restored work data sets (including the cloning JCL library). Then, create a job SERSTWRK to restore the work data sets from tape.

Follow the instructions for “Step 2: Prepare the target subsystem” on page 100 as described when cloning to DASD. Then go to “Step 4: Restore the target subsystem and restart it” on page 104.

Create a job TRSYSLIB to restore TVCAT.CLONE.TBSYSLIB.STEP02.DUMP from tape.

Before you submit the restore TEVTjjnn jobs, make sure that you edit the OUTDD1 DDDEF so that the data sets are restored from tape. Submit the TEVTjjnn jobs in the same sequence as they have been put into the library, which will correspond to the order in which they are stored on the tapes. Do not submit the restore jobs in parallel.

Edit TBSYSLIB and delete STEP02. Change the DUMP data set DDDEF for STEP03 so that the libraries are restored from tape.

Follow the instructions for “Step 5: Alter the target subsystem” on page 105 and “Step 6: Perform post cloning activities” on page 107 as described when cloning to DASD.

Data sharing to non-data sharing

The current release of Control Center only supports HSC between non-data sharing subsystems. The support for data sharing will be added in an upcoming release. However, cloning *between data sharing systems* can be accomplished even with the current Control Center support with some modifications of the JCL and the cloning instructions. An experienced DB2 system administrator can make these modifications.

As an example, we describe how the cloning process can be adapted when the source system *is* a data sharing group and the target system is *not* a data sharing group. The sequence of the steps is basically the same as when cloning between non-data sharing subsystems. Only a few JCLs need to be slightly modified.

Follow the instructions for “Step 1: Prepare for cloning” on page 99 and “Step 2: Prepare the target subsystem” on page 100 as described when cloning between non-data sharing subsystems.

Step 3: Check the source environment and copy the subsystem

Because the source system is a data sharing group, you must display activity on *all* the members of the source data sharing group. There should be no outstanding activity, utility, or database with spaces in restricted state. The job SAADSPY issues the following commands on *one* member of the data sharing group:

```
-DISPLAY THREAD(*)  
-DISPLAY UTILITY(*)  
-DISPLAY DB(*) SPACE(*) RES LIMIT(*)
```

The last two commands have a global scope. However, it is your responsibility to issue the command DISPLAY THREAD(*) on all the members of the source data sharing group and verify that there is no outstanding activity.

All members of the source data sharing group must be stopped before you can submit SDPRTLOG. You can modify SDPRTLOG, as shown in Example 5-6 on page 111, to take into account the BSDS of all members in the source data sharing group. STEP02 and STEP03 will not work properly in a data sharing environment (DSNACCC6 parses the log map report for RBA instead of LRSN). Therefore, you should remove these steps from the JCL. It is your responsibility to make sure that all the members of the source data sharing group were properly quiesced.


```
...
//*
//SDPRTLOG PROC
//STEP01 EXEC PGM=DSNJU004
//SYSUT1 DD DISP=SHR,DSN=&BSDS01
//SYSPRINT DD DSN=&HLQ..CLONE.SDPRTLOG.&MEMNO..SYSPRINT,
//          UNIT=SYSDA,
//          DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(2,1),RLSE),
//          DCB=(RECFM=FB,LRECL=125,BLKSIZE=12500)
//SYSUDUMP DD SYSOUT=*
//          PEND
//*-----END OF PROC-----
//JOB LIB DD DSN=DB7X7.SDSNLOAD,DISP=SHR
//GO EXEC SDPRTLOG,
//    BSDS01=DB7XU.D7X1.BSDS01,
//    HLQ=DB7WU,
//    MEMNO=01
//GO EXEC SDPRTLOG,
//    BSDS01=DB7XU.D7X2.BSDS01,
//    HLQ=DB7WU,
//    MEMNO=02
...

```

Step 4: Restore the target subsystem and restart it

The cloning process is an offline copy: The source systems are stopped and quiesced prior to copying, ensuring that the copied data is consistent. Therefore, the restart of the target system can be simplified by using a cold start (STARTRBA=ENDRBA). In a data sharing to non-data sharing case, the restart RBA of the target system must be the highest LRSN used in the source data sharing group, rounded up to the nearest 4 K. To round it up to the nearest 4 K, convert the LRSN to decimal, add 4096, convert it back to hexadecimal, and set the last three digits to 000.

You must manually check all TVCAT.CLONE.SDPRTLOG.&MEMNO..SYSPRINT files to find the highest used LRSN of the source data sharing group. Example 5-7 on page 112 shows an example of BSDS print with LRSN.

Example 5-7 BSDS print with LRSN

```
TIME OF CHECKPOINT      14:45:50 NOVEMBER 16, 2002
BEGIN CHECKPOINT RBA    0000022C27BC
END CHECKPOINT RBA      0000022C5160
END CHECKPOINT LRSN     B88780E25CD1
SHUTDOWN CHECKPOINT
```

You can then modify TLCHGLOG, as shown in Example 5-8, to create the conditional restart control record in the BSDS of the target subsystem.

Example 5-8 Modifying TLCHGLOG in a data sharing environment

```
...
//TLCHGLOG PROC
//STEP01 EXEC PGM=IEFBR14
//*
//STEP02 EXEC PGM=DSNJU003
//SYSUT1 DD DISP=SHR,DSN=&BSDS1.
//SYSUT2 DD DISP=SHR,DSN=&BSDS2.
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
CRESTART CREATE,STARTRBA=B88780E26000,ENDRBA=B88780E26000,BACKOUT=NO
//*
//STEP03 EXEC PGM=DSNJU004
//SYSUT1 DD DISP=SHR,DSN=&BSDS1.
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
// PEND
/*-----END OF PROC-----
//JOBLIB DD DSN=DB7W7.SDSNLOAD,DISP=SHR
//GO EXEC TLCHGLOG,
// BSDS1=DB7WU.BSDS01,
// BSDS2=DB7WU.BSDS02
/*-----END OF STEP-----
```

Step 5: Alter the target subsystem

WEWORKFL may not work properly in a data sharing environment. Because the source system is data sharing, and the target system is non-data sharing, the work database names could be different and will be different if the source system has more than one member in the group.

You can use the following SQL statement to determine the work database names on the source system:

```
SELECT NAME FROM SYSIBM.SYSDATABASE
WHERE TYPE='W';
```

You must remove these work databases and recreate DSNDDB07 (including the associated VSAM data sets as they were dropped in a previous step), as shown in Example 5-9.

Example 5-9 Dropping the source work databases and recreating DSNDDB07

```
//JOB LIB DD DISP=SHR,DSN=DB7W7.SDSNLOAD
//DSNTIC PROC
/* ***** */
/* DIRECTORY/CATALOG AMS INVOCATION INSTREAM JCL PROCEDURE */
/* ***** */
//DSNTIC EXEC PGM=IDCAMS
//SYS PRINT DD SYSOUT=*
//SYS DUMP DD SYSOUT=*
//DSNTIC PEND
/*
//DSNTIAS EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSPRT DD SYSOUT=*
//SYS PRINT DD SYSOUT=*
//SYS DUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB7W)
  -STOP DATABASE(WKDBD7X1)
  -STOP DATABASE(WKDBD7X2)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) PARM('RCO') -
    LIB('DB7WU.RUNLIB.LOAD')
//SYSIN DD *
  DROP DATABASE WKDBD7X1 ;
  DROP DATABASE WKDBD7X2 ;
/*
//DSNTICR EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT,DSNTIAS)
//SYSPRT DD SYSOUT=*
//SYS PRINT DD SYSOUT=*
//SYS DUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB7W)
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) PARM('RCO') -
    LIB('DB7WU.RUNLIB.LOAD')
//SYSIN DD *
  CREATE DATABASE DSNDDB07 ;
/*
//DSNTTMP EXEC DSNTIC,COND=((4,LT,DSNTIAS),(4,LT,DSNTICR))
//SYSIN DD *
  DEFINE CLUSTER -
    ( NAME(DB7WU.DSNDBC.DSNDDB07.DSN4K01.I0001.A001) -
      LINEAR -
      REUSE -
      VOLUMES(DB7W03) -
      CYLINDERS(300 10) -
```

```

        SHAREOPTIONS(3 3) ) -
    DATA -
    ( NAME(DB7WU.DSNDBD.DSNDB07.DSN4K01.I0001.A001) -
      ) -
    CATALOG(DB7WU)

DEFINE CLUSTER -
    ( NAME(DB7WU.DSNDBC.DSNDB07.DSN32K01.I0001.A001) -
      LINEAR -
      REUSE -
      VOLUMES(DB7W03) -
      CYLINDERS(30 30) -
      SHAREOPTIONS(3 3) ) -
    DATA -
    ( NAME(DB7WU.DSNDBD.DSNDB07.DSN32K01.I0001.A001) -
      ) -
    CATALOG(DB7WU)

//DSNTIST EXEC PGM=IKJEFT01,DYNAMNBR=20,
// COND=((4,LT,DSNTIAS),(4,LT,DSNTICR),(4,LT,DSNTTMP.DSNTIC))
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(DB7W)
    -STOP DATABASE(DSNDB07)
    RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
      LIB('DB7WU.RUNLIB.LOAD')
    -START DATABASE(DSNDB07)
    END
//SYSIN DD *
    CREATE TABLESPACE DSN4K01 IN DSNDB07
      BUFFERPOOL BP1
      CLOSE NO
      USING VCAT DB7WU;
    CREATE TABLESPACE DSN32K01 IN DSNDB07
      BUFFERPOOL BP32K1
      CLOSE NO
      USING VCAT DB7WU;
/*

```

PPT recovery of one system in an MCOD landscape

Each component of SAP installed into a Multiple Components in One Database (MCOD) landscape must conform to the same rules, restrictions, and general recommendations normally associated with running in a DB2 UDB for OS/390 and z/OS environment. These considerations are discussed in detail in a number of key IBM and SAP documents:

- ▶ *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, SAP document available on the SAP Service Marketplace quick link INSTGUIDES at:
<http://service.sap.com/instguides>
- ▶ *SAP R/3 on DB2 for OS/390: Database Availability Considerations*, SG24-5690
- ▶ *Storage Management for SAP and DB2 UDB: Split Mirror Backup/Recovery with IBM's Enterprise Storage Server (ESS)*, white paper by Sanjoy Das, Siegfried Schmidt, Jens Claussen, and BalaSanni Godavari, available at:
<http://www.storage.ibm.com/hardsoft/products/sap/smdb2.pdf>

The most important and current of these documents is *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*. Please note that this document contains information previously

found in *DB2/390: Backup and Recovery Options*, SAP Note 083000 and will contain the most up-to-date information about the topic of backup and recovery.

The most challenging issues to be addressed in an SAP on DB2 environment is obtaining points of consistency in a highly active SAP system environment. This challenge stems from both the sheer number of objects in an SAP system and the fact that all database objects must be considered interrelated, and hence, one combined unit of recovery.

Making use of the MCOD feature does not change the nature of these challenges. However, it does make one significant change in underlying assumptions: The DB2 subsystem is now no longer necessarily considered a unit of recovery in its own right. Depending on the SAP components contained in the MCOD DB2 subsystem, the correct definition is now that each component is a unit of recovery. As each component is installed with a unique schema, the units of recovery are now the sets of objects created under each schema.

6.1 Different reasons for MCODE usage

This section discusses the different reasons for MCODE usage.

Note: In this chapter, reference is made to the term *application owners*. This denotes that in most SAP implementations, there is a distinction drawn between the technical operations of the SAP software and knowledge of how the company customizes and implements the software. Organizationally, these are commonly in separate teams or business groups, although this is different in every situation. The references to application owners, therefore, acknowledge that this book is aimed at technical staff. However, almost always, the need for point-in-time recovery will be determined by the application staff. In this case, close cooperation between the teams in determining and carrying out the recovery requirements, and readers should equate references to application owners to the situation at their company.

6.1.1 Use of SAP MCODE for consistency

One of the major reasons for placing multiple SAP components into an MCODE environment is to allow database-level tools, particularly in the area of backup and recovery, to act on the entire DBMS content at one time. In this way, a DBMS-wide backup established with consistency also ensures the consistency of data between the contained SAP components. This is particularly the case when the logical movement of data between components is continuous and of a high volume.

In this scenario, the backup and recovery considerations are the same as for a normal single SAP component contained within a DB2 subsystem. The number of objects contained increases with the installation of more than one SAP component into a DB2 subsystem. Because one component alone could contain up to 12,000 databases, 12,000 tablespaces, and 24,000 indexes at SAP R/3 Release 4.6C, multiple, fully populated SAP components will result in a DB2 subsystem with a huge number of objects. Therefore, the challenges brought about by the size and complexity of the database are even greater.

The application of the *split mirror* technique of backing up an SAP database is highly applicable to an MCODE environment. The time required to perform such a backup is not dependent on the number of objects in the database, but only on the total volume of data on DASD and the time required to establish point-in-time copies of this data using DASD subsystem tools, such as ESS FlashCopy.

In this *Consistency Scenario*, it is unlikely that there will be a requirement to recover a subset of components. Doing so will violate the requirements for consistency that led to the adoption of an MCOD environment. However, for every rule, there is an exception. In the case where all application owners and the technical staff agree that the best course of action is to recover one or more components to different points in time than the remaining components, then 6.2, “Recovery scenarios” on page 118 should be followed.

6.1.2 Use of SAP MCOD for consolidation

If multiple components are installed into one DB2 subsystem as a means of reducing the number of DB2 subsystems required to support the SAP landscape, this is referred to here as a *Consolidation Scenario*. In a such a system, it is very possible that some action will introduce an error into one component, which requires the component in question to be restored to remove this error.

As an example, if a number of different development or quality assurance instances are installed into one DB2 subsystem, and a developer or administrator makes an error that results in severe data loss on one system, typically, the recovery of the data should not involve regressing work in the unaffected SAP components.

As is normally the case for any error that is introduced into an SAP component, the first recommended solution is to use SAP techniques to correct the problem if possible. This may involve restoring prior versions of SAP objects, determining if an SAP transport contains the objects, or potentially writing *compensation transactions* to reverse the DB work causing the problem. In all of these cases, the work in question must *only* be performed by persons suitably knowledgeable in the application component in question. Typically, this will also involve advice and assistance from SAP. However, it is possible that such techniques are not possible in some situations, and database-level backups are the only recovery method available.

6.2 Recovery scenarios

The following general steps need to be followed if recovery actions need to be taken for SAP components in an MCOD environment:

1. Determine the scope (SAP components) affected by the error.
2. Work with the “application owner” to determine if *application-level* recovery is possible.
3. If *database-level* recovery is required:
 - a. Determine the important times defined in Figure 6-1 on page 119.

- b. Determine the appropriate recovery technique to be used.
- c. Plan the coordination required for affected parties.
- d. Perform the recovery actions.

6.2.1 Determination of the scope of SAP components affected

Generally, the problem will affect one SAP component. For example, incorrect transport actions were performed, one or more ABAP programs in error was run, or a customizing activity was incorrectly executed. In most cases, the correction of the error will require all of the affected SAP components to be recovered to the same point prior to the problem being introduced.

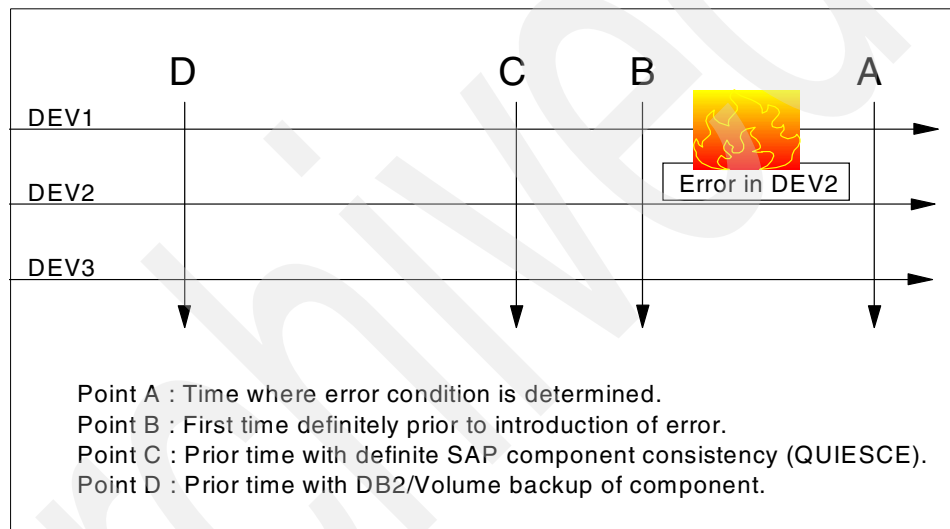


Figure 6-1 Important reference times required for recovery planning

6.2.2 Work with the application owners

As discussed, the first priority in a recovery situation is to determine if there is any way to address the problem by non-database means. This may involve analyzing the transport-induced problem through transport logs to determine if corrective transports can be created to fix the problem. It may involve the writing of a corrective ABAP program to reverse the effects of a previously executed program with errors. In the case where an SAP client has been corrupted by poor change control on customizing activity, this may involve a client copy of master and customizing data from a known good source, usually the production system.

Important: The corrective measures discussed are examples of common causes and possible recovery methods that may be available. Any action taken must be performed with full cooperation of the application owners' team.

6.2.3 Determining if a DB2 recovery is needed

Once we are sure that a recovery by DB2 means is required, the next step is to plan the appropriate recovery technique and target recovery time. A typical scenario is outlined in Figure 6-1 on page 119. In this case, we have a DB2 subsystem containing three SAP components. At some point in time, the cause of the error is triggered and runs for a period of time. Eventually, either during the introduction of the error, or afterwards, a determination is made that the error has occurred. This then triggers the recovery requirement. We can then determine the times, as illustrated in Figure 6-1 on page 119.

Point A

This is where the error condition is discovered. This point has no use from a recovery standpoint, because any recovery to this point will result in the error still existing. It does, however, define the point at which other actions may take place. These can include stopping the system to prevent further corruption or stopping users from taking action based on bad data.

Point B

This is the time determined to be the most recent time at which point the error condition did *not* exist in the system in question. In some circumstances, this can be defined exactly, such as the commencement of a SAP transport causing the error, with exact times available from logs. In other cases, the exact determination of Point B will be harder, for example, where SAP customizing activity over a period of time introduced the error, and the people performing this work are not aware of the exact step that caused the error.

Point C

Once Point B has been determined, either exactly or in the best manner possible, we determine the first point in time before this where a point of consistency exists for all objects in the SAP component. Typically, this will be a QUIESCE point that was obtained during the normal operation of administrative tasks. It is also possible to establish RBA/LRSN points by examining the DB2 log or logs where no units of work are in flight. A suitable tool is usually used for this purpose.

Point D

This is a point in time prior to Points A through C where a full copy of the DB2 subsystem has been made. Typically, most customers will perform this action using DASD subsystem tools acting at the media level, including RVA Snapshot, ESS FlashCopy, or EMC Timefinder. This point may also include an *offline* backup of DB2, where DB2 was stopped, or all databases are stopped or made read only, and DB2 COPY was used to establish a consistent copy of all data.

There is a wide spectrum of times possible based on the exact problem being addressed. In the case of a simple SAP transport caused problem, Points A through D may all lie within a 24 hour, or shorter, period. It is also possible that the issue being addressed is the slow “corruption” of a development system over a long period of time to the point where it is no longer usable. In this case, the determination of Point B may be difficult, or lie prior to the retention of suitable backups. There are, of course, a wide range of intermediate possibilities between these extremes.

6.3 Scenario: Recovery of one component

An example scenario is provided from the configuration used for this redbook, as illustrated in Figure 6-2 on page 122. This system has three SAP components that are all SAP Basis systems, WHT, YEL, and GRE, installed into the DB2 subsystem DB7X.

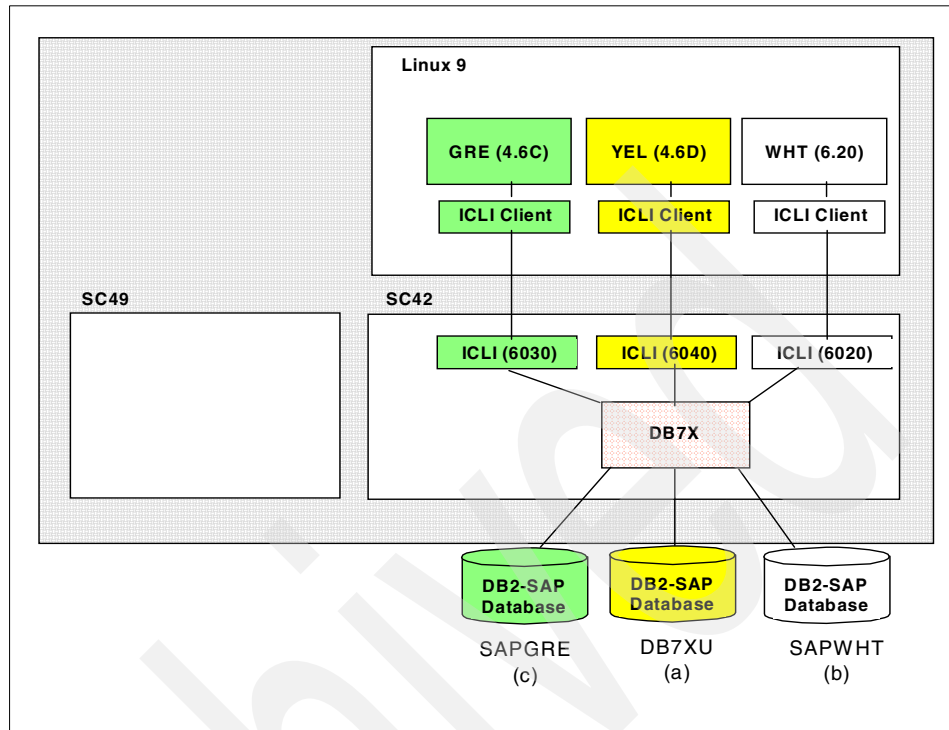


Figure 6-2 Prior point in time (PPT) recovery scenario: Initial configuration

6.3.1 Description of the cause of error

In the example scenario, we assume that the three systems are all non-production systems, where significant changes are made on a day-by-day basis, but do not have enterprise-critical activities taking place.

A typical example of the introduction of an error into the system is the importing of incorrect, or incorrectly sequenced, SAP transports into the system using the Transport Management System (TMS). The SAP system used to illustrate the error is YEL, a 4.6B Basis system. A typical SAP transport “window” is a fixed time of day where significant importing of objects using TMS occurs. As it is recommended that these windows occur during times of minimal activity, it follows that obtaining a point of consistency (QUIESCE) prior to the import starting should be achievable. We can then perform such a QUIESCE immediately prior to starting the TMS import.

This action results in units of work being applied to DB2, and we deem that the result of this work was our system reaching an unusable state.

6.3.2 Determination of recovery steps

The cause of error is a distinct action, being the import of all transports in the buffer for system YEL. Therefore, we are able to find Points A and B very quickly. We are also able to determine Point C because we had performed the -ARCHIVE LOG MODE(QUIESCE) command immediately prior to starting the TMS import.

We assume that we also had backed up the DB2 subsystem including all SAP data, both with ESS FlashCopy of the volumes while performing -SET LOG SUSPEND/RESUME. We also should have taken SHRLEVEL(CHANGE) image copies of all SAP objects as an additional possible recovery method at some time not too long before the problem occurred. Typically, this will be nightly image copies of all, or at least the most, volatile objects.

We want to carry out actions to recover the YEL system while still allowing users of the WHT and GRE SAP systems to carry out normal activity. We also assume that the contents of WHT and GRE systems are not to be restored, that is, their current state is preserved. If the recovery actions include steps, such as restoring from a FlashCopy backup, or a conditional restart, these actions become mutually exclusive both to continuing normal activities in the WHT and GRE systems and not regressing any data them.

It may be possible to recover objects for the YEL system without stopping DB2, such as by determining a QUIESCE point and, by analyzing the DB2 logs, determining which objects should be recovered to that point. This will only be the case if the following conditions are met:

- ▶ Number of objects is relatively small.
- ▶ Image copies are available from prior to Point B/Point C.
- ▶ The time duration and, more importantly, the amount of work performed in the system between the available copies and the recovery RBA is relatively small.

In this case, the YEL system would not be available for analysis during the course of the recovery of data as might be required.

If any of the above conditions are not met, the following actions may need to be performed to restore the SAP component to the required state:

1. Determine desired recovery target.
2. Create a clone of the current DB2 subsystem.
3. Restore the cloned copy to the desired recovery target point in time.

4. Use the merging techniques (either SAP export/import or merge in place) described in Chapter 3, “MCOD installation and merge using SAP tools” on page 27 and Chapter 4, “Merging SAP components without moving data” on page 41.

Determine recovery target

This has been discussed in some detail in previous sections. The broad trade-off is between complexity and duration of a recovery effort and the preservation of all units of work immediately prior to the error condition.

Create a clone of the current DB2 subsystem

This action is described in detail in *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287. By following the procedures outlined in this book, you are able to make a copy of the current DB2 system containing all SAP components. Chapter 5, “Cloning one component out of an MCOD landscape” on page 69 describes the process of cloning a single component from an MCOD system using the Control Center, and this technique is also applicable for this recovery scenario. It has the advantage of only working with the data of the SAP component we are recovering.

Restore the copy to the desired recovery target

Depending on the recovery point chosen, some of the recovery work may be incorporated into actions during the creation of the clone of the current system. Subsequent actions then may be necessary to take the cloned copy to the point in time we require. Options for recovery scenarios are documented thoroughly in *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, and this should be used as the primary reference in determining your course of action.

Merge recovered component back into the MCOD system

Once we are satisfied that we have recovered the component to the appropriate target time, we now want to restore the system to its original state with the recovered component back in the original DB2 subsystem. To do this, we are able to use either of the techniques outlined in Chapter 3, “MCOD installation and merge using SAP tools” on page 27 or Chapter 4, “Merging SAP components without moving data” on page 41.

Prior to the merge, it will either be necessary to delete the objects of the SAP component that existed prior to the clone, or allow the old and new copies to co-exist by altering the schema and VCAT chosen.

Figure 6-3 on page 125 summarizes the recovery steps.

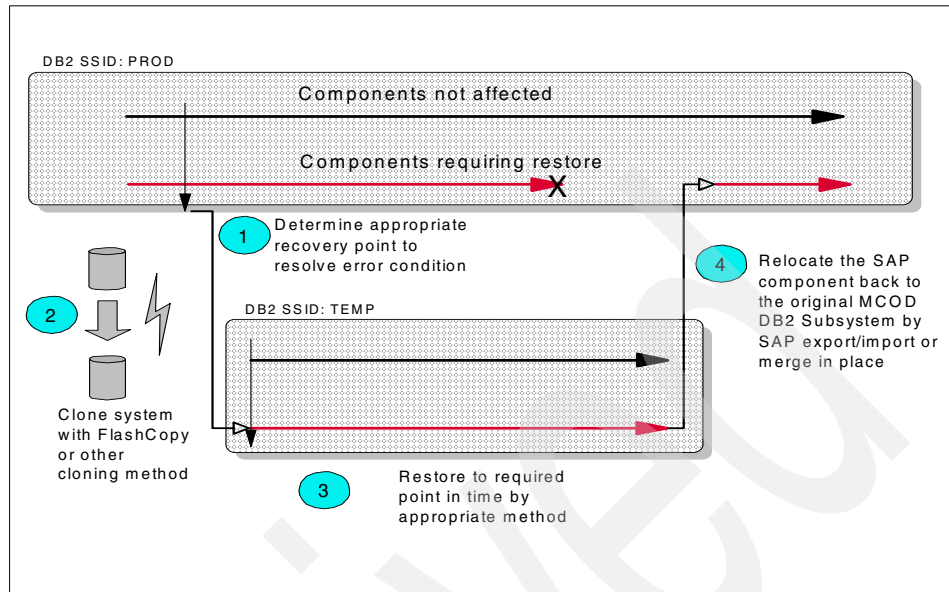


Figure 6-3 Steps involved in non-disruptive recovery of one SAP component

6.4 Additional considerations

This section discusses the applicability of the techniques described in this chapter to other applications, considerations involved where multiple SAP components require PPT recovery. In addition, we suggest the testing of these techniques in advance of requiring them in a real situation.

6.4.1 Application of scenario to other applications

As described in this chapter, the technique of cloning part of the content of a DB2 subsystem, recovering it through appropriate methods, and merging the subset of data back into the original system is applicable to non-SAP applications as well.

The definition of the component subset to clone is, in our case, assisted by the use of a distinct schema delimiting the DB2 objects in the SAP component. It is possible to use other techniques to delimit the data cloned to the subset required. It is also possible for simplicity in a PPT of one MCOB component to simply clone the entire system with all components if the basis of the recovery is that the cloned system will eventually be discarded. This negates the need for steps to limit the data involved and cleanup non-required objects. However, it may require more resources.

The choice of techniques, such as object-level recovery and conditional restart, in the cloned system is then subject to the requirements of the recovery. For SAP components, *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20* provides guidance and a discussion of the considerations. For non-SAP applications, the same general philosophies discussed in this guide also apply.

The merge back into the original DB2 subsystem restores the original configuration with no down time required for the applications residing there aside from any required for the cloning technique employed.

6.4.2 Considerations for multiple SAP component recovery

If a PPT recovery is required for more than one SAP component, as described in our scenario, the overall techniques are precisely the same. A cloning method needs to be chosen that will result in the required components being copied, for example, using the DB2 Control Center cloning wizard, which automates this process. The merge of the recovered systems also needs to be performed by adapting the techniques described to select the required data: either multiple executions of the merge with SAP tools, or adapting the data selection criteria in the merge without moving data technique.

6.4.3 Testing in advance

If you plan to set up an MCOD landscape, it is important that you understand and plan for the backup and recovery requirements. This will typically involve testing in advance the techniques that will be used to achieve your goals. If it is possible that you will require a PPT recovery of one component, we suggest that as soon as you have an MCOD landscape installed that is suitable for testing, and you have the resources available, you plan to perform a test of the techniques described that you have chosen to use for your system. This will allow you to discover any additional considerations unique to your implementations and also to expedite the recovery time when you need to perform the procedures in a real recovery situation.

Computer Center Management System (CCMS) and MCOD

This chapter discusses the following topics:

- ▶ Setup of Computer Center Management System (CCMS) in a Multiple Components in One Database (MCOD) landscape
- ▶ MCOD-specific enhancements

For additional information, refer to:

- ▶ *DB2/390: Installing saposcol manually*, SAP Note 103135
- ▶ *DB2/390: DB Performance Monitor/IFI Data Collector*, SAP Note 426863
- ▶ *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, SAP document available on the SAP Service Marketplace quick link INSTGUIDES at:

<http://service.sap.com/instguides>

7.1 Setup of CCMS in an MCODE landscape

In this section, we describe what needs to be set up to run CCMS in an MCODE landscape.

7.1.1 Required patches

To enable the new MCODE features, in particular with SAP Basis Releases 4.6C/D, you need to apply a variety of patches, transports, and support packages. Table 7-1 lists all the fixes that are needed.

Table 7-1 Patch levels and corrections

Component	Basis Release (system)		
	4.6C (GRE)	4.6D (YEL)	6.20 (WHT)
Basis support package	SAPKB46C24	SAPKB46D13	SAPKB62015
DDIC correction	SAPK46COSA	SAPK46DOS8	SAPKB62015
CCMS transport	M5CK000114	M5DK000180	SAPK620OC6
disp+work	4.6D / 1347		6.20 / 480
DBSL	4.6D / 1314		6.20 / 424
tp	4.6D / 1352		6.20 / 484
R3trans	4.6D / 1342		6.20 / 450

7.1.2 Setup of SAP collector tools saposcol and rfcocol

The main steps for the setup of the SAP collector tools saposcol and rfcocol are as follows:

1. Install collector tools on the z/OS host.

2. Bind rfcoscol using transaction DB2 (select **Checks/Settings** and then **SAP Collector Settings**). Figure 7-1 shows our input. Figure 7-2 on page 130 lists the resulting output.

Basic Settings	
Destination for	DB7X(wtsc42) is: SAP0SCOL_WT6C42
Plan name	CWHT620
Collection ID	CWHT620
Number of users	5
HFS path of RFCOSCOL	/u/sapres1/saposcol
RFCOSCOL trace directory	/u/sapres1/saposcol
SAP Collector user id	SAPRES1

Figure 7-1 Transaction DB2: SAP collector settings

```

List Edit Goto System Help
SAP
List JCL file contents

Job name: File name: 20021127_120943_linux9_20_w01.bi

      JES2 JOB LOG — SYSTEM SC42 — NODE W

12.09.46 JOB20938 — WEDNESDAY, 27 NOV 2002 —
12.09.46 JOB20938 ICH70001I SAPRES1 LAST ACCESS AT 12:09:46 ON WEDNESDAY, NOVEMB
12.09.47 JOB20938 $HASP373 SAPRES1Y STARTED - INIT A - CLASS B - SYS SC42
12.09.47 JOB20938 IEF403I SAPRES1Y - STARTED - TIME=12.09.47 - ASID=001A - SC42
12.09.47 JOB20938 - —TIMINGS (MINS.)—
12.09.47 JOB20938 -JOBNAME STEPNAME PROCSTEP RC EXCP CPU SRB CLOCK
12.09.47 JOB20938 -SAPRES1Y OCOPY 00 90 .00 .00 .0
12.09.48 JOB20938 -SAPRES1Y DBPACK 00 120 .01 .00 .0
12.09.48 JOB20938 -SAPRES1Y DBBIND 04 87 .00 .00 .0
12.09.48 JOB20938 IEF404I SAPRES1Y - ENDED - TIME=12.09.48 - ASID=001A - SC42
12.09.48 JOB20938 -SAPRES1Y ENDED. NAME- TOTAL CPU TIME=
12.09.48 JOB20938 $HASP395 SAPRES1Y ENDED

— JES2 JOB STATISTICS —
27 NOV 2002 JOB EXECUTION DATE
47 CARDS READ
178 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
8 SYSOUT SPOOL KBYTES
0.03 MINUTES EXECUTION TIME
!! END OF JES SPOOL FILE !!
1 //SAPRES1Y JOB USER=SAPRES1,CLASS=B,MSGCLASS=X, J
// MSGLEVEL=(1,1)
/* sample job head
/* $USER and $SUFFIX will be automatically replaced
/* $JOBHEAD WILL BE REPLACED AUTOMATICALLY

Return code of BIND job is 4 WHT (3) (000) linux9 INS

```

Figure 7-2 Transaction DB2: Output of the bind of SAP collector

For more details, refer to the following documents:

- ▶ Chapter 4 in *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, available at: <http://service.sap.com/instguides>
- ▶ *DB2/390: Installing saposcol manually*, SAP Note 103135 and *DB2/390: DB Performance Monitor/IFI Data Collector*, SAP Note 426863

7.1.3 Central DBA planning calendar

Starting with SAP Basis Release 6.10, transaction DB13C offers a central DBA planning calendar. For details, refer to 7.2.2, “Central DBA planning calendar (transaction DB13C)” on page 134.

The setup for a system based on SAP Basis Release 6.20 (for example, WHT) is as follows:

1. Ensure that all fixes listed in Table 7-1 on page 128 are applied to the SAP components involved.
2. Log on to the 6.20 system (WHT).
3. Call transaction SM59 and specify R/3 connections to both components GRE and YEL, as shown in Figure 7-3.

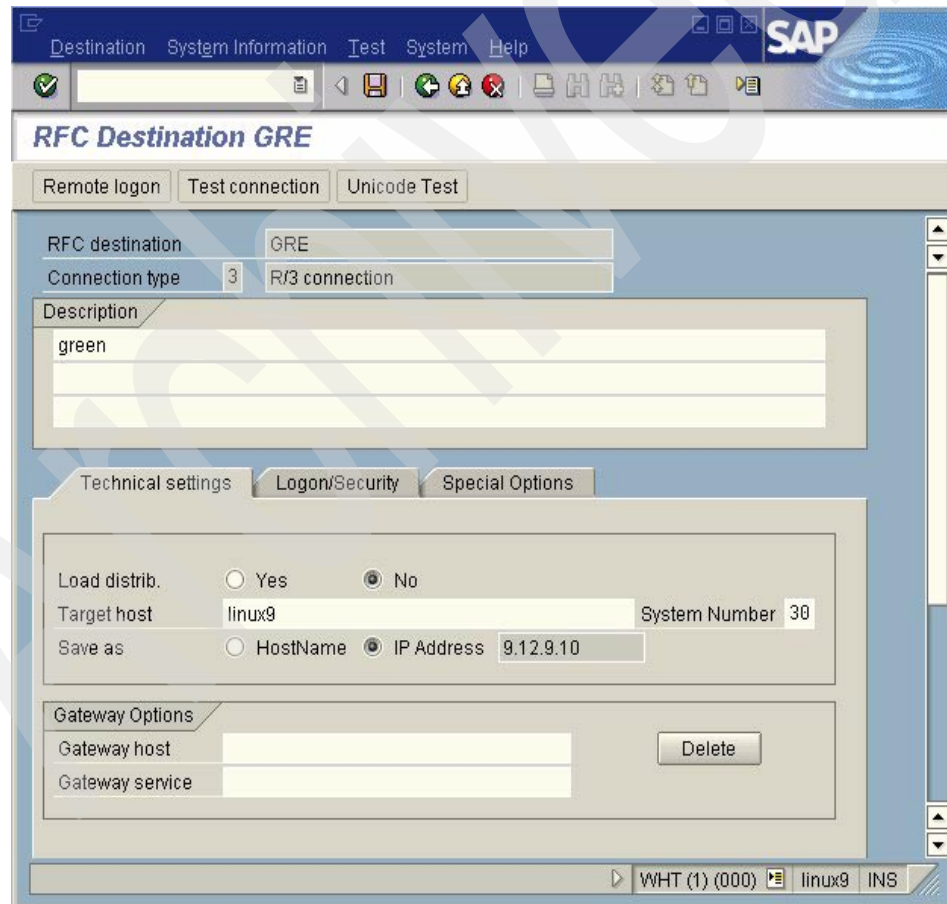


Figure 7-3 Transaction SM59: Create connection

4. Call transaction DB13C.
5. Select **Test connection** to verify that the connection works, as shown in Figure 7-4.

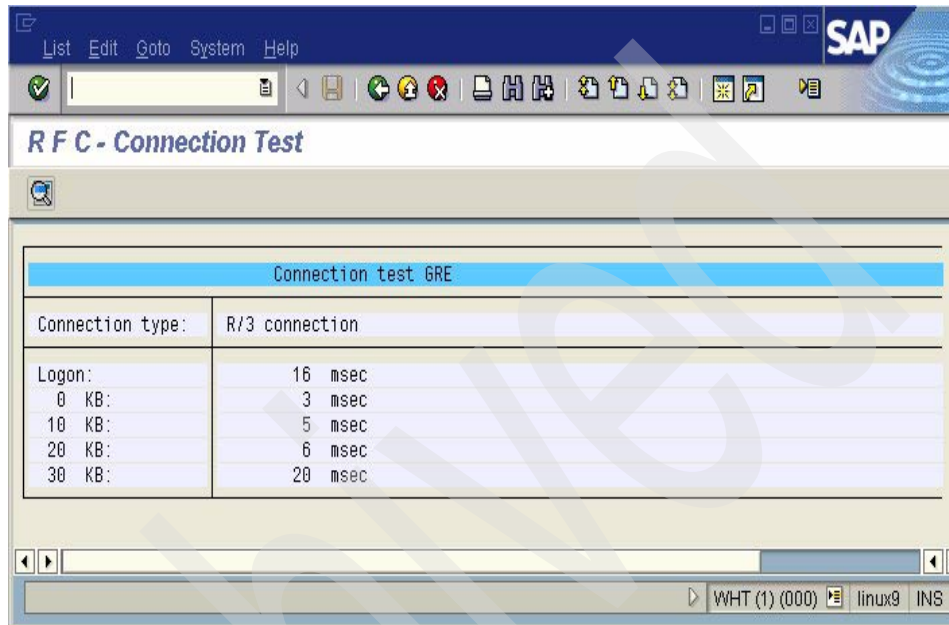


Figure 7-4 Transaction SM59: Test connection

6. Select **Configuration** → **Add Systems**, as shown in Figure 7-5.

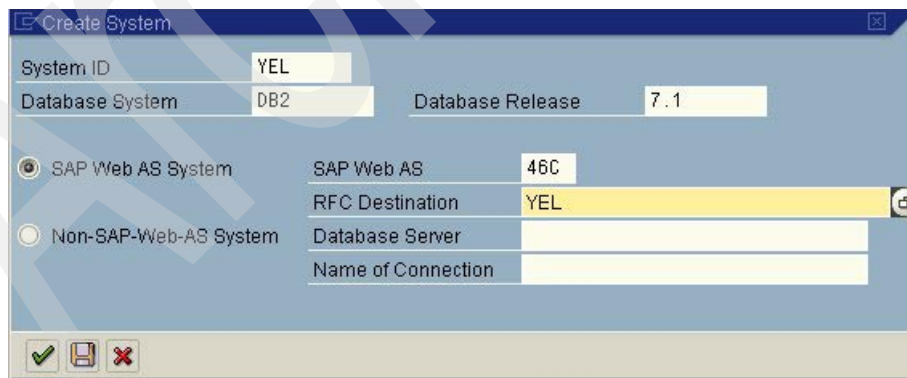


Figure 7-5 Transaction DB13C: Configuration of DBA planning calendar

7.2 MCOD enhancements

Most of the MCOD-related enhancements are not visible to the user. They are more or less hidden in the background, but guarantee that every transaction can be used in a non-MCOD environment.

For example, once started, SAP components become automatically visible to every other SAP component within the MCOD landscape. We see this by calling transaction DB02, as shown in Figure 7-6. All systems, YEL, GRE, and WHT, are listed in the second line of the SAP System frame.

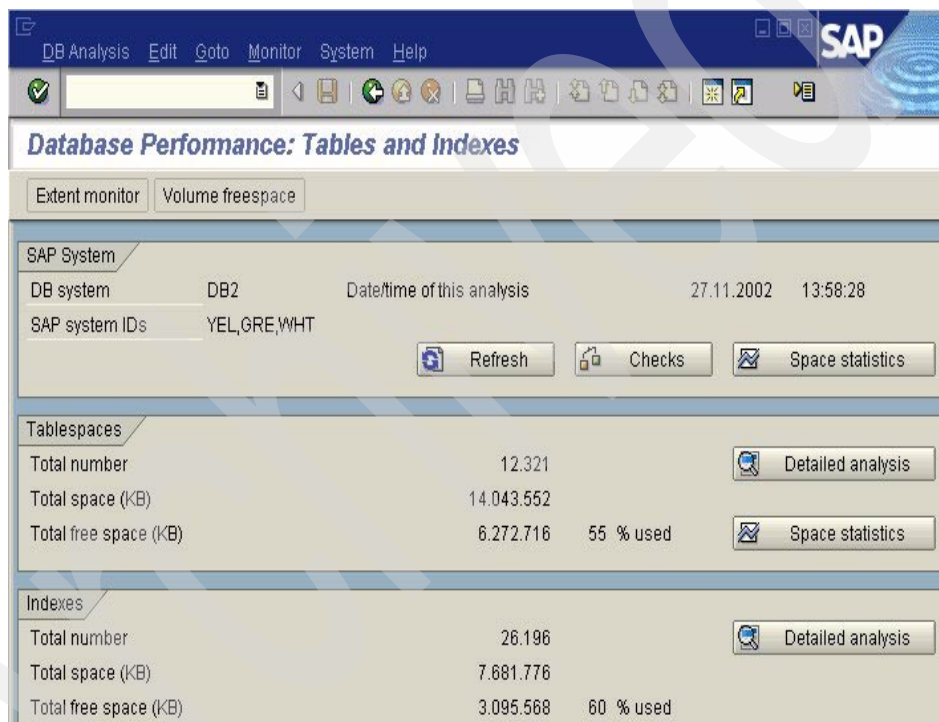


Figure 7-6 Transaction DB02: Initial screen

In the following sections, we introduce some of the transactions that now have an MCOD flavor added.

7.2.1 Tables and indexes monitor (transaction DB02)

In transaction DB02, statistical data is now collected and displayed for the complete MCOD landscape. If you select **Detailed Analysis**, as shown in Figure 7-6, the indexes are listed with their creator name, as shown in Figure 7-7.

Unfortunately, at the time of writing this book, the respective functionality for tablespaces does not yet provide any information about the related SAP component. However, this may change with future CCMS patches.

Index	Creator	Indexspace	Database	Number of Partitions	Part	Number of Extents	Allocated Size
ATAB~0	SAPWHT	ATABH0	P020XAAA	0	0	25	
ATAB~0	SAPGRE	ATABH0	P020XPCE	0	0	2	
ATAB~0	SAPR3	ATABH0	P020X000	0	0	1	

Figure 7-7 Transaction DB02: Detailed analysis output for indexes

7.2.2 Central DBA planning calendar (transaction DB13C)

The new transaction DB13C (central DBA planning calendar) is available with SAP Basis Release 6.10:

- ▶ It can be used to administer other SAP components (not necessarily installed in an MCODE environment).
- ▶ SAP components based on SAP Basis Release 4.6C and 4.6D passively support the transaction, that is, they can be administered from a SAP system with SAP Basis Release 6.20 and later.

For details, refer to *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*.

7.2.3 DBA planning calendar (transaction DB13): Backup

In transaction DB13, the Backup for all SAP tablespaces option triggers a full image copy of *all* SAP tablespaces in the MCODE installation. In our MCODE installation, we created a backup after finishing the installation of all SAP components. The steps are as follows:

1. Call transaction DB13.
2. Select **Edit** → **Schedule Action**.
3. Specify **Backup for all SAP tablespaces**, as shown in Figure 7-8.

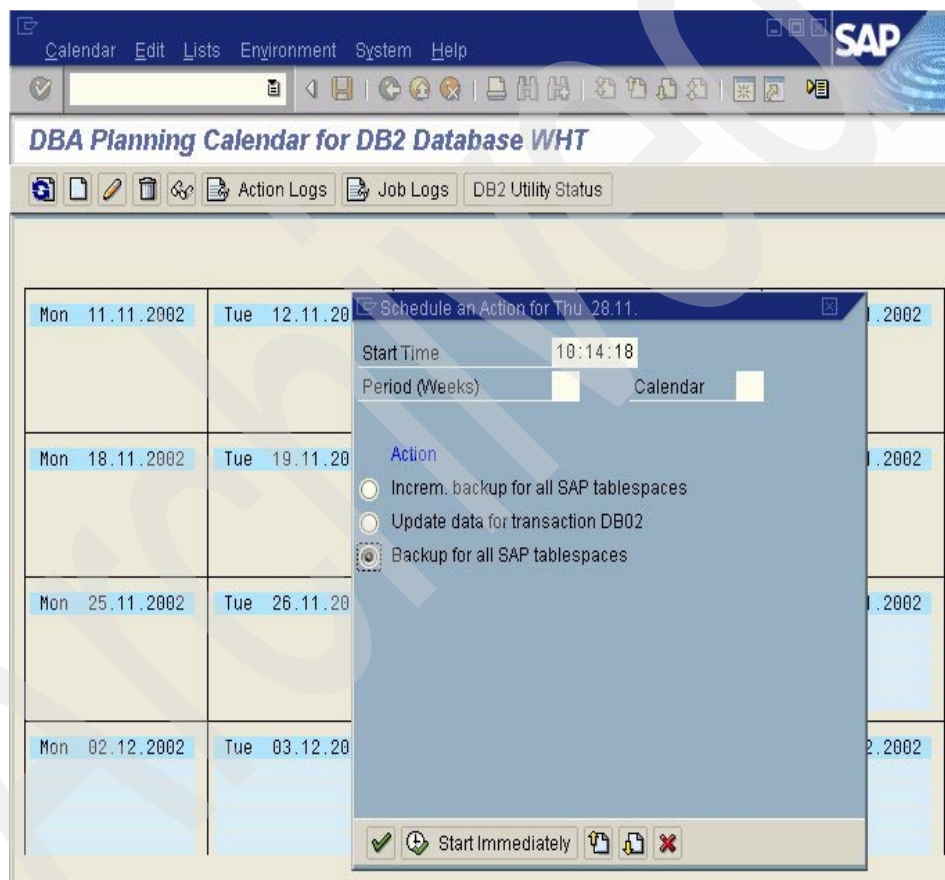


Figure 7-8 Transaction DB13: Backup

4. Continue as usual. The log output is shown in Figure 7-9 on page 136.

The screenshot shows the SAP DBA Planning Calendar for DB2 Database WHT. The interface includes a menu bar (Protocols, System, Help), a toolbar with various icons, and a title bar with the SAP logo. Below the title bar, there are two tabs: 'R/3 JES interface' and 'JES job output'. The main content area displays the date '28.11.2002' and a message: 'Jobs scheduled by user SAPRES1 and submitted or uploaded by the R/3 System. Use the R/3 JES interface (DB2J) to display your OS/390 jobs'. Below this, the target for upload is specified as 'SAPRES1.JCL(FICOPYA)'. The main part of the screenshot shows the JCL code for a job named 'SAPRES1A'.

```

>>>>>>>> BEGIN JCL
//SAPRES1A JOB USER=SAPRES1,CLASS=B,MSGCLASS=X,
//          MSGLEVEL=(1,1)
/** sample job head
/** $USER and $SUFFIX will be automatically replaced
/** $JOBHEAD WILL AUTOMATICALLY REPLACED
//DSNUPROC PROC LIB='DB7X7.SDSNLOAD',
//          SYSTEM=DB7X,
//          SIZE=0K,UID='SAPRES1.000040A',UTPROC=''
//DSNUPROC EXEC PGM=DSNUTILB,REGION=&SIZE,
//          PARM='&SYSTEM,&UID,&UTPROC'
//STEPLIB DD DSN=&LIB,DISP=SHR
//CVCDDTMT DD CVCDDT-*
```

Figure 7-9 Transaction DB13: Log output

7.2.4 Backup monitor (transaction DB12)

To monitor the backup situation, proceed as follows:

1. Call transaction DB12.
2. Select **New Analysis** to collect the backup information.
3. Select **Last successful tablespace backups**.

Figure 7-10 on page 137 shows the output.

The screenshot shows the SAP interface for transaction DB12, titled "Content of SYSIBM.SYSCOPY". It displays a table with columns for DB name, TS name, DSNum, Start RBA/LSRN, FileSeqNo, DevType, and Dataset name or dbname.tsname. The table lists 30 rows of backup data for various tables, including TBTCCT, BAPITool, SEUDEP, ICONXCL, TUTYPA, LANGUMS, TER13, ALTSTL, SEUDEP, TOASO, DSVASR, SXBNFR, DELREPS, SAASYSTT, OJDEFS, SCPRE, ALMTEC, IWAUTH, SWDSM, USR41, TLSYTAB, and TLSYOBJT. Each row shows a unique Start RBA/LSRN and FileSeqNo, and all DevType values are 3390. The dataset names are in the format SAPRES1.A000X00A.TBTCCTXT.IC2000.

DB name	TS name	DSNum	Start RBA/LSRN	FileSeqNo	DevType	Dataset name or dbname.tsname
A000X00A	TBTCCT...	0	F 0007E380E58A	0	3390	N SAPRES1.A000X00A.TBTCCTXT.IC2000
A000X02Y	BAPITool	0	F 0007F0FF7F40	0	3390	N SAPRES1.A000X02Y.BAPITool.IC2000
A000X03C	SEUDEP...	0	F 0007F11829BE	0	3390	N SAPRES1.A000X03C.SEUDEPTT.IC2000
A000X0CI	ICONXCL	0	F 0007F95FD188	0	3390	N SAPRES1.A000X0CI.ICONXCL.IC2000
A000X0CJ	TUTYPA	0	F 0007F09D49DF	0	3390	N SAPRES1.A000X0CJ.TUTYPA.IC2000
A000X0DP	LANGUMS	0	F 0007E385A54	0	3390	N SAPRES1.A000X0DP.LANGUMS.IC2000
A000X0...	TER13	0	F 0007F960B962	0	3390	N SAPRES1.A000X0DR.TER13.IC2000
A000X0I5	ALTSTL...	0	F 0007F101AF15	0	3390	N SAPRES1.A000X0I5.ALTSTL.DIC2000
A000X0NJ	SEUDEP...	0	F 0007F11B8D6B	0	3390	N SAPRES1.A000X0NJ.SEUDEPTT.IC2000
A000X0...	TOASO	0	F 0007EFF53FC	0	3390	N SAPRES1.A000X0QG.TOASO.IC2000
A000X0TS	DSVASR...	0	F 0007F09E5326	0	3390	N SAPRES1.A000X0TS.DSVASREP.IC2000
A000X15F	SXBNFR...	0	F 0007E38AD000	0	3390	N SAPRES1.A000X15F.SXBNFRUL.IC2000
A000X162	DELREPS	0	F 0007E38ADE...	0	3390	N SAPRES1.A000X162.DELREPS.IC2000
A000X1L4	SAASYSTT	0	F 0007F103A462	0	3390	N SAPRES1.A000X1L4.SAASYSTT.IC2000
A000X1MV	OJDEFS	0	F 0007F105DA19	0	3390	N SAPRES1.A000X1MV.OJDEFS.IC2000
A000X1...	SCPRE...	0	F 0007F11DE0...	0	3390	N SAPRES1.A000X1QM.SCPRENTY.IC2000
A000X1...	ALMTEC...	0	F 0007F0003962	0	3390	N SAPRES1.A000X1QU.ALMTECLS.IC2000
A000X25J	IWAUTH...	0	F 0007F0A00492	0	3390	N SAPRES1.A000X25J.IWAUTHFC.IC2000
A000X29A	SWDSM...	0	F 0007E3944D19	0	3390	N SAPRES1.A000X29A.SWDSMETH.IC2000
A000X2MI	USR41	0	F 0007F107EC33	0	3390	N SAPRES1.A000X2MI.USR41.IC2000
A000X2...	TLSTYTAB	0	F 0007F122AA38	0	3390	N SAPRES1.A000X2OU.TLSTYTAB.IC2000
A000X283	TLSTYOBJT	0	F 0007F0012492	0	3390	N SAPRES1.A000X283.TLSTYOBJT.IC2000

Figure 7-10 Transaction DB12: List of last successful backups

7.2.5 CCMS monitor set (transaction RZ20)

If the DB2 UDB for OS/390 and z/OS Version 7 is used, the recommendations on objects that need RUNSTATS, REORG, or COPY are given for the complete MCOD installation.

7.2.6 Database performance analysis (transaction ST04)

In general, most functions in transaction ST04, as shown in Figure 7-11, now offer a global view of the subsystem. Here are some examples.

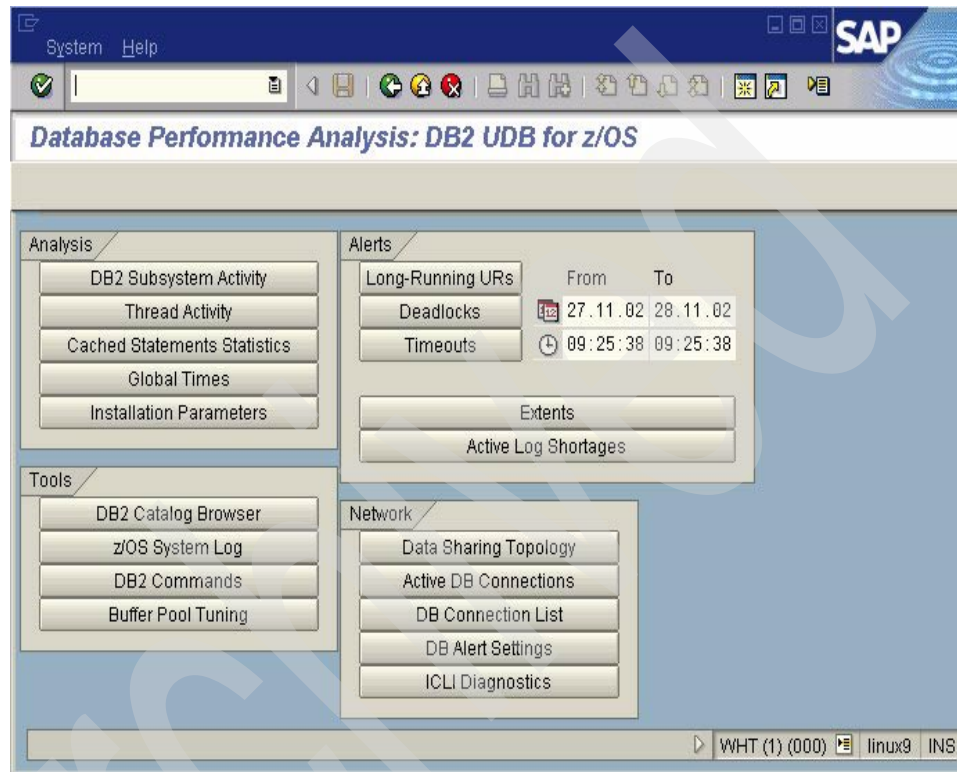
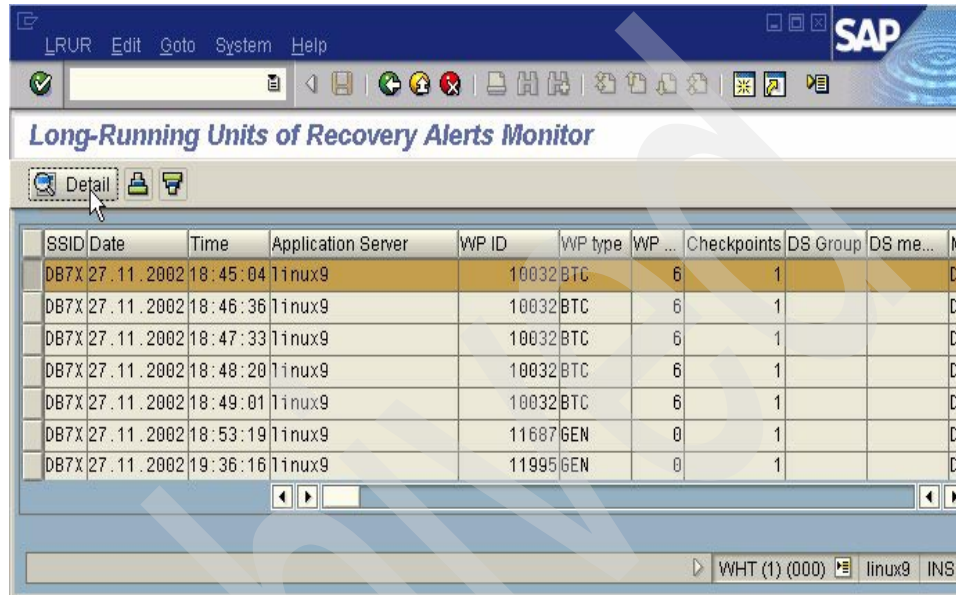


Figure 7-11 Transaction ST04: Initial screen

Alerts

All alerts within the MCOD landscape (deadlock, time-out, long-running units of recovery, log shortage, and extents) are visible. A sample output of Long-Running URs is depicted in Figure 7-12.



The screenshot shows the SAP interface for 'Long-Running Units of Recovery Alerts Monitor'. The window title is 'LRUR Edit Goto System Help'. The SAP logo is in the top right corner. Below the title bar, there is a toolbar with various icons. The main area contains a table with the following columns: SSID, Date, Time, Application Server, WP ID, WP type, WP ..., Checkpoints, DS Group, and DS me... (partially visible). The table contains seven rows of data. A 'Detail' button is visible above the table, and a status bar at the bottom shows 'WHT (1) (000) linux9 INS'.

SSID	Date	Time	Application Server	WP ID	WP type	WP ...	Checkpoints	DS Group	DS me...
DB7X	27.11.2002	18:45:04	linux9	10032	BTC	6	1		D
DB7X	27.11.2002	18:46:36	linux9	10032	BTC	6	1		D
DB7X	27.11.2002	18:47:33	linux9	10032	BTC	6	1		D
DB7X	27.11.2002	18:48:20	linux9	10032	BTC	6	1		D
DB7X	27.11.2002	18:49:01	linux9	10032	BTC	6	1		D
DB7X	27.11.2002	18:53:19	linux9	11687	GEN	0	1		D
DB7X	27.11.2002	19:36:16	linux9	11995	GEN	0	1		D

Figure 7-12 Transaction ST04: Long-running URs

Select **Detail**, and in the output, the related SAP component can be identified by its plan name, as shown in Figure 7-13.

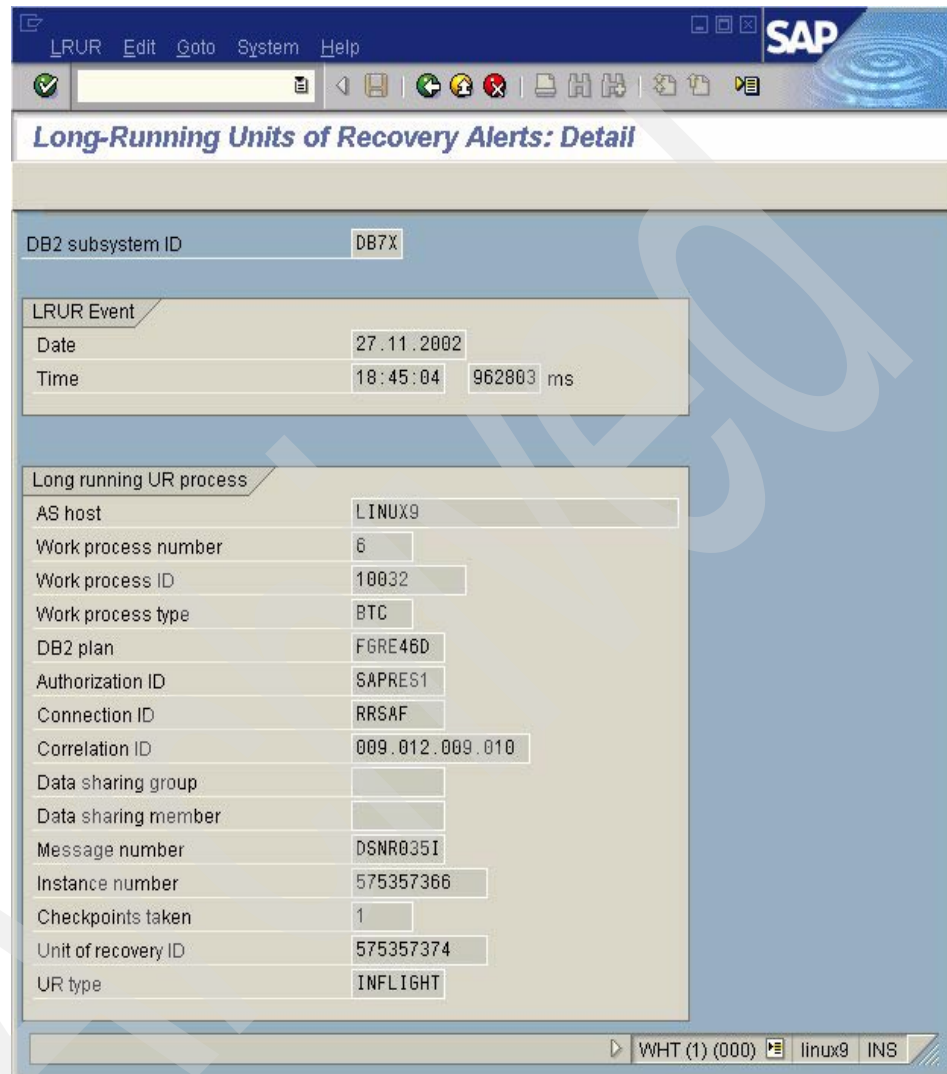


Figure 7-13 Transaction ST04: Long-running URs (Detail)

Statement cache statistics

The function to display cached statements is another example of a CCMS functionality adjusted to MCOD and is as follows:

1. Call transaction ST04.

2. Select **Statement Cache Statistics**.

We focus on the two ADMI_RUN tables. The output is shown in Figure 7-14.

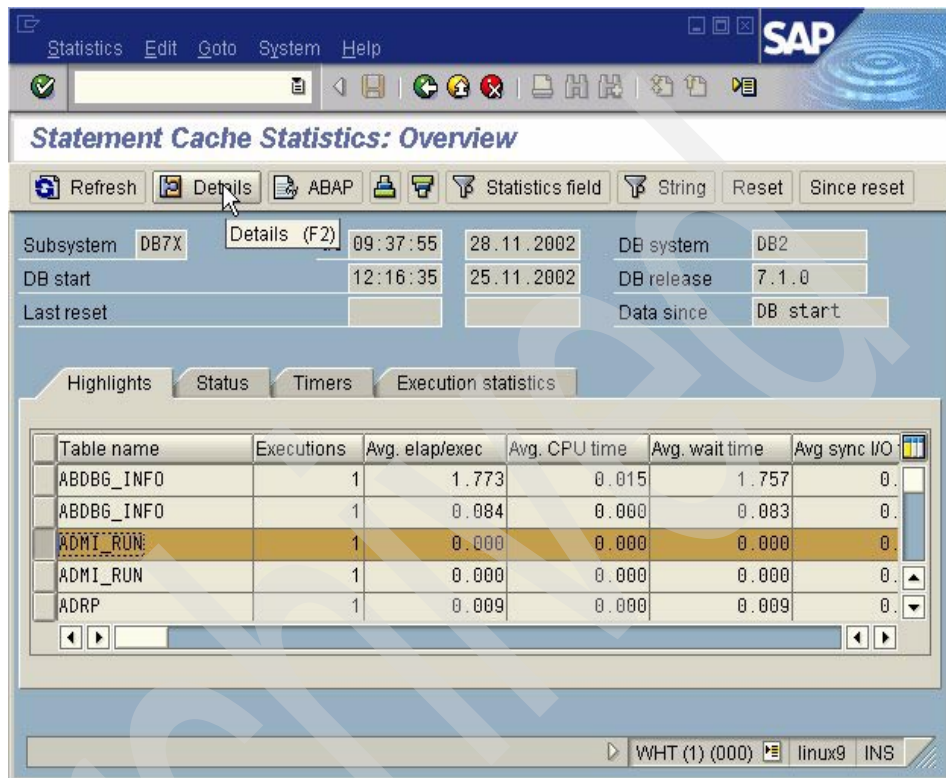


Figure 7-14 Transaction ST04: Statement Cache Statistics

3. Select **Details** for the output listed in Figure 7-15.

The screenshot shows the SAP 'Statement Cache Statistics: Details' window. At the top, there is a menu bar with 'Statistics', 'Edit', 'Goto', 'System', and 'Help'. Below the menu bar is a toolbar with various icons. The main content area is divided into several sections:

- Header:** 'Statement Cache Statistics: Details' with a search icon and 'Statistics field' and 'ABAP' buttons.
- Summary Fields:**
 - Subsystem: DB7X, at: 09:39:03, 28.11.2002, DB system: DB2
 - DB start: 12:16:35, 25.11.2002, DB release: 7.1.0
 - Last reset: (empty), Data since: DB start
- Tabs:** 'Statement text', 'Identification and status', 'Avg.time distr.per exec', 'Total time distr...'. The 'Identification and status' tab is selected.
- Fields in 'Identification and status' tab:**
 - Prep. stmn. cache ID: 1807
 - Program name: FOME6202
 - Stmt line number: (empty)
 - Transaction name: 1inux9
 - ABAP program name: RSDAMON
 - User ID: SAPRES1
 - User Group: SAPWHT
 - Initial prepare time: 11/26 03:36:14
 - User of statement: 0
 - Copies of statement: 0
 - Unqual tab names: SAPWHT
 - ISOLATION BIND: UR
 - CURDATA BIND: Yes
 - DYNAMICRULES BIND: RUN
 - CURRENT DEGREE: 1
 - CURRENT RULES: DB2
 - CURRENT PRECISION: DEC15
 - CURSOR HOLD: No
 - Status: valid

Figure 7-15 Transaction ST04: Statement Cache Statistics (Details 1/2)

4. For the second ADMI_RUN table, we obtained the output shown in Figure 7-16.

The SAP component can be easily identified. The parameter Unqual tab names displays the associated schema/creator.

The screenshot shows the SAP 'Statement Cache Statistics: Details' window. At the top, there's a menu bar with 'Statistics', 'Edit', 'Goto', 'System', and 'Help'. Below the menu is a toolbar with various icons. The main content area is divided into several sections:

- System Information:** Subsystem: DB7X, at: 09:40:09, 28.11.2002, DB system: DB2, DB start: 12:16:35, 25.11.2002, DB release: 7.1.0, Last reset: (empty), Data since: DB start.
- Statement Details:** A tabbed interface with 'Statement text', 'Identification and status', 'Avg.time distr.per exec', and 'Total time distr...'. The 'Identification and status' tab is active, showing:

Prep. stmn. cache ID	5859	Unqual tab names	SAPGRE
Program name	F0ME46D2	ISOLATION BIND	CS
Stmt line number		CURDATA BIND	No
Transaction name	linux9	DYNAMICRULES BIND	RUN
ABAP program name	RSDAMON	CURRENT DEGREE	1
User ID	SAPRES1	CURRENT RULES	DB2
User Group	SAPGRE	CURRENT PRECISION	DEC15
Initial prepare time	11/27 06:03:45	CURSOR HOLD	No
User of statement	0	Status	valid
Copies of statement	0		
- Footer:** A status bar at the bottom shows 'WHT (1) (000)', 'linux9', and 'INS'.

Figure 7-16 Transaction ST04: Statement Cache Statistics (Details 2/2)

Archived

Database layout

This appendix covers the following topics:

- ▶ Description of the database layout
- ▶ Historic overview about how the database layout has developed since SAP Release 3.0F
- ▶ Discussion of the rename procedure

For additional information, refer to the various release-dependent database administration guides: *BC SAP Database Administration Guides: DB2 for OS/390*

- ▶ SAP Release 3.1I: material number 51002788
- ▶ SAP Release 4.0B: material number 51002661
- ▶ SAP Release 4.5B: material number 51006377
- ▶ SAP Release 4.6C: material number 51009638
- ▶ SAP Releases 6.10/6.20: *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, available on the SAP Service Marketplace quick link INSTGUIDES at:

<http://service.sap.com/instguides>

Overview

Following is a short overview about how the database layout has developed since SAP Release 3.0F (when DB2/390 was first supported). The overview serves the following purposes:

- ▶ To better understand the enhancements, described in 1.2.2, “DB2-specific modifications” on page 5
- ▶ To handle the merge of historically grown systems

In the case of systems with mixed layouts, the rename procedure described in Chapter 4, “Merging SAP components without moving data” on page 41 may be modified to clean up the layout mixture. Possible solutions are discussed in “Rename procedure” on page 150.

General remarks

We do not describe the SAP technical settings (data class, buffering, and size category) here. For this and for a more detailed description of the database layout, refer to the administration guides listed in “Related publications” on page 269.

Table A-1 may serve as an useful reference in the following release-dependent sections (<NN> = two-digit number; <SID> = SAP System ID; the other two patterns <SB> and <ABC> are described as follows).

Table A-1 Comparison of database and stogroup names for SAP releases

	3.0F, 3.1H, 3.1I, 4.0B		4.6C/D, 6.10, 6.20	
Data class	Data stogroup	Database	Data stogroup	Database
APPL0	DSTAB<NN>	STAB<NN>	<SID>STD	A0<SB>X<ABC>
APPL1	DBTAB<NN>	BTAB<NN>	<SID>BTD	A1<SB>X<ABC>
APPL2	DPOOL<NN>	POOL<NN>	<SID>POD	A2<SB>X<ABC>
CLUST	DCLU<NN>	CLU<NN>	<SID>CLD	CL<SB>X<ABC>
POOL	DPOOL<NN>	POOL<NN>	<SID>POD	PO<SB>X<ABC>
SDIC	DDDIC<NN>	DDIC<NN>	<SID>DID	DI<SB>X<ABC>
SDOCU	DDOCU<NN>	DOCU<NN>	<SID>DOD	DO<SB>X<ABC>
SLDEF	DEL40B<NN>	EL40B<NN>	<SID>ELD	LD<SB>X<ABC>
SLEXC	DEL40B<NN>	EL40B<NN>	<SID>ELD	LX<SB>X<ABC>

	3.0F, 3.1H, 3.1I, 4.0B		4.6C/D, 6.10, 6.20	
SLOAD	DLOAD<NN>	LOAD<NN>	<SID>LOD	LO<SB>X<ABC>
SPROT	DPROT<NN>	PROT<NN>	<SID>PRD	PR<SB>X<ABC>
SSDEF	DES40B<NN>	ES40B<NN>	<SID>ESD	SD<SB>X<ABC>
SSEXC	DES40B<NN>	ES40B<NN>	<SID>ESD	SX<SB>X<ABC>
SSRC	DSOURC<NN>	SOURC<NN>	<SID>SOD	SO<SB>X<ABC>
TEMP	DPROT<NN>	PROT<NN>	<SID>PRD	TM<SB>X<ABC>
USER	DUSER1<NN>	USER1<NN>	<SID>U1D	U0<SB>X<ABC>
USER1	DUSER1<NN>	USER1<NN>	<SID>U1D	U1<SB>X<ABC>

R/3 Releases 3.0F, 3.1H, 3.1I, and 4.0B

The following applies to these releases:

- ▶ Multi-table tablespaces

Several databases are defined for each SAP data class (APPL0, APPL1, and so on). The name of a database consists of the storage ID that derives from the data class (APPL0 → STAB, SLOAD → LOAD) followed by a two-digit number (for example, STAB01).

Each database is associated with one or two tablespaces. The name of the tablespace is the prefix “B” (=32 k page) or “S” (=4 k page) followed by the database name. For example, tablespaces BSTAB01 and SSTAB01 are defined for database STAB01.

In addition, two stogroups are defined for each database. Their names are “D” (=data) for tablespaces and “I” (=index) for indexes, followed by the database name (for example, ISTAB01 and DSTAB01 for database STAB01).

More than 98% of all tables are created in multi-table tablespaces. They contain a maximum of 100 tables.

- ▶ Single-table tablespaces

Only large tables (SAP size category 3 or higher) have a tablespace, a database, and two stogroups of their own. The names of the database and tablespace consist of the first seven characters of the table name. For the two stogroup names a prefix of “D” or “I” is added, respectively.

- ▶ Stogroups

The number of active stogroups is twice the number of databases.

- ▶ Number of database objects
In a 4.0B R/3 system, there are approximately 400 databases and tablespaces, 800 stogroups, 13,000 tables, 3,000 views, and 15,000 indexes. Slightly more than 200 tables are located in a single-table tablespace.

R/3 Releases 4.5A and 4.5B

Release 4.5A brought major changes within the database layout. Still, we distinguish between multi-table and single-table tablespaces. However, the use and the naming convention is different from SAP Release 4.0B:

- ▶ Databases

For a given table the associated database name is constructed as follows:

<STORID><SB>#<ABC>

Where the parameters have the following meaning:

- <STORID> = SAP data class identified by two characters
- <SB> = Two digits that represent:
 - SAP size category (1 digit: 0, 1, 2, 3, and so on)
 - SAP buffering on application server (1 digit: 0 - no buffering, 1 - full buffering, 2 - generic buffering, 3 - single record buffering)
- <ABC> = Three random characters

There is a placeholder (#) for future developments (for SAP Basis Release 4.6B and later, the placeholder becomes X). Typical database names are A001#B35, CL10#Y94, and DI22#2H3.

- ▶ Multi-table tablespaces

All tables that are SAP buffered on the application server are placed into multi-table tablespaces. The name #SAP is assigned to this type of tablespace (for SAP Basis Release 4.6B and later the default name XSAP is used).

- ▶ Single-table tablespaces

Tables that are not SAP buffered are put into a single-tablespace. The name of the tablespace is derived from the table's name.

- ▶ Stogroups

There is a fixed number of stogroups defined for an SAP component. They are closely related to the data class. The relation is specified in tables TADB2 and IADB2 for data and index stogroups, respectively.

- ▶ Number of database objects

In a 4.5B R/3 system, we have approximately 16,000 tables, 19,000 indexes, and 4,000 views. These objects are stored in 32 stogroups, 7,400 databases, 200 multi-table tablespaces, and 7,200 single-table tablespaces.

Basis Releases 4.6B, 4.6C, and 4.6D

There are only minor changes as compared to R/3 Releases 4.5A/B.

- ▶ Databases and tablespaces (character # → X)

Character # in the database and tablespace names is not employed any more. Instead, character X is used. For example, the multi-table tablespaces are named XSAP now, and the database name is <STORID><SB>X<ABC>. That means, that the database names listed “R/3 Releases 4.5A and 4.5B” on page 148 are now A001XB35, CL10XY94, and DI22X2H3.

- ▶ Stogroups

In new MCOD installations, the first three characters of the stogroup names are identical to the SAP System ID if the creator is not identical to SAPR3. Refer to “Stogroups” on page 6 for details.

- ▶ Number of database objects

A 4.6C R/3 system is made up of approximately 26,000 tables, 31,000 indexes, 5,000 views, 24 stogroups, 12,000 databases, 11,700 single-table tablespaces, and 300 multi-table tablespaces. The numbers are considerably smaller in other SAP products (SAP BW, SAP APO, and so forth).

Basis Releases 6.10 and 6.20

The SAP Basis Releases 6.10 and 6.20 introduced the following changes:

- ▶ LOB columns

SAP components now utilize the DB2 data types CLOB and BLOB. Their usage requires the creation of additional objects: LOB tablespaces and auxiliary tables with special indexes. Their naming is derived from the names of the tables and their LOB columns.

- ▶ Databases and tablespaces

With Basis Release 6.20, the one-to-one relation between tablespace and database is removed. A database can now hold up to 100 single-table tablespaces or one multi-table tablespace with up to 100 tables.

- ▶ Number of database objects

For a newly installed SAP Enterprise 4.7 system (which is based on SAP Basis Release 6.20), the number of databases drops to around 1,000 (from 12,000 with R/3 4.6C SR2). There are approximately 33,000 tables and 5,700 views.

Rename procedure

When merging SAP components, we need to ensure the uniqueness of database and stogroup names. It is very likely that database names of SAP installations in different subsystems (that is, two non-MCOD installations) use identical names for the databases.

General procedure

In theory, the naming conflicts could be resolved manually:

1. Create a list of all databases in the source subsystem.
2. For each database name, find out whether it already exists in the target subsystem.
3. Determine a new unique database name if necessary.

In reality, this approach is not feasible. SAP components typically contain several thousands of databases and resolving the naming conflicts manually would take several days if not weeks. Therefore, we use an algorithm in this book (see Chapter 4, “Merging SAP components without moving data” on page 41) that automatically resolves the naming conflicts:

- ▶ The first three characters of the stogroup names are substituted by the SAP System ID.
- ▶ Database naming conflicts are removed by replacing the last three characters by random characters.

Mixed layouts

In our case (4.6C/D and 6.20 systems), the rename procedure is perfect. The algorithm also works (more or less) for mixed layouts (for example, 4.0B and 4.6C). However, the new names created from 4.0B and earlier database objects are a little bit awkward. For example:

- ▶ Database name STAB01 may become STAZ83 or STAB0Z83 (in case of a naming conflict).
- ▶ Stogroup name DSTAB01 transfers to PRDAB01 (with SAP System ID PRD).

One should take the opportunity to adjust the 4.0B database objects to a more 4.6x/6.x like database layout. This can be done by modifying the entries of the metadata tables described in Appendix B, “Merge in place: Working with metadata tables” on page 153.

Here are some suggestions:

- ▶ Apply new stogroup names, for example, map DSTAB01, DSTAB02, and DSTAB03 to <SID>STD and ISTAB01, ISTAB02, and so on to <SID>STI. The relation can be retrieved from Table A-1 on page 146.
- ▶ Similarly, the database names can be renamed (for example, STAB01 to A000XABC, DDDIC02 to DI00XCDE).
- ▶ For tables in single-table tablespaces, the handling is more difficult because the data class cannot be retrieved directly from the names of the database and tablespace. Instead, you may use ABAP function module DD_GET_STORAGE_CLASS in a little self-written ABAP program to determine the setting and store it in an additional metadata table.

Archived

Merge in place: Working with metadata tables

This appendix contains material that supports the metadata creation and population steps of the merge in place process. This includes DDL, REXX procedures, and JCL to perform the following tasks:

- ▶ Creating metadata tables
- ▶ Moving metadata tables across systems
- ▶ Working with metadata tables
- ▶ Updating metadata tables

Creating metadata tables

The metadata tables consist of the following five tables:

- ▶ ZMCOB_STOGRUP
- ▶ ZMCOB_DATABASE
- ▶ ZMCOB_TABLESPACE
- ▶ ZMCOB_TABLES
- ▶ ZMCOB_INDEXES

The DDL for creating these tables is included here.

DDL for creation of ZMCOBDB

In Example B-1, we provide the DDL for the database ZMCOBDB in which all the metadata objects are created.

Example: B-1 DDL for creation of ZMCOBDB

```
CREATE DATABASE ZMCOBDB
BUFFERPOOL BP3 STOGRUP BLUBTI CCSID ASCII ;
COMMIT;
```

DDL for creation of ZMCOB_STOGRUP

In Example B-2, we provide the DDL for the table ZMCOB_STOGRUP, which contains all the metadata related to storage groups.

Example: B-2 DDL for creation of ZMCOB_STOGRUP

```
CREATE TABLESPACE ZMCOBXS IN ZMCOBDB
USING STOGRUP SAPU1D PRIQTY 40 SECQTY 40
FREEPAGE 3 PCTFREE 20 GBPCACHE CHANGED
COMPRESS NO BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 4
LOCKMAX 0 CCSID ASCII ;

CREATE TABLE "ZMCOB_STOGRUP"
("SRCSTOGRUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSTOGRUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' )
CCSID ASCII IN ZMCOBDB.ZMCOBXS ;

CREATE TYPE 2 UNIQUE INDEX "ZMCOB_STOGRUP~0" ON "ZMCOB_STOGRUP"
("SRCSTOGRUP", "SRCVCAT")
USING STOGRUP SAPU1I PRIQTY 16 SECQTY 160
```

```
FREEPAGE 10 PCTFREE 10 GBPCACHE CHANGED  
BUFFERPOOL BP3 PIECESIZE 2097152 K CLUSTER ;
```

```
ALTER TABLE "ZMCOB_STOGROUP" ADD PRIMARY KEY ("SRCSTOGROUP", "SRCVCAT") ;
```

DDL for creation of ZMCOB_DATABASE

In Example B-3, we provide the DDL for the table ZMCOB_DATABASE, which contains all the metadata related to databases.

Example: B-3 DDL for creation of ZMCOB_DATABASE

```
CREATE TABLESPACE ZMCOBIDX IN ZMCOBDB  
USING STOGROUP SAPUID PRIQTY 104 SECQTY 160  
FREEPAGE 11 PCTFREE 20 GBPCACHE CHANGED  
COMPRESS NO BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 12  
LOCKMAX 0 CCSID ASCII ;
```

```
CREATE TABLE "ZMCOB_DATABASE"  
("SRCSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,  
"SRCRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,  
"SRCDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,  
"SRCSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,  
"SRCSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,  
"TRGSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,  
"TRGRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,  
"TRGDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,  
"TRGSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,  
"TRGSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' )  
CCSID ASCII IN ZMCOBDB.ZMCOBIDX ;
```

```
CREATE TYPE 2 UNIQUE INDEX "ZMCOB_DATABASE~0" ON "ZMCOB_DATABASE"  
("SRCSSID", "SRCRELEASE", "SRCDBNAME")  
USING STOGROUP SAPUII PRIQTY 16 SECQTY 160  
FREEPAGE 10 PCTFREE 10 GBPCACHE CHANGED  
BUFFERPOOL BP3 PIECESIZE 2097152 K CLUSTER ;
```

```
ALTER TABLE "ZMCOB_DATABASE"  
ADD PRIMARY KEY ("SRCSSID", "SRCRELEASE", "SRCDBNAME") ;
```

DDL for creation of ZM COD _TABLESPACE

In Example B-4, we provide the DDL for the table ZM COD _TABLESPACE, which contains all the metadata related to tablespaces.

Example: B-4 DDL for creation of ZM COD _TABLESPACE

```
CREATE TABLESPACE ZM COD XT IN ZM COD DB
USING STOGROUP SAPU1D PRIQTY 168 SECQTY 640
FREEPAGE 16 PCTFREE 20 GBPCACHE CHANGED
COMPRESS NO BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 20
LOCKMAX 0 CCSID ASCII ;

CREATE TABLE "ZM COD _TABLESPACE"
("SRCTSNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ;
"SRCDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCPART" INTEGER NOT NULL DEFAULT 0 ,
"SRCDBID" VARCHAR (000010) NOT NULL DEFAULT '0000000000' ,
"SRCRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"SRCIPREFIX" VARCHAR (000001) NOT NULL DEFAULT ' ' ,
"SRCPSID" SMALLINT NOT NULL DEFAULT 0 ,
"SRCSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCSPACE" SMALLINT NOT NULL DEFAULT 0 ,
"TRGTSNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,
"TRGRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"TRGDBID" VARCHAR (000010) NOT NULL DEFAULT '0000000000' ,
"TRGDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGIPREFIX" VARCHAR (000001) NOT NULL DEFAULT ' ' ,
"TRGPSID" SMALLINT NOT NULL DEFAULT 0 ,
"TRGSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' )
CCSID ASCII IN ZM COD DB.ZM COD XT ;

CREATE TYPE 2 UNIQUE INDEX "ZM COD _TABLESPACE~0" ON "ZM COD _TABLESPACE"
("SRCTSNAME", "SRCSSID", "SRCDBNAME", "SRCPART")
USING STOGROUP SAPU1I PRIQTY 16 SECQTY 160
FREEPAGE 10 PCTFREE 10 GBPCACHE CHANGED
BUFFERPOOL BP3 PIECESIZE 2097152 K CLUSTER ;

ALTER TABLE "ZM COD _TABLESPACE"
ADD PRIMARY KEY ("SRCTSNAME", "SRCSSID", "SRCDBNAME", "SRCPART") ;
```

DDL for creation of ZM COD _ TABLES

In Example B-5, we provide the DDL for the table ZM COD _ TABLES, which contains all the metadata related to tables.

Example: B-5 DDL for creation of ZM COD _ TABLES

```
CREATE TABLESPACE ZM CODXTX IN ZM CODDB
USING STOGROUP SAPU1D PRIQTY 104 SECQTY 160
FREEPAGE 11 PCTFREE 20 GBPCACHE CHANGED
COMPRESS NO BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 12
LOCKMAX 0 CCSID ASCII ;

CREATE TABLE "ZM COD _ TABLES"
("SRCSCHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCTABLE" VARCHAR (000018) NOT NULL DEFAULT ' ' ,
"SRCDNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,
"SRCRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"SRCTABOBID" VARCHAR (000005) NOT NULL DEFAULT ' ' ,
"SRCTSNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSCHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,
"TRGRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"TRGDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGTABOBID" VARCHAR (000005) NOT NULL DEFAULT ' ' ,
"TRGTSNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' )
CCSID ASCII IN ZM CODDB.ZM CODXTX ;

CREATE TYPE 2 UNIQUE INDEX "ZM COD _ TABLES~0" ON "ZM COD _ TABLES"
("SRCSCHEMA", "SRCTABLE")
USING STOGROUP SAPU1I PRIQTY 16 SECQTY 160
FREEPAGE 10 PCTFREE 10 GBPCACHE CHANGED
BUFFERPOOL BP3 PIECESIZE 2097152 K CLUSTER ;

ALTER TABLE "ZM COD _ TABLES"
ADD PRIMARY KEY ("SRCSCHEMA", "SRCTABLE") ;
```

DDL for creation of ZM COD _ INDEXES

In Example B-6 on page 158, we provide the DDL for the table ZM COD _ INDEXES, which contains all the metadata related to indexes.

Example: B-6 DDL for creation of ZM COD_INDEXES

```
CREATE TABLESPACE ZM CODXIX IN ZM CODDB
USING STOGROUP SAPU1D PRIQTY 104 SECQTY 160
FREEPAGE 11 PCTFREE 20 GBPCACHE CHANGED
COMPRESS NO BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 12
LOCKMAX 0 CCSID ASCII ;
```

```
CREATE TABLE "ZM COD_INDEXES"
("SRCINDEX" VARCHAR (000018) NOT NULL DEFAULT ' ' ,
"SRCSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCPART" INTEGER NOT NULL DEFAULT 0 ,
"SRCSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,
"SRCRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"SRCDBID" SMALLINT NOT NULL DEFAULT 0 ,
"SRCINDEXSPACE" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCIPREFIX" VARCHAR (000001) NOT NULL DEFAULT ' ' ,
"SRCISOBID" SMALLINT NOT NULL DEFAULT 0 ,
"SRCPRIQTY" VARCHAR (000010) NOT NULL DEFAULT '0000000000' ,
"SRCSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"SRCSPACE" SMALLINT NOT NULL DEFAULT 0 ,
"TRGSSID" VARCHAR (000004) NOT NULL DEFAULT ' ' ,
"TRGRELEASE" VARCHAR (000003) NOT NULL DEFAULT ' ' ,
"TRGDBID" SMALLINT NOT NULL DEFAULT 0 ,
"TRGDBNAME" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGINDEX" VARCHAR (000018) NOT NULL DEFAULT ' ' ,
"TRGINDEXSPACE" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGIPREFIX" VARCHAR (000001) NOT NULL DEFAULT ' ' ,
"TRGISOBID" SMALLINT NOT NULL DEFAULT 0 ,
"TRGSHEMA" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGSTOGROUP" VARCHAR (000008) NOT NULL DEFAULT ' ' ,
"TRGVCAT" VARCHAR (000008) NOT NULL DEFAULT ' ' )
CCSID ASCII IN ZM CODDB.ZM CODXIX ;
```

```
CREATE TYPE 2 UNIQUE INDEX "ZM COD_INDEXES~0" ON "ZM COD_INDEXES"
("SRCINDEX", "SRCSHEMA", "SRCDBNAME", "SRCPART")
USING STOGROUP SAPU1I PRIQTY 16 SECQTY 160
FREEPAGE 10 PCTFREE 10 GBPCACHE CHANGED
BUFFERPOOL BP3 PIECESIZE 2097152 K CLUSTER ;
```

```
ALTER TABLE "ZM COD_INDEXES"
ADD PRIMARY KEY ("SRCINDEX", "SRCSHEMA", "SRCDBNAME", "SRCPART") ;
```

Populating metadata tables

Once the metadata tables are created, we must populate them with data from the source DB2 catalog. In this section, we provide the REXX procedures used to populate the metadata tables, as well as the JCL we used to submit these procedures.

SETUPSTG

In Example B-7, we provide the REXX procedure used to populate the metadata table ZMCOB_STOGROUP.

Example: B-7 REXX to populate ZMCOB_STOGROUP

```

/*****/ 00010004
/* DB2 INITIALISATION - Populate MCOB Stogroup Table */ 00020004
/* VERBOSE - Print all SQL Return codes */ 00021004
/* PREDELETE - Delete table contents prior to run, allow reruns */ 00022004
/*****/ 00030004
00031003
PARSE ARG SRCSSID SRCSCHEMA SRCVERSION, 00031212
        TRGSSID TRGSCHEMA TRGVERSION, 00031312
        TRGVCAT TRGSTOGRPPRE 00031512
                                00032011
VERBOSE = N 00033015
PREDELETE = Y 00040015
                                00041003
/* Add DSNREXX to the host command environment table if not there. */ 00050000
                                00051000
'SUBCOM DSNREXX' 00052000
IF RC THEN , 00053000
    S_RC = RXSUBCOM('ADD', 'DSNREXX', 'DSNREXX') 00054000
ADDRESS DSNREXX 00055000
                                00056000
/*****/ 00057004
/* Connect to DB2 */ 00057104
/*****/ 00057204
                                00057304
'CONNECT' 'DB7S' 00058013
                                00060004
/*****/ 00070004
/* Initialize hardcoded variable values */ 00080004
/*****/ 00081004
/* 00082008
TRGVCAT = "MCDBLU" 00090006
TRGSTOGRPPRE = "BLU" 00100006
*/ 00122508
/*****/ 00122704
```

```

/* For rerunning this procedure, to prevent duplicates, set flag */ 00122804
/* PREDELETE to delete table contents before populating table */ 00122904
/*****/ 00123004
00123204
IF PREDELETE = Y THEN DO 00123303
    "EXECSQL DELETE FROM "SRCSCHEMA".ZMCOB_STOGRUP" 00123407
    SAY "SQL from delete is "SQLCODE 00123504
END 00123604
00123704
/*****/ 00123804
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00123904
/* This cursor remains open and loops through SYSSTOGRUP rows */ 00124004
/*****/ 00124104
00124204
SQLSTMT = "SELECT NAME,      ", 00124300
          "VCATNAME ", 00124400
          " FROM SYSIBM.SYSSTOGRUP ", 00124500
          " WHERE CREATOR = 'SRCSCHEMA'" 00124608
"EXECSQL DECLARE C1 CURSOR FOR S1" 00124700
"EXECSQL PREPARE S1 INTO :OUTREC FROM :SQLSTMT" 00124800
IF VERBOSE = Y THEN 00124903
    SAY "SQLCODE from Select PREPARE is "SQLCODE 00125004
00126004
/*****/ 00127004
/* Setup and PREPARE INSERT statement for ZMCOB_STOGRUP */ 00127104
/*****/ 00127304
00127404
ISRT_STMT = "INSERT INTO "SRCSCHEMA".ZMCOB_STOGRUP (", 00128007
            "SRCSTOGRUP, ", 00129001
            "SRCVCAT, ", 00130000
            "TRGSTOGRUP, ", 00131000
            "TRGVCAT ", 00132000
            ") ", 00160000
            "VALUES (?, ?, ?, ?)" 00170000
"EXECSQL PREPARE S2 FROM :ISRT_STMT" 00180000
IF VERBOSE = Y THEN 00181003
    SAY "SQLCODE from Insert PREPARE is "SQLCODE 00190004
00193000
/*****/ 00194000
/* Open Cursors for Fetch */ 00195000
"EXECSQL OPEN C1" 00196000
IF VERBOSE = Y THEN 00196103
    SAY "SQLCODE from Select OPEN is "SQLCODE 00197004
00198004
/*****/ 00199004
/* Open Cursor for reading through SYSSTOGRUP and fetch */ 00199104
/* the first row. */ 00199204
/*****/ 00199304
00199404

```

```

"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "           00200000
IF VERBOSE = Y THEN                                   00201003
  SAY "SQLCODE from select FETCH is "SQLCODE          00210004
                                                       00220004
/*****/                                               00230004
/* Set loop counter and loop while rows are found     */ 00231004
/*****/                                               00232004
                                                       00233004

loopctr = 1                                           00240001
DO WHILE SQLCODE = 0                                  00250000
                                                       00260004
/*****/                                               00261004
/* Setup variables from SYSSTOGRROUP fetch for inserting, */ 00262004
/* including replacing the first 3 chars of the STOGRROUP name. */ 00263004
/*****/                                               00264004
                                                       00265004
SRCSTOGRROUP = OUTREC.1.SQLDATA                       00270000
TRGSTOGRROUP = TRGSTOGRPPRE||substr(OUTREC.1.SQLDATA,4,3) 00271001
SRCVCAT      = OUTREC.2.SQLDATA                       00280000
                                                       00290004
/*****/                                               00300004
/* Execute insert statement, and check RC             */ 00310004
/*****/                                               00320004
                                                       00330004
"EXECSQL EXECUTE S2 USING ",                          00373000
  ":SRCSTOGRROUP, ",                                  00374000
  ":SRCVCAT, ",                                       00375000
  ":TRGSTOGRROUP, ",                                  00376000
  ":TRGVCAT "                                         00379700
IF VERBOSE = Y THEN                                   00379803
  SAY "SQLCODE from Insert is "SQLCODE               00379904
                                                       00381000
/*****/                                               00381204
/* Fetch next SYSTABLESPACE record, increment counter */ 00381304
/*****/                                               00381404
                                                       00381504
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "         00382400
IF VERBOSE = Y THEN                                   00382503
  SAY "SQLCODE from FETCH is "SQLCODE                00382604
                                                       00382704
loopctr=loopctr+1                                    00382804
                                                       00382900
END                                                    00383000
                                                       00383100
/*****/                                               00383304
/* Close main cursor and exit                        */ 00383404
/*****/                                               00383504
                                                       00383604
SAY "-----"                                         00383703

```

SAY "Total stogroups processed "loopctr	00383803
SAY "-----"	00383903
"EXECSQL CLOSE C1"	00384000
SAY"SQLCODE from CLOSE is "SQLCODE	00384100
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')	00385000

SETUPDB

In Example B-8, we provide the REXX procedure used to populate the metadata table ZMCOB_DATABASE.

Example: B-8 REXX to populate ZMCOB_DATABASE

```

/*****/ 00010014
/* DB2 INITIALISATION - Populate MCOB Database Table */ 00020014
/* VERBOSE - Print all SQL Return codes */ 00030014
/* PREDELETE - Delete table contents prior to run, allow reruns */ 00040014
/*****/ 00040114
00040220
/*****/ 00040422
/* */ 00040522
/* Initialize variables passed to program */ 00040622
/* Name Description */ 00040722
/* SRCSSID Source DB2 Subsystem ID */ 00040822
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00040922
/* SRCRELEASE Source DB2 Version Number */ 00041022
/* TRGSSID Target DB2 Subsystem ID */ 00041122
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00041222
/* TRGRELEASE Target DB2 Version Number */ 00041322
/* TRGVCAT Target DB2 VCAT name */ 00041422
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00041522
/* */ 00041622
/*****/ 00041722
00041822
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00041922
          TRGSSID TRGSCHEMA TRGRELEASE, 00042022
          TRGVCAT TRGSTOGRPPRE 00042122
00042222
VERBOSE = N 00042325
PREDELETE = Y 00042422
00043022
/* Add DSNREXX to the host command environment table if not there. */ 00050000
00051009
'SUBCOM DSNREXX' 00060000
IF RC THEN , 00070000
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX') 00080000
ADDRESS DSNREXX 00090000
00100014

```

```

/*****/ 00101014
/* Connect to DB2 */ 00102014
/*****/ 00103014
00104014
'CONNECT' SRCSSID 00110018
/* trace i */ 00120008
00120114
/*****/ 00125314
/* For rerunning this procedure, to prevent duplicates, set flag */ 00125414
/* PREDELETE to delete table contents before populating table */ 00125514
/*****/ 00125614
00125714
IF PREDELETE = Y THEN DO 00126113
    "EXECSQL DELETE FROM "SRCSCHEMA".ZMCOB_DATABASE" 00127018
    Say "SQL from delete is "SQLCODE 00128013
END 00129013
00130014
/*****/ 00140014
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00141014
/* This cursor remains open and loops through SYSDATABASE rows */ 00142014
/*****/ 00143014
00144014
SQLSTMT = "SELECT NAME, STGROUP, CREATOR ", 00150008
        " FROM SYSIBM.SYSDATABASE ", 00160008
        " WHERE CREATOR = 'SRCSCHEMA'" " 00170018
"EXECSQL DECLARE C1 CURSOR FOR S1" 00190002
"EXECSQL PREPARE S1 INTO :OUTREC FROM :SQLSTMT" 00200006
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00201024
    SAY "SQLCODE from Select PREPARE is "SQLCODE 00210014
    00211014
/*****/ 00212014
/* Setup and PREPARE INSERT statement for ZMCOB_DATABASE */ 00213014
/*****/ 00214014
00215014
ISRT_STMT = " INSERT INTO "SRCSCHEMA".ZMCOB_DATABASE ", 00220018
        "(SRCSSID, SRCRELEASE, SRCDBNAME, ", 00221008
        "SRCSCHEMA, SRCSTOGROUP, ", 00221108
        "TRGSSID, TRGRELEASE, TRGDBNAME, ", 00221208
        "TRGSHEMA, TRGSTOGROUP) ", 00221308
        "VALUES (?,?,?,?,?,?,?,?)" 00223008
"EXECSQL PREPARE S2 FROM :ISRT_STMT" 00225008
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00225124
    SAY "SQLCODE from Insert PREPARE is "SQLCODE 00226014
    00227014
/*****/ 00228014
/* Open Cursor for reading through SYSDATABASE and fetch */ 00229014
/* the first row. */ 00230014
/*****/ 00240014
00241014

```

```

"EXECSQL OPEN C1" 00250000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00251024
  SAY "SQLCODE from Select OPEN is "SQLCODE 00260014
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00300006
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00301024
  SAY "SQLCODE from select FETCH is "SQLCODE 00310014
  00320014
/*****/ 00330014
/* Set loop counter and loop while rows are found */ 00331014
/*****/ 00332014
00333014
loopctr = 0 00340000
DO WHILE SQLCODE = 0 00350000
  00360014
/*****/ 00361014
/* Setup variables from SYSDATABASE fetch for inserting, */ 00361114
/* including replacing the first 3 chars of the STOGROUP name. */ 00361214
/*****/ 00361314
  00361414
  SRCDBNAME = OUTREC.1.SQDATA 00362008
  SRCSTOGROUP = OUTREC.2.SQDATA 00363008
  SRCSCHEMA = OUTREC.3.SQDATA 00364008
  TRGDBNAME = OUTREC.1.SQDATA 00365008
  IF LENGTH(STRIIP(OUTREC.2.SQDATA)) = 6 THEN 00365223
    TRGSTOGROUP = TRGSTOGRPPRE||substr(OUTREC.2.SQDATA,4,3) 00366022
  ELSE 00366122
    TRGSTOGROUP = OUTREC.2.SQDATA 00366222
  00367014
/*****/ 00368014
/* Execute insert statement, and check RC */ 00369014
/*****/ 00370014
  00380014
  "EXECSQL EXECUTE S2 USING ", 00384308
    ":SRCSSID, :SRCRELEASE, :SRCDBNAME, ", 00384408
    ":SRCSCHEMA, :SRCSTOGROUP, ", 00384508
    ":TRGSSID, :TRGRELEASE, :TRGDBNAME, ", 00384608
    ":TRGVCAT, :TRGSTOGROUP " 00384721
  IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00384824
    SAY "SQLCODE from Insert is "SQLCODE 00384914
  00385014
/*****/ 00386014
/* Fetch next SYSDATABASE record, increment counter */ 00387014
/*****/ 00388014
  00389014
  "EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00441007
  IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00442024
    SAY "SQLCODE from FETCH is "SQLCODE 00460014
  00470000
  loopctr=loopctr+1 00471014

```

```

END 00472014
00472114
00472214
/*****/ 00473014
/* Close main cursor and exit */ 00474014
/*****/ 00475014
00476014
SAY "-----" 00481013
SAY "Total databases processed "loopctr 00482013
SAY "-----" 00490013
"EXECSQL CLOSE C1" 00510000
SAY"SQLCODE from CLOSE is "SQLCODE 00520000
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX') 00530000

```

SETUPTS

In Example B-9, we provide the REXX procedure used to populate the metadata table ZM COD_TABLESPACE.

Example: B-9 REXX to populate ZM COD_TABLESPACE

```

/*****/ 00010010
/* DB2 INITIALISATION - Populate M COD Tablespace Table */ 00020010
/* VERBOSE - Print all SQL Return codes */ 00021010
/* PREDELETE - Delete table contents prior to run, allow reruns */ 00022010
/*****/ 00030010
00031024
VERBOSE = N 00031125
PREDELETE = Y 00031225
00031325
/*****/ 00031425
/* */ 00031525
/* Initialize variables passed to program */ 00031625
/* Name Description */ 00031725
/* SRCSSID Source DB2 Subsystem ID */ 00031825
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00031925
/* SRCRELEASE Source DB2 Version Number */ 00032025
/* TRGSSID Target DB2 Subsystem ID */ 00032125
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00032225
/* TRGRELEASE Target DB2 Version Number */ 00032325
/* TRGV CAT Target DB2 VCAT name */ 00032425
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00032525
/* */ 00032625
/*****/ 00032725
00032825
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00032922
TRGSSID TRGSCHEMA TRGRELEASE, 00033022
TRGV CAT TRGSTOGRPPRE 00033122

```

```

/*****/ 00033222
/* Add DSNREXX to the host command environment table if not there. */ 00033325
/*****/ 00050000
'SUBCOM DSNREXX' 00051025
IF RC THEN , 00060000
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX') 00070000
ADDRESS DSNREXX 00080000
00090000
/*****/ 00100000
/* Connect to DB2 */ 00110000
/*****/ 00120006
'CONNECT' SRCSSID 00130006
/* trace i */ 00140006
/*****/ 00150006
/* Initialize hardcoded variable values */ 00160022
/*****/ 00170006
/* 00180006
SRCSSID = "DB7S" 00190006
SRCRELEASE = "710" 00200006
TRGSSID = "DB7T" 00210006
TRGRELEASE = "710" 00220022
TRGSCHEMA = "SAPBLU" 00230000
TRGSTOGRPPRE = "BLU" 00240002
TRGVCAT = "MCDBLU" 00250000
*/ 00260002
/*****/ 00270021
/* For rerunning this procedure, to prevent duplicates, set flag */ 00280021
/* PREDELETE to delete table contents before populating table */ 00290021
/*****/ 00302010
IF PREDELETE = Y THEN DO 00320210
    "EXECSQL DELETE FROM "SRCSHEMA".ZMCOB_TABLESPACE" 00320310
    Say "SQL from delete is "SQLCODE 00320410
END 00321010
00330008
00340022
00350008
00360008
00380006
/*****/ 00390006
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00400006
/* This cursor remains open and loops through SYSTABLESPACE rows */ 00410010
/*****/ 00420006
SQLSTMT = "SELECT ", 00430006
    "TS.NAME, ", 00440000
    "TS.CREATOR, ", 00450006
    "TS.DBNAME, ", 00460006
    "TS.DBID, ", 00470006
00480006

```



```

"TS.PSID,      ", 00490006
"TSP.STORNAME, ", 00500006
"TSP.VCATNAME, ", 00510006
"TSP.IPREFIX,  ", 00520006
"TSP.SPACE,    ", 00530009
"TSP.PARTITION ", 00531009
" FROM SYSIBM.SYSTABLESPACE TS JOIN SYSIBM.SYSTABLEPART TSP", 00540002
" ON TS.NAME = TSP.TSNAME AND TS.DBNAME = TSP.DBNAME ", 00550002
" WHERE TS.CREATOR = 'SRCSHEMA'" 00560022
"EXECSQL DECLARE C1 CURSOR FOR S1" 00570002
"EXECSQL PREPARE S1 INTO :OUTREC FROM :SQLSTMT" 00580002
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00581026
    SAY "SQLCODE from S1 PREPARE is "SQLCODE 00590010
00600000
/*****/ 00610006
/* Open Cursor for reading through SYSTABLESPACE and fetch */ 00620006
/* the first row. */ 00630006
/*****/ 00640006
00650006
"EXECSQL OPEN C1" 00660000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00661026
    SAY "SQLCODE from MAINLOOP FETCH OPEN is "SQLCODE 00670011
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00680000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00681026
    SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE 00690011
00700000
/*****/ 00710006
/* Set loop counter and loop while rows are found */ 00720006
/*****/ 00730006
00740006
loopctr = 0 00750011
DO WHILE SQLCODE = 0 00760000
00770000
/*****/ 00780006
/* Setup variables from SYSTABLESPACE fetch for inserting, */ 00790006
/* including replacing the first 3 chars of the STOGROUP name. */ 00800010
/*****/ 00810006
00820006
SRCTSNAME = OUTREC.1.SQLDATA 00830006
TRGTSNAME = OUTREC.1.SQLDATA 00840006
SRCSHEMA = OUTREC.2.SQLDATA 00850006
SRCDBNAME = OUTREC.3.SQLDATA 00860006
TRGDBNAME = OUTREC.3.SQLDATA 00870006
SRCDBID = OUTREC.4.SQLDATA 00880006
TRGDBID = OUTREC.4.SQLDATA 00890006
SRCPSID = OUTREC.5.SQLDATA 00900006
TRGPSID = OUTREC.5.SQLDATA 00910006
SRCSTOGROUP = OUTREC.6.SQLDATA 00920006
TRGSTOGROUP = TRGSTOGRPPRE | substr(OUTREC.6.SQLDATA,4,3) 00930006

```

SRCVCAT	=	OUTREC.7.SQLDATA	00940006
SRCIPREFIX	=	OUTREC.8.SQLDATA	00950006
TRGIPREFIX	=	OUTREC.8.SQLDATA	00960006
SRCSPACE	=	OUTREC.9.SQLDATA	00970006
SRCPART	=	OUTREC.10.SQLDATA	00971009
			00980005
/*****			00990006
/* Setup insert statement string			*/ 01000006
/*****			01010006
			01020006
INSERTSQL =	"INSERT INTO "SRCSHEMA".ZMCOB_TABLESPACE ("		01030022
	"SRCTSNAME",		01040002
	"SRCSSID",		01050002
	"SRCDBNAME",		01060002
	"SRCDBID",		01070002
	"SRCRELEASE",		01080002
	"SRCIPREFIX",		01090002
	"SRCPSID",		01100002
	"SRCSHEMA",		01110002
	"SRCSTOGRUP",		01120002
	"SRCVCAT",		01130002
	"SRCSPACE",		01140005
	"SRCPART",		01141009
	"TRGTSNAME",		01150005
	"TRGSSID",		01160005
	"TRGRELEASE",		01170005
	"TRGDBID",		01180005
	"TRGDBNAME",		01190005
	"TRGIPREFIX",		01200005
	"TRGPSID",		01210005
	"TRGSHEMA",		01220005
	"TRGSTOGRUP",		01230005
	"TRGVCAT",		01240005
	")",		01250005
	" VALUES('SRCTSNAME'",		01260005
	"SRCSSID'",		01270005
	"SRCDBNAME'",		01280005
	"SRCDBID'",		01290005
	"SRCRELEASE'",		01300005
	"SRCIPREFIX'",		01310005
	"SRCPSID'",		01320005
	"SRCSHEMA'",		01330005
	"SRCSTOGRUP'",		01340005
	"SRCVCAT'",		01350005
	"SRCSPACE'",		01360009
	"SRCPART'",		01361009
	"TRGTSNAME'",		01370005
	"TRGSSID'",		01380005
	"TRGRELEASE'",		01390005

SETUPTAB

In Example B-10, we provide the REXX procedure used to populate the metadata table ZMCOB_TABLES.

Example: B-10 REXX to populate ZMCOB_TABLES

```

/*****/ 00010000
/* DB2 INITIALISATION - Populate MCOB Tables Table */ 00020000
/*****/ 00030000
00040000
VERBOSE = N 00040112
PREDELETE = Y 00040212
00040312
/*****/ 00040412
/* Initialize variables passed to program */ 00040612
/* Name Description */ 00040712
/* SRCSSID Source DB2 Subsystem ID */ 00040812
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00040912
/* SRCRELEASE Source DB2 Version Number */ 00041012
/* TRGSSID Target DB2 Subsystem ID */ 00041112
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00041212
/* TRGRELEASE Target DB2 Version Number */ 00041312
/* TRGVCAT Target DB2 VCAT name */ 00041412
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00041512
/* */ 00041612
/*****/ 00041712
00041812
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00041909
          TRGSSID TRGSCHEMA TRGRELEASE, 00042009
          TRGVCAT TRGSTOGRPPRE 00042109
00042209
/*****/ 00042312
/* Add DSNREXX to the host command environment table if not there. */ 00050000
/*****/ 00051012
00060000
'SUBCOM DSNREXX' 00070000
IF RC THEN , 00080000
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX') 00090000
ADDRESS DSNREXX 00100000
00110000
/*Connect to DB2 */ 00120000
'CONNECT' SRCSSID 00130009
/* trace i 00140000
*/ 00150000
/* Initialize Variables 00160009
SRCSSID = "DB7S" 00170000
SRCRELEASE = "710" 00180004
TRGSSID = "DB7T" 00190000
```

```

TRGRELEASE = "710"                                00200004
TRGSHEMA = "SAPBLU"                                00202004
*/                                                    00210009
IF PREDELETE = 'Y' THEN DO                          00211007
  "EXECSQL DELETE FROM "SRCSHEMA".ZMCOB_TABLES"    00212009
  SAY "SQL from delete is "SQLCODE                 00213007
END                                                  00213107
                                                    00214006

/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00220000
SQLSTMT = "SELECT CREATOR,",                        00230004
          "NAME,",                                  00240004
          "DBNAME,",                                00250004
          "TSNAME,",                                00260004
          "OBID",                                   00270004
          " FROM SYSIBM.SYSTABLES ",                00280000
          " WHERE CREATOR = '"SRCSHEMA"' AND TYPE = 'T'" 00290009
"EXECSQL DECLARE C1 CURSOR FOR S1"                 00300000
"EXECSQL PREPARE S1 INTO :OUTREC FROM :SQLSTMT"    00310000
IF VERBOSE = Y THEN                                00311007
  SAY "SQLCODE from Select PREPARE is "SQLCODE     00320008
                                                    00330000
/*****/                                             00340000
/* Open Cursors for Fetch */                        00350000
"EXECSQL OPEN C1"                                  00360000
IF VERBOSE = Y THEN                                00361007
  SAY "SQLCODE from Select OPEN is "SQLCODE        00370008
                                                    00380000

/* Fetch the first record into host vars */         00390000
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "      00400000
IF VERBOSE = Y THEN                                00401007
  SAY "SQLCODE from select FETCH is "SQLCODE       00410008
/* While the fetches are OK, continue to loop */   00430000
loopctr = 0                                        00440000
DO WHILE SQLCODE = 0                               00450000
                                                    00460004
  SRCSHEMA = OUTREC.1.SQLDATA                       00470004
  SRCTABLE = OUTREC.2.SQLDATA                       00480004
  TRGTABLE = OUTREC.2.SQLDATA                       00490004
  SRCDBNAME = OUTREC.3.SQLDATA                      00500004
  TRGDBNAME = OUTREC.3.SQLDATA                      00510004
  SRCTSNAME = OUTREC.4.SQLDATA                      00520004
  TRGTSNAME = OUTREC.4.SQLDATA                     00530004
  SRCTABOBID = OUTREC.5.SQLDATA                    00540004
  TRGTABOBID = OUTREC.5.SQLDATA                    00550004
                                                    00560000

INSERTSQL = "INSERT INTO "SRCSHEMA".ZMCOB_TABLES (", 00570009
          ||"SRCSHEMA",",                          00580004
          ||"SRCTABLE",",                          00590004
          ||"SRCDBNAME",",                          00600004

```

"SRSSID",	00610004
"SRCRELEASE",	00620004
"SRCTABOBID",	00630004
"SRCTSNAME",	00640004
"TRGSCHEMA",	00650004
"TRGSSID",	00660004
"TRGRELEASE",	00670004
"TRGDBNAME",	00680004
"TRGTABOBID",	00690004
"TRGTSNAME",	00700004
") ",	00710004
" VALUES('SRCSHEMA'",	00720004
SRCTABLE'",	00721004
SRCDBNAME'",	00730004
SRCSSID'",	00750004
SRCRELEASE'",	00760004
SRCTABOBID'",	00770004
SRCTSNAME'",	00780004
TRGSCHEMA'",	00790004
TRGSSID'",	00800004
TRGRELEASE'",	00810004
TRGDBNAME'",	00820004
TRGTABOBID'",	00830004
TRGTSNAME') "	00840004
	00841004
"EXECSQL "INSERTSQL	00842004
IF VERBOSE = Y THEN	00843007
SAY loopctr "INSERTSQL is "SQLCODE	00850008
loopctr=loopctr+1	00910011
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "	00950000
IF VERBOSE = Y THEN	00951007
SAY "SQLCODE from FETCH is "SQLCODE	00960008
	00970000
END	00980000
SAY "-----"	00981007
SAY "Total tables processed "loopctr	00982007
SAY "-----"	00990007
/* Close Cursor */	01000000
"EXECSQL CLOSE C1"	01010000
SAY"SQLCODE from CLOSE is "SQLCODE	01020000
S_RC = RXSUBCOM('DELETE', 'DSNREXX', 'DSNREXX')	01030000

SETUPIX

In Example B-11, we provide the REXX procedure used to populate the metadata table ZMCOD_INDEXES.

Example: B-11 REXX to populate ZMCOD_INDEXES

```

/*****/ 00010011
/* DB2 INITIALISATION - Populate MCODE Index Table */ 00020011
/* VERBOSE - Print all SQL Return codes */ 00030011
/* PREDELETE - Delete table contents prior to run, allow reruns */ 00040011
/*****/ 00050011
00060020
VERBOSE = N 00070020
PREDELETE = Y 00080020
00090020
/*****/ 00100020
/* */ 00110020
/* Initialize variables passed to program */ 00120020
/* Name Description */ 00130020
/* SRCSSID Source DB2 Subsystem ID */ 00140020
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00150020
/* SRCRELEASE Source DB2 Version Number */ 00160020
/* TRGSSID Target DB2 Subsystem ID */ 00170020
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00180020
/* TRGRELEASE Target DB2 Version Number */ 00190020
/* TRGVCAT Target DB2 VCAT name */ 00200020
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00210020
/* */ 00220020
/*****/ 00230020
00240020
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00250019
TRGSSID TRGSCHEMA TRGRELEASE, 00260019
TRGVCAT TRGSTOGRPPRE 00270019
00280005
/*****/ 00290020
/* Add DSNREXX to the host command environment table if not there. */ 00300000
/*****/ 00310020
00320000
'SUBCOM DSNREXX' 00330000
IF RC THEN , 00340000
S_RC = RXSUBCOM('ADD', 'DSNREXX', 'DSNREXX') 00350000
ADDRESS DSNREXX 00360000
00370011
/*****/ 00380011
/* Connect to DB2 */ 00390011
/*****/ 00400011
00410011
'CONNECT' SRCSSID 00420020
```

```

/*****/ 00430011
/* For rerunning this procedure, to prevent duplicates, set flag */ 00440011
/* PREDELETE to delete table contents before populating table */ 00450011
/*****/ 00460011
IF PREDELETE = 'Y' THEN DO 00470011
  "EXECSQL DELETE FROM "SRCSCHEMA".ZMCOB_INDEXES" 00480011
  Say "SQL from delete is "SQLCODE 00490005
END 00500016
00510005
00520005
00530011
/*****/ 00540011
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00550011
/* This cursor remains open and loops through SYSINDEXES rows */ 00560011
/*****/ 00570011
SQLSTMT = "SELECT ", 00580011
          "IX.NAME,", 00590000
          "IX.CREATOR,", 00600000
          "IX.DBNAME,", 00610000
          "IX.DBID,", 00620000
          "IX.INDEXSPACE,", 00630000
          "IXPART.IPREFIX,", 00640000
          "IX.ISOBID,", 00650000
          "IXPART.PQTY,", 00660023
          "IXPART.SQTY,", 00670000
          "IXPART.STORNAME,", 00680000
          "IXPART.VCATNAME,", 00690000
          "IXPART.SPACE,", 00700003
          "IXPART.PARTITION ", 00710007
          " FROM SYSIBM.SYSINDEXES IX JOIN SYSIBM.SYSINDEXPART IXPART", 00720008
          " ON IX.NAME = IXPART.IXNAME AND IX.CREATOR = IXPART.IXCREATOR ", 00730000
          " WHERE IX.CREATOR = 'SRCSCHEMA'" 00740001
"EXECSQL DECLARE C1 CURSOR FOR S1" 00750016
"EXECSQL PREPARE S1 INTO :OUTREC FROM :SQLSTMT" 00760000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00770000
SAY "SQLCODE from Select PREPARE is "SQLCODE 00780020
/*****/ 00790000
/* Open Cursor for reading through SYSINDEXES and fetch */ 00800011
/* the first row. */ 00810011
/*****/ 00820011
"EXECSQL OPEN C1" 00830011
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00840011
SAY "SQLCODE from Select OPEN is "SQLCODE 00850011
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00860000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00870020
SAY "SQLCODE from select FETCH is "SQLCODE 00880000
00890000
00900020
00910011

```



```

00920011
/*****/ 00930011
/* Set loop counter and loop while rows are found */ 00940011
/*****/ 00950011
00960011
loopctr = 0 00970000
DO WHILE SQLCODE = 0 00980000
00990011
/*****/ 01000011
/* Setup variables from SYSINDEXES fetch for inserting, */ 01010011
/* including replacing the first 3 chars of the STOGROUP name. */ 01020011
/*****/ 01030011
01040011
SRCINDEX = OUTREC.1.SQDATA 01050000
TRGINDEX = OUTREC.1.SQDATA 01060000
SRCSHEMA = OUTREC.2.SQDATA 01070000
SRCDBNAME = OUTREC.3.SQDATA 01080000
TRGDBNAME = OUTREC.3.SQDATA 01090000
SRCDBID = OUTREC.4.SQDATA 01100000
TRGDBID = OUTREC.4.SQDATA 01110000
SRCINDEXSPACE = OUTREC.5.SQDATA 01120000
TRGINDEXSPACE = OUTREC.5.SQDATA 01130000
SRCIPREFIX = OUTREC.6.SQDATA 01140000
TRGIPREFIX = OUTREC.6.SQDATA 01150000
SRCISOBID = OUTREC.7.SQDATA 01160000
TRGISOBID = OUTREC.7.SQDATA 01170000
SRCPRIQTY = OUTREC.8.SQDATA 01180000
SRCSTOGROUP = OUTREC.10.SQDATA 01190000
TRGSTOGROUP = TRGSTOGRPPRE || substr(OUTREC.10.SQDATA,4,3) 01200001
SRCVCAT = OUTREC.11.SQDATA 01210000
SRCSPACE = OUTREC.12.SQDATA 01220003
SRCPART = OUTREC.13.SQDATA 01230009
01240011
/*****/ 01250011
/* Setup insert statement string */ 01260011
/*****/ 01270011
01280011
INSERTSQL = "INSERT INTO "SRCSHEMA".ZMCOB_INDEXES (" 01290016
          || "SRCINDEX," 01300001
          || "SRCSHEMA," 01310001
          || "SRCDBNAME," 01320001
          || "SRCPART," 01330009
          || "SRCSSID," 01340001
          || "SRCRELEASE," 01350001
          || "SRCDBID," 01360001
          || "SRCINDEXSPACE," 01370001
          || "SRCIPREFIX," 01380001
          || "SRCISOBID," 01390001
          || "SRCPRIQTY," 01400001

```

```

| "SRCSTOGROUP", " 01410001
| "SRCVCAT", " 01420001
| "SRCSPACE", " 01430003
| "TRGSSID", " 01440007
| "TRGRELEASE", " 01450007
| "TRGDBID", " 01460007
| "TRGDBNAME", " 01470007
| "TRGINDEX", " 01480007
| "TRGINDEXSPACE", " 01490007
| "TRGIPREFIX", " 01500007
| "TRGISOBID", " 01510007
| "TRGSCHEMA", " 01520007
| "TRGSTOGROUP", " 01530007
| "TRGVCAT", " 01540007
| ") ", 01550000
" VALUES(' ", 01560001
| SRCINDEX", " ', " 01570001
| SRCSCHEMA", " ', " 01580001
| SRCDATABASE", " ', " 01590009
| SRCPART", " ', " 01600009
| SRCSSID", " ', " 01610001
| SRCRELEASE", " ', " 01620001
| SRCDBID", " ', " 01630001
| SRCINDEXSPACE", " ', " 01640001
| SRCIPREFIX", " ', " 01650001
| SRCISOBID", " ', " 01660001
| SRCPRIQTY", " ', " 01670001
| SRCSTOGROUP", " ', " 01680001
| SRCVCAT", " ', " 01690003
| SRCSPACE", " ', " 01700009
| TRGSSID", " ', " 01710007
| TRGRELEASE", " ', " 01720007
| TRGDBID", " ', " 01730007
| TRGDBNAME", " ', " 01740007
| TRGINDEX", " ', " 01750007
| TRGINDEXSPACE", " ', " 01760007
| TRGIPREFIX", " ', " 01770007
| TRGISOBID", " ', " 01780007
| TRGSCHEMA", " ', " 01790007
| TRGSTOGROUP", " ', " 01800001
| TRGVCAT", " ') " 01810001
/***** 01830011
/* Execute insert statement, and check RC */ 01840011
/***** 01850011
"EXECSQL "INSERTSQL 01870001
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01880020
SAY "INSERTSQL is "SQLCODE 01890011
01900011

```

```

/*****/ 01910011
/* Fetch next SYSTABLESPACE record, increment counter */ 01920011
/*****/ 01930011

      "EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 01940011
      IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01950011
        SAY "SQLCODE from FETCH is "SQLCODE 01960020
        SAY "SQLCODE from FETCH is "SQLCODE 01970011
        SAY "SQLCODE from FETCH is "SQLCODE 01980011

      loopctr=loopctr+1 01990011
                                02000000
END 02010000
                                02020000

/*****/ 02030011
/* Close main cursor and exit */ 02040011
/*****/ 02050011

SAY "-----" 02060011
SAY "Total indexes processed "loopctr 02070005
SAY "-----" 02080005
"EXECSQL CLOSE C1" 02090005
SAY"SQLCODE from CLOSE is "SQLCODE 02100011
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX') 02110000
                                02120000

```

JCL to run the population REXX procedures

In Example B-12, we provide the JCL used to run the population REXX procedures on the source subsystem.

Example: B-12 JCL to run population REXX procedures

```

//SAPRES2D JOB (999,TIM),'POPULATE',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,REGION=0M
/*JOBPARM SYSAFF=SC42,L=9999
//*
//RUNREXX EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DB7S7.SDSNEXIT,DISP=SHR
// DD DSN=DB7S7.SDSNLOAD,DISP=SHR
//SYSEXEC DD DISP=SHR,DSN=SAPRES2.MCOD.REXX
//OUTDDDB DD SYSOUT=*
//OUTDDTS DD SYSOUT=*
//OUTDDTB DD SYSOUT=*
//OUTDDIX DD SYSOUT=*
//OUTDDRE DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%SETUPSTG DB7S SAPBLU 710 DB7T SAPBLU 710 MCDBLU BLU
%SETUPDB DB7S SAPBLU 710 DB7T SAPBLU 710 MCDBLU BLU
%SETUPTS DB7S SAPBLU 710 DB7T SAPBLU 710 MCDBLU BLU

```

```

%SETUPTAB DB7S SAPBLU 710 DB7T SAPBLU 710 MCDBLU BLU
%SETUPIX DB7S SAPBLU 710 DB7T SAPBLU 710 MCDBLU BLU
/*
/** For faster elapsed time, this job can be split into
/** 5 jobs running each REXX, executing in parallel

```

Moving metadata tables across systems

Once the metadata tables have been populated with the data from the source system, they have to be transferred to the target system using the UNLOAD and LOAD utilities.

JCL to unload metadata tables on the source system

In Example B-13, we provide the JCL to unload the metadata tables on the source system.

Example: B-13 JCL used to unload metadata tables

```

//SAPRES2U JOB (999,TIM), 'DB2UNLOD ', CLASS=A, MSGCLASS=T,
// NOTIFY=&SYSUID, TIME=1440, REGION=OM , RESTART=HERE
/*JOBPARM SYSAFF=SC42, L=9999
// JCLLIB ORDER=(DB7SU.PROCLIB)
/**
/** ENSURE YOU TYPE IN THE CORRECT DATABASE.TABLESPACE NAMES
/**
/** NAME OWNER T DB NAME TS NAME
/** ZMCOB_STOGROUP SAPR3 T U000X42M ZMCOBXS
/** ZMCOB_TABLESPACE SAPR3 T U020X6NA ZMCOBXTX
/** ZMCOB_TABLES SAPR3 T U010X16F ZMCOBXT
/** ZMCOB_INDEXES SAPR3 T U010XFYM ZMCOBXIX
/** ZMCOB_DATABASE SAPR3 T U010X6P3 ZMCOBXDX
/**
//HERE EXEC PGM=IEFBR14
//DSNTIC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DELETE (SAPRES4.UNLOAD.SYSREC.STOG)
DELETE (SAPRES4.UNLOAD.SYSPUNCH.STOG)
DELETE (SAPRES4.UNLOAD.SYSREC.DATABASE)
DELETE (SAPRES4.UNLOAD.SYSPUNCH.DATABASE)
DELETE (SAPRES4.UNLOAD.SYSREC.TSPACE)
DELETE (SAPRES4.UNLOAD.SYSPUNCH.TSPACE)
DELETE (SAPRES4.UNLOAD.SYSREC.TABLES)
DELETE (SAPRES4.UNLOAD.SYSPUNCH.TABLES)

```

```

DELETE (SAPRES4.UNLOAD.SYSREC.INDEXES)
DELETE (SAPRES4.UNLOAD.SYSPUNCH.INDEXES)
SET MAXCC=0

/*
//STEPST EXEC DSNUPROC,PARM='DB7S,DSNTEX'
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.STOG,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPUNCH DD DSN=SAPRES4.UNLOAD.SYSPUNCH.STOG,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE U000XGSR.ZMCOXDS
/*
//STEPDB EXEC DSNUPROC,PARM='DB7S,DSNTEX',COND=(0,LT)
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.DATABASE,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPUNCH DD DSN=SAPRES4.UNLOAD.SYSPUNCH.DATABASE,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE U010XZSZ.ZMCOXD

//STEPTS EXEC DSNUPROC,PARM='DB7S,DSNTEX'
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.TSPACE,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPUNCH DD DSN=SAPRES4.UNLOAD.SYSPUNCH.TSPACE,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE U020XTAF.ZMCOXDT
/*
//STEPTB EXEC DSNUPROC,PARM='DB7S,DSNTEX',COND=(0,LT)
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.TABLES,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPUNCH DD DSN=SAPRES4.UNLOAD.SYSPUNCH.TABLES,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE U010XX1B.ZMCOXDT
/*
//STAPIX EXEC DSNUPROC,PARM='DB7S,DSNTEX',COND=(0,LT)
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.INDEXES,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPUNCH DD DSN=SAPRES4.UNLOAD.SYSPUNCH.INDEXES,
// DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE U010XZ8Y.ZMCOXDI

```

JCL to load metadata tables into the target system

In Example B-14, we provide the JCL to load the metadata tables into the target system.

Example: B-14 JCL to load metadata tables into the target system

```
//SAPRES2A JOB (999,P0K),'DB2LOAD ',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM
//*JOBPARM SYSAFF=SC49,L=9999
// JCLLIB ORDER=(DB7TU.PROCLIB)
/**
/** NAME          OWNER    T DB NAME  TS NAME
/** ZMCOB_STOGROUP  SAPR3    T U000X42M ZMCOBXS
/** ZMCOB_TABLESPACE SAPR3    T U020X8AV ZMCOBXTX
/** ZMCOB_TABLES    SAPR3    T U010X16F ZMCOBXT
/** ZMCOB_INDEXES   SAPR3    T U010XT5B ZMCOBXIX
/** ZMCOB_DATABASE  SAPR3    T U010X6P3 ZMCOBXDX
/**
//HERE EXEC PGM=IEFBR14
/**
/** SYSREC CONTAINS THE DATA
/** SYSPUNCH CONTAINS THE LOAD STATEMENTS
/** - ASSUMES TABLE IS EMPTY, OTHERWISE ALTER SYSPUNCH TO BE
/** RESUME NO REPLACE YES
//STEP01 EXEC DSNUPROC,PARM='DB7T,DSNTEX'
//SYSPRINT DD SYSOUT=*
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.STOG,DISP=SHR
//SYSIN DD DSN=SAPRES4.UNLOAD.SYSPUNCH.STOG,DISP=SHR
//SYSUT1 DD DSN=SAPRES4.UNLOAD.SYSUT1.STOG,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD DSN=SAPRES4.UNLOAD.SORTOUT.STOG,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
/**
//STEP02 EXEC DSNUPROC,PARM='DB7T,DSNTEX'
//SYSPRINT DD SYSOUT=*
//SYSREC DD DSN=SAPRES4.UNLOAD.SYSREC.DATABASE,DISP=SHR
//SYSIN DD DSN=SAPRES4.UNLOAD.SYSPUNCH.DATABASE,DISP=SHR
//SYSUT1 DD DSN=SAPRES4.UNLOAD.SYSUT1.DATABASE,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
//SORTOUT DD DSN=SAPRES4.UNLOAD.SORTOUT.DATABASE,
// DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(4000,(20,20),,,ROUND)
/**
//STEP03 EXEC DSNUPROC,PARM='DB7T,DSNTEX'
//SYSPRINT DD SYSOUT=*
```



```

//SYSPRINT DD  SYSOUT=*
//SYSIN      DD  *
REPAIR SET    TABLESPACE U000XY9N.ZMCOXDS    NOCOPYPEND
REPAIR SET    TABLESPACE U010XVUY.ZMCOXDX    NOCOPYPEND
REPAIR SET    TABLESPACE U010X08U.ZMCOXIX    NOCOPYPEND
REPAIR SET    TABLESPACE U010X2K9.ZMCOXTX    NOCOPYPEND
REPAIR SET    TABLESPACE U020XPC1.ZMCOXT     NOCOPYPEND
/*

```

Working with metadata tables

After the metadata extracted from the source system has been transferred to the target system, it is then necessary to detect and correct any clashes that exist in database names between the two systems.

DUPLICDB

In Example B-16, we provide the REXX procedure used to detect and correct duplicate database names.

Example: B-16 REXX procedure to detect and correct duplicate database names

```

/*****/ 00010003
/* SAP MCOB Migration - Change DBNAME if duplicate exists */ 00020003
/*****/ 00030003
00040017
VERBOSE = N 00050017
00060018
/*****/ 00061016
/* */ 00062016
/* Initialize variables passed to program */ 00063016
/* Name Description */ 00064016
/* SRCSSID Source DB2 Subsystem ID */ 00065016
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00066016
/* SRCRELEASE Source DB2 Version Number */ 00067016
/* TRGSSID Target DB2 Subsystem ID */ 00068016
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00069016
/* TRGRELEASE Target DB2 Version Number */ 00069116
/* TRGVCAT Target DB2 VCAT name */ 00069216
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00069316
/* TRGOWNER Target DB2 Owner of SAP Objects */ 00069417
/* */ 00069517
/*****/ 00069617
00069717
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00069817
TRGSSID TRGSCHEMA TRGRELEASE, 00069917

```



```

TRGVCAT TRGSTOGRPPRE TRGOWNER                                00070017
                                                            00070117
/*****/                                                       00070217
/* Add DSNREXX to the host command environment table if not there. */ 00071001
/*****/                                                       00072016
                                                            00080001
'SUBCOM DSNREXX'                                             00090001
IF RC THEN ,                                                00100001
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')              00110001
ADDRESS DSNREXX                                             00120001
                                                            00130003
/*****/                                                       00140003
/* Connect to DB2 */                                         00150003
/*****/                                                       00160003
                                                            00170003
'CONNECT' TRGSSID                                           00180016
/* trace i */                                               00190003
                                                            00200003
/*****/                                                       00210003
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00220003
/* This cursor remains open and loops through ZMCOB_DATABASE rows */ 00230003
/*****/                                                       00240003
                                                            00250003
MAINLOOP = "SELECT TRGDBNAME ",                               00260003
           "FROM   "TRGOWNER".ZMCOB_DATABASE ",               00270016
           "WHERE  TRGSHEMA = '"TRGSHEMA'" "                 00280016
"EXECSQL DECLARE C1 CURSOR FOR S1"                           00290001
IF (VERBOSE = Y) | (SQLCODE < 0) THEN                        00291018
    SAY "SQLCODE from MAINLOOP DECLARE is "SQLCODE           00292017
"EXECSQL PREPARE S1 INTO :OUTREC FROM :MAINLOOP"              00300003
IF (VERBOSE = Y) | (SQLCODE < 0) THEN                        00301018
    SAY "SQLCODE from MAINLOOP PREPARE is "SQLCODE           00302017
                                                            00310001
/*****/                                                       00320003
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00330003
/* This cursor is opened to read SYSDATABASE looking for clashes */ 00340003
/*****/                                                       00350003
                                                            00360003
SYSDBASESQL = "SELECT CREATOR FROM SYSIBM.SYSDATABASE WHERE ", 00370003
              ||"NAME = ?"                                    00380001
"EXECSQL DECLARE C2 CURSOR FOR S2"                           00390001
IF (VERBOSE = Y) | (SQLCODE < 0) THEN                        00391018
    SAY "SQLCODE from SYSDBASESQL DECLARE is "SQLCODE        00392016
"EXECSQL PREPARE S2 INTO :OUTDB FROM :SYSDBASESQL"           00400003
IF (VERBOSE = Y) | (SQLCODE < 0) THEN                        00410018
    SAY "SQLCODE from SYSDBASESQL PREPARE is "SQLCODE        00420016
                                                            00430003
/*****/                                                       00440003
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00450003

```

```

/* This cursor is opened when database name is re-randomized, to */ 00460003
/* look for clashes within the new database names */ 00470003
/*****/ 00480003
00490003
ZMCOBSQL = "SELECT TRGDBNAME FROM "TRGOWNER".ZMCOB_DATABASE WHERE ", 00500016
          ||"TRGDBNAME = ?" 00510003
"EXECBSQL DECLARE C3 CURSOR FOR S3" 00520003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00521018
  SAY "SQLCODE from ZMCOBSQL DECLARE is "SQLCODE 00522016
"EXECBSQL PREPARE S3 INTO :DUPDB FROM :ZMCOBSQL" 00530003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00540018
  SAY "SQLCODE from ZMCOBSQL PREPARE is "SQLCODE 00550016
00560003
/*****/ 00570003
/* Open Cursor for reading through ZMCOB_DATABASE and fetch */ 00580003
/* the first row. */ 00590003
/*****/ 00600003
00610003
"EXECBSQL OPEN C1" 00620001
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00630016
  SAY "SQLCODE from MAINLOOP OPEN is "SQLCODE 00640016
"EXECBSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00650001
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00660016
  SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE 00670016
00680003
/*****/ 00690003
/* Set loop counter and loop while rows are found */ 00700003
/*****/ 00710003
00720003
countok = 0 00730005
countchanges = 0 00740005
countunresolved = 0 00750005
counttotal = 0 00760005
DO WHILE SQLCODE = 0 00770001
00780001
  IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00790016
    SAY counttotal "Fetched DB is "OUTREC.1.SQLDATA 00800016
    TRGDBNAME = OUTREC.1.SQLDATA 00810002
00820003
/*****/ 00830003
/* Open cursor in SYSDATABASE and check if row exists */ 00840003
/*****/ 00850003
00860003
"EXECBSQL OPEN C2 USING :TRGDBNAME" 00870002
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00880016
  SAY "SQLCODE from SYSDATABASE OPEN is "SQLCODE 00890016
"EXECBSQL FETCH C2 INTO :OUTREC2" 00900001
IF VERBOSE = Y THEN 00910018
  SAY "SQLCODE from SYSDATABASE Fetch is "SQLCODE 00920016

```

```

TEMPSQLCODE = SQLCODE                                00930003
"EXECSQL CLOSE C2"                                  00940003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN              00950016
    SAY "SQLCODE from ZTABCHK CLOSE is "SQLCODE      00960016
SQLCODE = TEMPSQLCODE                                00970003
00980003
/*****/                                              00990003
/* If no row exists, drop into main loop             */ 01000003
/*****/                                              01010003
01020003
    If SQLCODE = 100 THEN DO                          01030002
        countok = countok + 1                        01040005
        IF VERBOSE = Y THEN                          01050018
            SAY " Database " OUTREC.1.SQLDATA " No Dup "counttotal 01060016
        END                                           01070002
01080003
/*****/                                              01090003
/* If row exists, DBNAME must be re-randomized and checked for */ 01100003
/* uniqueness in SYSDATABASE and ZMCOD_DATABASE-TRGDDBNAME */ 01110003
/*****/                                              01120003
01130003
    ELSE do                                           01140001
        loopctr2 = 0                                  01150001
        do while SQLCODE = 0                          01160001
            loopctr2 = loopctr2 + 1                    01170001
01180003
/*****/                                              01190003
/* If we could not get unique name after 20 tries, drop out and */ 01200003
/* print error message for customer to investigate */ 01210003
/*****/                                              01220003
01230003
        if loopctr2 = 20 then do                       01240003
            SAY "Could not produce random for " TRGDDBNAME 01250016
            countunresolved = countunresolved + 1      01260005
            leave                                       01270001
            end                                         01280001
        else nop                                       01290001
01300003
/*****/                                              01310003
/* Randomize the last 3 characters to new values */ 01320003
/* Ugly code, but it'll do the job on the few occasions that clash*/ 01330003
/* True randomizing of all 3 characters */ 01340003
/*****/                                              01350003
01360003
        NEWDBSTART = SUBSTR(TRGDDBNAME,1,5)           01370002
        RANDOMCHAR = RANDOM(1,36)                     01380003
        If RANDOMCHAR <10 then RANDOMCHAR = RANDOMCHAR + 192 01390003
            else If RANDOMCHAR <19 then RANDOMCHAR = RANDOMCHAR + 199 01400003
            else If RANDOMCHAR <27 then RANDOMCHAR = RANDOMCHAR + 207 01410003

```

```

else If RANDOMCHAR <37 then RANDOMCHAR = RANDOMCHAR + 213 01420003
TRGDBPOS6 = D2C(RANDOMCHAR) 01430003
RANDOMCHAR = RANDOM(1,36) 01440003
If RANDOMCHAR <10 then RANDOMCHAR = RANDOMCHAR + 192 01450003
  else If RANDOMCHAR <19 then RANDOMCHAR = RANDOMCHAR + 199 01460003
  else If RANDOMCHAR <27 then RANDOMCHAR = RANDOMCHAR + 207 01470003
  else If RANDOMCHAR <37 then RANDOMCHAR = RANDOMCHAR + 213 01480003
TRGDBPOS7 = D2C(RANDOMCHAR) 01490003
RANDOMCHAR = RANDOM(1,36) 01500003
If RANDOMCHAR <10 then RANDOMCHAR = RANDOMCHAR + 192 01510003
  else If RANDOMCHAR <19 then RANDOMCHAR = RANDOMCHAR + 199 01520003
  else If RANDOMCHAR <27 then RANDOMCHAR = RANDOMCHAR + 207 01530003
  else If RANDOMCHAR <37 then RANDOMCHAR = RANDOMCHAR + 213 01540003
TRGDBPOS8 = D2C(RANDOMCHAR) 01550003
01560003
/*****/ 01570003
/* Build new DBNAME open cursor on SYSDATABASE and test it */ 01580003
/*****/ 01590003
01600003
NEWDBNAME = NEWDBSTART||TRGDBPOS6||TRGDBPOS7||TRGDBPOS8 01610003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01620016
  SAY TRGDBNAME" now "NEWDBNAME 01630017
  "EXECSQL OPEN C2 USING :NEWDBNAME" 01640016
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01650016
  SAY "SQLCODE from SYSDATABASES Open is "SQLCODE 01660016
  "EXECSQL FETCH C2 INTO :OUTREC2" 01670016
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01680016
  SAY "SQLCODE from SYSDATABASES is "SQLCODE 01690016
TEMPSQLCODE = SQLCODE 01700016
"EXECSQL CLOSE C2" 01710016
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01720016
  SAY "SQLCODE from ZTABCHK CLOSE is "SQLCODE 01730016
  SQLCODE = TEMPSQLCODE 01740016
01750003
/*****/ 01760003
/* If it is unique in SYSDATABASE, need to test it against our */ 01770003
/* ZMCO_DATABASE-TRGDBNAME as well, in case we already used it */ 01780003
/*****/ 01790003
01800003
IF SQLCODE = 100 then do 01810003
  "EXECSQL OPEN C3 USING :NEWDBNAME" 01820003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01830016
  SAY "SQLCODE from ZTABCHK OPEN is "SQLCODE 01840016
  "EXECSQL FETCH C3 INTO :OUTREC3" 01850016
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01860016
  SAY "SQLCODE from ZTABCHK Fetch is "SQLCODE 01870016
TEMPSQLCODE = SQLCODE 01880016
"EXECSQL CLOSE C3" 01890003
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01900016

```

```

        SAY "SQLCODE from ZTABCHK CLOSE is "SQLCODE           01910016
        SQLCODE = TEMPSQLCODE                                01920016
        END                                                  01930016
    END                                                    01940016
END                                                        01950003
/*****/ 01960003
/* If we dropped out of loop because of success, update our table */ 01970003
/* and all other ZMCOD tables with the target DBNAME */ 01980003
/*****/ 01990003
if loopctr2 < 20 then do 02000003
    countchanges = countchanges + 1 02020005
    UPDATESQL1 = "UPDATE "TRGOWNER".ZMCOD_DATABASE ", 02030016
    "SET TRGDBNAME ='"|NEWDBNAME"' WHERE SRCDBNAME ='"TRGDBNAME"'" 02040016
    IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02050016
        SAY "UPDATESQL is "UPDATESQL1 02060016
        "EXECSQL "UPDATESQL1 02070002
        IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02080016
            SAY "SQLCODE from UPDATESQL1 is "SQLCODE SQLERRMC 02090016
            02100008
        UPDATESQL2 = "UPDATE "TRGOWNER".ZMCOD_TABLES ", 02110016
        "SET TRGDBNAME ='"|NEWDBNAME"' WHERE SRCDBNAME ='"TRGDBNAME"'" 02120016
        IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02130016
            SAY "UPDATESQL is "UPDATESQL2 02140016
            "EXECSQL "UPDATESQL2 02150013
            IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02160016
                SAY "SQLCODE from UPDATESQL2 is "SQLCODE SQLERRMC 02170016
                02180008
            UPDATESQL3 = "UPDATE "TRGOWNER".ZMCOD_TABLESPACE ", 02190016
            "SET TRGDBNAME ='"|NEWDBNAME"' WHERE SRCDBNAME ='"TRGDBNAME"'" 02200016
            IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02210016
                SAY "UPDATESQL is "UPDATESQL3 02220016
                "EXECSQL "UPDATESQL3 02230013
                IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02240016
                    SAY "SQLCODE from UPDATESQL3 is "SQLCODE SQLERRMC 02250016
                    02260008
                UPDATESQL4 = "UPDATE "TRGOWNER".ZMCOD_INDEXES ", 02270016
                "SET TRGDBNAME ='"|NEWDBNAME"' WHERE SRCDBNAME ='"TRGDBNAME"'" 02280016
                IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02290016
                    SAY "UPDATESQL is "UPDATESQL4 02300016
                    "EXECSQL "UPDATESQL4 02310013
                    IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 02320016
                        SAY "SQLCODE from UPDATESQL4 is "SQLCODE SQLERRMC 02330016
                        02340013
                    SAY " Resolved "TRGDBNAME " to "NEWDBNAME 02350016
                END 02360016
            END 02370016
        END 02380003
    /*****/ 02390003

```

```

/* Increment counter, fetch next record for cursor1 and continue */ 02400003
/* loop */ 02410003
/*****/ 02420003
                                02430003
                                02440005
                                02450003
                                02460016
                                02470016
                                02480001
                                02490001
                                02500003
/*****/ 02510003
/* Close main cursor and exit */ 02520006
/*****/ 02530003
                                02540003
                                02550016
                                02560016
                                02570016
                                02580016
                                02590016
                                02600016
                                02610001
                                02620016
                                02630001
SAY "-----"
SAY "Total databases processed "counttotal
SAY "Total databases without duplicates "countok
SAY "Total databases changed "countchanges
SAY "Total databases unresolved "countunresolved
SAY "-----"
"EXECSQL CLOSE C1"
SAY"SQLCODE from MAINLOOP CLOSE is "SQLCODE
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')

```

JCL to run the duplicate resolution REXX procedure

In Example B-17, we provide the JCL used to run the duplicate resolution REXX procedure on the target subsystem.

Example: B-17 JCL to run the duplicate resolution REXX procedure

```

//SAPRES2R JOB (999,TIM),'RESOLVE DUPS ',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,REGION=0M
/*JOBPARM SYSAFF=SC49,L=9999
/*
//RUNREXX EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DB7T7.SDSNEXIT,DISP=SHR
// DD DSN=DB7T7.SDSNLOAD,DISP=SHR
//SYSEXEC DD DISP=SHR,DSN=SAPRES2.MCOD.REXX
//OUTDDDB DD SYSOUT=*
//OUTDDTS DD SYSOUT=*
//OUTDDTB DD SYSOUT=*
//OUTDDIX DD SYSOUT=*
//OUTDDRE DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%DUPLICDB DB7S SAPBLU '710' DB7T SAPBLU '710' MCDBLU BLU SAPR3
/*

```

Updating metadata tables

Once the objects are defined in the target DB2 catalog, the metadata tables must be updated with new tablespace and index values.

POSTDDL

In Example B-18, we provide the REXX procedure used to update the target DB2 catalog with new tablespace values.

Example: B-18 REXX procedure to update target catalog with new tablespace values

```

/*****/ 00010000
/* SAP MCOB Migration - Change DBNAME if duplicate exists */ 00020000
/*****/ 00030000
00040009
/*****/ 00050000
/* */ 00060000
/* Initialize variables passed to program */ 00070000
/* Name Description */ 00080000
/* SRCSSID Source DB2 Subsystem ID */ 00090000
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00100000
/* SRCRELEASE Source DB2 Version Number */ 00110000
/* TRGSSID Target DB2 Subsystem ID */ 00120000
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00130000
/* TRGRELEASE Target DB2 Version Number */ 00140000
/* TRGVCAT Target DB2 VCAT name */ 00150000
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00160000
/* TRGOWNER Target DB2 Owner of SAP Objects */ 00161005
/* */ 00170000
/*****/ 00180000
00190000
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00200000
TRGSSID TRGSCHEMA TRGRELEASE, 00210000
TRGVCAT TRGSTOGRPPRE TRGOWNER 00220005
00230000
VERBOSE = N 00240011
00250000
/*****/ 00260000
/* Add DSNREXX to the host command environment table if not there. */ 00270000
/*****/ 00280000
00290000
'SUBCOM DSNREXX' 00300000
IF RC THEN , 00310000
S_RC = RXSUBCOM('ADD', 'DSNREXX', 'DSNREXX') 00320000
ADDRESS DSNREXX 00330000
00340000
/*****/ 00350000
```

```

/* Connect to DB2 */ 00360000
/*****/ 00370000
00380000
'CONNECT' TRGSSID 00390004
/* trace i */ 00400000
00410000
/*****/ 00420000
/* Setup Fetch SQL statement, declare cursor, prepare stament */ 00430000
/* This cursor remains open and loops through ZMCOB_TABLESPACE */ 00440010
/*****/ 00450000
00460000
MAINLOOP = "SELECT TRGDBID,TRGIPREFIX,TRGPSID,TRGTSNAME ", 00470002
          "FROM "TRGOWNER".ZMCOB_TABLESPACE ", 00480006
          "WHERE TRGSCHEMA = '"TRGSCHEMA'" ", 00490002
          "FOR UPDATE OF TRGDBID,TRGIPREFIX,TRGPSID" 00491002
"EXECSQL DECLARE C1 CURSOR FOR S1" 00500002
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00510010
  SAY "SQLCODE from MAINLOOP DECLARE is "SQLCODE 00521010
"EXECSQL PREPARE S1 INTO :OUTREC FROM :MAINLOOP" 00530000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00531010
  SAY "SQLCODE from MAINLOOP PREPARE is "SQLCODE 00540010
00550000
/*****/ 00560000
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00570000
/* This cursor is opened to read SYSTABLESPACE looking for clashes */ 00580000
/*****/ 00590000
00600000
SQLSTMT = "SELECT ", 00610000
          "TS.DBID, ", 00620000
          "TSP.IPREFIX, ", 00630000
          "TS.PSID ", 00640000
          " FROM SYSIBM.SYSTABLESPACE TS JOIN SYSIBM.SYSTABLEPART TSP", 00650000
          " ON TS.DBNAME = TSP.DBNAME ", 00660002
          " WHERE TS.NAME = ? AND TS.CREATOR = '"TRGSCHEMA'" " 00670002
"EXECSQL DECLARE C2 CURSOR FOR S2" 00680000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00681010
  SAY "SQLCODE from C2 PREPARE is "SQLCODE 00690010
"EXECSQL PREPARE S2 INTO :OUTREC2 FROM :SQLSTMT" 00700002
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00701010
  SAY "SQLCODE from C2 PREPARE is "SQLCODE 00710010
00720000
/*****/ 00730000
/* Open Cursor for reading through ZMCOB_TABLESPACE and fetch */ 00740000
/* the first row. */ 00750000
/*****/ 00760000
00770000
"EXECSQL OPEN C1" 00780000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00790010
  SAY "SQLCODE from MAINLOOP OPEN is "SQLCODE 00800010

```



```

"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "           00810000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                00820010
  SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE        00830010
                                                       00840000
/*****/                                                00850000
/* Set loop counter and loop while rows are found     */ 00860000
/*****/                                                00870000
                                                       00880000
counttotal = 0                                       00890000
DO WHILE SQLCODE = 0                                  00900000
/* if counttotal = 5 then leave */                    00910008
  IF (VERBOSE = Y) | (SQLCODE <> 0) THEN              00920010
    SAY counttotal "Fetched TS is "OUTREC.4.SQLDATA   00930002
                                                       00950000
/*****/                                                00960000
/* Open cursor in SYSTABLESPACE join and read new fields */ 00970000
/*****/                                                00980000
                                                       00990000
"EXECSQL OPEN C2 USING :OUTREC.4.SQLDATA"             01000002
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                01010010
  SAY "SQLCODE from SYSTSQL OPEN is "SQLCODE SQLERRMC SQLERRP 01020002
"EXECSQL FETCH C2 USING DESCRIPTOR :OUTREC2"         01030002
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                01040010
  SAY "SQLCODE from SYSTSQL Fetch is "SQLCODE SQLERRMC SQLERRP 01050002
                                                       01051002
/*****/                                                01070000
/* setup update statement for ZMCD_TABLESPACE and execute */ 01080000
/*****/                                                01100000
  UPDATESQL = "UPDATE "TRGOWNER".ZMCD_TABLESPACE SET ", 01110000
              "TRGDBID = 'OUTREC2.1.SQLDATA'", ",      01120002
              "TRGIPREFIX = 'OUTREC2.2.SQLDATA'", ",    01130002
              "TRGPSID = "OUTREC2.3.SQLDATA" ",         01140002
              "WHERE CURRENT OF C1"                     01150000
"EXECSQL "UPDATESQL                                   01170007
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                01180010
  UPDATESQL = "UPDATE "TRGOWNER".ZMCD_TABLESPACE SET ", 01110000
              "TRGDBID = 'OUTREC2.1.SQLDATA'", ",      01120002
              "TRGIPREFIX = 'OUTREC2.2.SQLDATA'", ",    01130002
              "TRGPSID = "OUTREC2.3.SQLDATA" ",         01140002
              "WHERE CURRENT OF C1"                     01150000
"EXECSQL "UPDATESQL                                   01170007
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                01180010
  SAY "SQLCODE from UPDATESQL is "SQLCODE SQLERRMC     01190010
                                                       01200000
"EXECSQL CLOSE C2"                                   01210000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                01220010
  SAY "SQLCODE from UPDATESQL CLOSE is "SQLCODE        01230010
                                                       01240000
/*****/                                                01250000

```

```

/* Increment counter, fetch next record for cursor1 and continue */ 01260000
/* loop */ 01270000
/*****/ 01280000
                                01290000
counttotal=counttotal+1 01300000
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 01310000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 01320010
    SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE 01330010
                                01340000
END 01350000
                                01360000
/*****/ 01370000
/* Close main cursor and exit */ 01380000
/*****/ 01390000
                                01400000
SAY "-----" 01410000
SAY "Total tablespaces processed "counttotal 01420000
SAY "-----" 01430000
"EXECSQL CLOSE C1" 01440000
SAY"SQLCODE from MAINLOOP CLOSE is "SQLCODE 01450000
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX') 01460000

```

POSTDDL2

In Example B-19, we provide the REXX procedure used to update the target DB2 catalog with new index values.

Example: B-19 REXX procedure to update target catalog with new index values

```

/*****/ 00010000
/* SAP MCOB Migration - Update ZMCOB_INDEXES with new fields */ 00020000
/* after running of DDL in target system. */ 00030000
/*****/ 00040000
                                00050000
/*****/ 00060000
/* */ 00070000
/* Initialize variables passed to program */ 00080000
/* Name Description */ 00090000
/* SRCSSID Source DB2 Subsystem ID */ 00100000
/* SRCSCHEMA Source DB2 Owner of SAP Objects */ 00110000
/* SRCRELEASE Source DB2 Version Number */ 00120000
/* TRGSSID Target DB2 Subsystem ID */ 00130000
/* TRGSCHEMA Target DB2 Owner of SAP Objects */ 00140000
/* TRGRELEASE Target DB2 Version Number */ 00150000
/* TRGVCAT Target DB2 VCAT name */ 00160000
/* TRGSTOGRPPRE Target DB2 Storage Group Prefix (1st 3 Characters)*/ 00170000
/* TRGOWNER Target DB2 Owner of SAP Objects */ 00180000
/* */ 00190000

```

```

/*****/ 00200000
00210000
PARSE ARG SRCSSID SRCSCHEMA SRCRELEASE, 00220000
      TRGSSID TRGSCHEMA TRGRELEASE,      00230000
      TRGVCAT TRGSTOGRPPRE TRGOWNER      00240000
00250000
VERBOSE = N 00260002
00270000
/*****/ 00280000
/* Add DSNREXX to the host command environment table if not there. */ 00290000
/*****/ 00300000
00310000
'SUBCOM DSNREXX' 00320000
IF RC THEN , 00330000
    S_RC = RXSUBCOM('ADD', 'DSNREXX', 'DSNREXX') 00340000
ADDRESS DSNREXX 00350000
00360000
/*****/ 00370000
/* Connect to DB2 */ 00380000
/*****/ 00390000
00400000
'CONNECT' TRGSSID 00410000
/* trace i */ 00420000
00430000
/*****/ 00440000
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00450000
/* This cursor remains open and loops through ZMCOD_INDEXES rows */ 00460001
/*****/ 00470000
00480000
MAINLOOP = "SELECT TRGDBID,TRGIPREFIX,TRGISOBID, ", 00490001
          "TRGINDEXSPACE,TRGINDEX ", 00500001
          "FROM "TRGOWNER".ZMCOD_INDEXES ", 00510000
          "WHERE TRGSCHEMA = 'TRGSCHEMA' ", 00520000
          "FOR UPDATE OF TRGDBID,TRGIPREFIX,TRGISOBID,TRGINDEXSPACE" 00530001
"EXECSQL DECLARE C1 CURSOR FOR S1" 00540000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00550001
    SAY "SQLCODE from MAINLOOP DECLARE is "SQLCODE SQLERRMC SQLERRP 00560001
"EXECSQL PREPARE S1 INTO :OUTREC FROM :MAINLOOP" 00570000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00580001
    SAY "SQLCODE from MAINLOOP PREPARE is "SQLCODE SQLERRMC SQLERRP 00590001
00600000
/*****/ 00610000
/* Setup Fetch SQL statement, declare cursor, prepare statement */ 00620001
/* fetching new data for INDEXES after execution of DDL on target */ 00630001
/*****/ 00640000
00650000
SQLSTMT = "SELECT ", 00660000
          "IX.DBID, ", 00670000
          "IXP.IPREFIX, ", 00680000

```

```

"IX.ISOBID, ", 00690001
"IX.INDEXSPACE ", 00700000
" FROM SYSIBM.SYSINDEXES IX JOIN SYSIBM.SYSINDEXPART IXP", 00710001
" ON IX.NAME = IXP.IXNAME AND IX.CREATOR = IXP.IXCREATOR", 00720001
" WHERE IX.NAME = ? AND IX.CREATOR = 'TRGSCHEMA'" 00730000
"EXECSQL DECLARE C2 CURSOR FOR S2" 00740000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00750001
  SAY "SQLCODE from C2 PREPARE is "SQLCODE SQLERRMC SQLERRP 00760001
"EXECSQL PREPARE S2 INTO :OUTREC2 FROM :SQLSTMT" 00770000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00780001
  SAY "SQLCODE from C2 PREPARE is "SQLCODE SQLERRMC SQLERRP 00790001
00800000
/*****/ 00810000
/* Open Cursor for reading through ZMCOB_TABLESPACE and fetch */ 00820000
/* the first row. */ 00830000
/*****/ 00840000
00850000
"EXECSQL OPEN C1" 00860000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00870001
  SAY "SQLCODE from MAINLOOP OPEN is "SQLCODE 00880001
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC " 00890000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 00900001
  SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE 00910001
00920000
/*****/ 00930000
/* Set loop counter and loop while rows are found */ 00940000
/*****/ 00950000
00960000
counttotal = 0 00970000
DO WHILE SQLCODE = 0 00980000
  IF (VERBOSE = Y) | (SQLCODE <> 0) THEN 00990001
    SAY counttotal "Fetched IX is "OUTREC.5.SQLDATA 01000001
  01010000
  01020000
/*****/ 01030000
/* Open cursor in SYSTABLESPACE join and read new fields */ 01040000
/*****/ 01050000
01060000
"EXECSQL OPEN C2 USING :OUTREC.5.SQLDATA" 01070000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 01080001
  SAY "SQLCODE from SYSTSSQL OPEN is "SQLCODE SQLERRMC SQLERRP 01090000
"EXECSQL FETCH C2 USING DESCRIPTOR :OUTREC2" 01100000
IF (VERBOSE = Y) | (SQLCODE < 0) THEN 01110001
  SAY "SQLCODE from SYSTSSQL Fetch is "SQLCODE SQLERRMC SQLERRP 01120000
01130000
/*****/ 01140000
/* setup update statement for ZMCOB_TABLESPACE and execute */ 01150000
/*****/ 01160000
01170000

```

```

UPDATESQL = "UPDATE "TRGOWNER".ZMCOB_INDEXES SET ",           01180000
            "TRGDBID   = "OUTREC2.1.SQLDATA", ",           01190001
            "TRGIPREFIX = '"OUTREC2.2.SQLDATA"', ",       01200000
            "TRGISOBID  = "OUTREC2.3.SQLDATA", ",       01210001
            "TRGINDEXSPACE = '"OUTREC2.4.SQLDATA"' ",     01220000
            "WHERE CURRENT OF C1"                          01230000

"EXECSQL "UPDATESQL
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                    01260001
    Say "SQLCODE from UPDATESQL is "SQLCODE SQLERRMC     01270000
                                                    01280000

"EXECSQL CLOSE C2"                                       01290000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                    01300001
    SAY "SQLCODE from UPDATESQL CLOSE is "SQLCODE       01310001
                                                    01320000

/*****/                                                  01330000
/* Increment counter, fetch next record for cursor1 and continue */ 01340000
/* loop */                                              01350000
/*****/                                                  01360000

counttotal=counttotal+1                                 01370000
"EXECSQL FETCH C1 USING DESCRIPTOR :OUTREC "            01380000
IF (VERBOSE = Y) | (SQLCODE <> 0) THEN                    01390000
    SAY "SQLCODE from MAINLOOP FETCH is "SQLCODE       01400001
                                                    01410001
                                                    01420000

END                                                       01430000
                                                    01440000

/*****/                                                  01450000
/* Close main cursor and exit */                        01460000
/*****/                                                  01470000
                                                    01480000

SAY "-----"                                           01490000
SAY "Total indexes processed "counttotal                01500000
SAY "-----"                                           01510000

"EXECSQL CLOSE C1"                                       01520000
SAY"SQLCODE from MAINLOOP CLOSE is "SQLCODE           01530000
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')         01540000

```

JCL used to run the update REXX procedures

In Example B-20, we provide the JCL used to run the update REXX procedures on the target subsystem.

Example: B-20 JCL to run update REXX procedures

```

//SAPRES3D JOB (999,KEL),'POST DDL CHANGES',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,REGION=0M
/*JOBPARM SYSAFF=SC49,L=9999
//*
```

```
//RUNREXX EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DB7T7.SDSNEXIT,DISP=SHR
//          DD DSN=DB7T7.SDSNLOAD,DISP=SHR
//SYSEXEC DD DISP=SHR,DSN=SAPRES2.MCOD.REXX
//OUTDDDB DD SYSOUT=*
//OUTDDTS DD SYSOUT=*
//OUTDDTB DD SYSOUT=*
//OUTDDIX DD SYSOUT=*
//OUTDDRE DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%POSTDDL DB7S SAPR3 '710' DB7T SAPBLU '710' MCDBLU BLU SAPR3
%POSTDDL2 DB7S SAPR3 '710' DB7T SAPBLU '710' MCDBLU BLU SAPR3
//*
```

Merge in place: Defining source objects in target system

This appendix contains material that supports the DDL generation, tailoring, and execution steps of the merge in place process. This includes DDL, REXX procedures, and JCL to perform the following tasks:

- ▶ Generating DDL using DB2 Admin
- ▶ Tailoring the output from DB2 Admin
- ▶ Using filler tablespaces to reserve OBIDs

Generating DDL using DB2 Admin

We use the reverse engineering functionality of DB2 Admin to generate the data definition language (DDL) for creating source databases, tablespaces, tables, and indexes in the target DB2 catalog.

Invoking DB2 Admin

In Example C-1, we provide a sample JCL used to invoke DB2 Admin. This JCL is generated by the REXX procedure JCLGEN.

Example: C-1 JCL used to invoke DB2 Admin

```
//A000X00V EXEC PGM=IKJEFT01,DYNAMNBR=100
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(DB7S)
    RUN PROG(ADB2GEN) PLAN(ADB2GEN) PARMS('/MASK')
    END
/*
//MASKS DD *
    SGNAMESAP*,BLU*
/*
//SQL DD DISP=MOD,
// DSN=SAPRES4.MERGBLU.REXXFILE.DDLDB.UNTAIL
//SYSPRINT DD SYSOUT=*
//IN DD *
    DB2SYS = 'DB7S'
    DB2ALOC = ' ',
    DB2SERV = 'DB7S',
    DB2AUTH = 'SAPRES1',
    DB2REL = '710',
    RUNSQLID = 'SAPBLU',
    GENSG = 'N',
    GENDB = 'Y',
    GENTS = 'N',
    GENTABLE = 'N',
    GENVIEW = 'N',
    GENINDEX = 'N',
    GENSYN = 'N',
    GENALIAS = 'N',
    GENLABEL = 'N',
    GENCOMM = 'N',
    GENRELS = 'N',
    GENTRIG = 'N',
    GRANTDB = 'N',
    GRANTTS = 'N',
    GRANTTAB = 'N',
```



```

GRANTVW = 'N',
GRANTSG = 'N',
NEWDB   = 'A000X02A',
NEWSSG  = '',
NEWIXSG = '',
NEWSQLID = 'SAPBLU',
SPCALLOC = 'DEFINED',
COMMITFR = 'N',
DEFAULTS = 'R',
TGTDB2  = '710';
DB='A000X00V', TS='';
/*

```

In this example:

- ▶ DB2 Admin is invoked for database A000X00Y (DB='A000X00Y') that is being renamed to A000X02A (NEWDB = 'A000X02A').
- ▶ This JCL generates statements for CREATE DATABASE (GENDB='Y'). Other jobs for creating DDL for other objects would use parameters GENTS, GENTABLE, and GENINDEX.
- ▶ The generated DDL is appended to the SAPRES4.MRGBLU.REXXFILE.DDLDB.UNTAIL file.
- ▶ The storage group name is renamed from SAP* to BLU* using the MASK parameter.

JCLGEN

In Example C-2, we provide the REXX procedure used to generate JCL to invoke DB2 Admin. The source of this procedure (the JCLGEN.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-2 REXX procedure to generate JCL to invoke DB2 Admin

```

/*****
/* REXXSQL

```

```

REXX routine name: JCLGEN

```

```

This routine is used to generate JCL for invoking
DB2 Administration Tool in batch. To simplify the REXX it
uses the following external subroutines:
JCLGENDB (for generating JCL for CREATE DATABASE)
JCLGENTS (for generating JCL for CREATE TABLESPACE)
JCLGENTB (for generating JCL for CREATE TABLE)
JCLGENIX (for generating JCL for CREATE INDEX)

```

The logic of the routine is:
 Loop through the ZMCOB_DATABASE table. For each database:
 - call subroutine JCLGENDB for database JCL
 - call subroutine JCLGENTS for tablespace JCL
 - call subroutine JCLGENTB for table JCL
 - call subroutine JCLGENIX for index JCL

Parameters to call this procedure:
 METADATA_SSID Target DB2 subsystem containing metadata tables
 METADATA_OWNER Owner of the metadata tables in target system
 DATASET_PREFIX Prefix for the dataset names to generate
 DDL & JCL into
 AUTHID AUTHID for Admin Tool to use

Note that this REXX routine needs to run against the metadata tables located on the target DB2 system. Review the parameters to ensure this.

```

*****/
/* TRACE(R) */

PARSE ARG METADATA_SSID METADATA_OWNER DATASET_PREFIX AUTHID
SAY ' '
SAY 'JCLGEN executing in DB2 subsystem' METADATA_SSID
SAY '      using metadata from table ' METADATA_OWNER'.ZMCOB_DATABASE'
SAY '      and SQL authority of      ' AUTHID
SAY '      JCL will be generated into datasets prefixed' DATASET_PREFIX
SAY ' '

/*****/
/* Initialize variables for use within this routine */
/*****/
SRC.DB2REL='999'
SRC.DB='DDDDDDDD'
SRC.SSID='SSSS'
SRC.STOG3='GGG'
SRC.SCHEMA='SSSSSSS'
TARG.DB='DDDDDDDD'
TARG.DB2REL='999'
TARG.OWNER='00000000'
TARG.SQLID='SSSSSSS'
TARG.STOG3='GGG'

/*****/
/* Add DSNREXX to the host command environment table if not there. */
/*****/
'SUBCOM DSNREXX'
IF RC THEN ,

```

```

        S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
ADDRESS DSNREXX
/*****/
/* Connect to TARGET DB2 subsystem */
/*****/
'CONNECT' METADATA_SSID

/*****/
/* Setup the cursor for reading from the ZMCOB_DATABASE table */
/* The ZMCOB_DATABASE table contains one row for every database*/
/* being merged. */
/*****/

SQLSTMT = "SELECT SRCDBNAME, SRCRELEASE, SRCSSID, TRGDBNAME, "
SQLSTMT = SQLSTMT " TRGRELEASE, TRGSCHEMA, SRCSTOGROUP, "
SQLSTMT = SQLSTMT " TRGSTOGROUP, SRCSCHEMA "
SQLSTMT = SQLSTMT " FROM " METADATA_OWNER".ZMCOB_DATABASE "
SQLSTMT = SQLSTMT " ORDER BY SRCDBNAME "
/* SAY "JCLGEN SQLSTMT is " SQLSTMT */

"EXECSQL DECLARE C1 CURSOR FOR S1"
IF SQLCODE <> 0 THEN
    DO
        SAY "JCLGEN SQLCODE from DECLARE is "SQLCODE SQLERRMC SQLERRD.5
        EXIT(12)
    END
ELSE NOP
"EXECSQL PREPARE S1 FROM :SQLSTMT"
IF SQLCODE <> 0 THEN
    DO
        SAY "JCLGEN SQLCODE from PREPARE is "SQLCODE SQLERRMC SQLERRD.5
        EXIT(12)
    END
ELSE NOP

/* Open Cursor */
"EXECSQL OPEN C1"
IF SQLCODE <> 0 THEN
    DO
        SAY "JCLGEN SQLCODE from OPEN is "SQLCODE SQLERRMC SQLERRD.5
        EXIT(12)
    END
ELSE NOP

/*****/
/* Loop through each database record */
/*****/
"EXECSQL FETCH C1 INTO " || ,

```

```

        " :SRC.DB, :SRC.DB2REL, :SRC.SSID, :TARG.DB," || ,
        " :TARG.DB2REL, :TARG.OWNER, :SRC.STOG, :TARG.STOG, :SRC.SCHEMA"
IF SQLCODE <> 0 THEN
DO
SAY "JCLGEN SQLCODE from FIRST FETCH is "SQLCODE SQLERRMC SQLERRD.5
EXIT(12)
END
ELSE NOP

/* While the fetches are OK, continue to loop */
loopctr=0
StepsInJob=0
ADDJOB CARD='Y'

DO WHILE SQLCODE = 0

CALL GETSTOG3 /* Get first 3 chars of stogroup for JCL Mask */

SAY 'JCLGEN Processing database 'SRC.DB,
' (renamed to:' TARG.DB ')'

/* Call external subroutines to generate JCL
JCLGENDB (for generating JCL for CREATE DATABASE)
JCLGENTS (for generating JCL for CREATE TABLESPACE)
JCLGENTB (for generating JCL for CREATE TABLE)
JCLGENIX (for generating JCL for CREATE INDEX)
Strip off leading and trailing spaces off parameters */

ADDRESS TSO
CALL JCLGENDB STRIP(SRC.DB),
STRIP(SRC.DB2REL),
STRIP(SRC.SSID),
STRIP(TARG.DB),
STRIP(TARG.DB2REL),
STRIP(TARG.OWNER),
STRIP(TARG.SQLID),
STRIP(SRC.STOG3),
STRIP(TARG.STOG3),
STRIP(AUTHID),
ADDJOB CARD,
STRIP(SRC.SCHEMA),
DATASET_PREFIX

CALL JCLGENTS STRIP(SRC.DB),
STRIP(SRC.DB2REL),
STRIP(SRC.SSID),
STRIP(TARG.DB),
STRIP(TARG.DB2REL),
STRIP(TARG.OWNER),

```

```

        STRIP(TARG.SQLID),
        STRIP(SRC.STOG3),
        STRIP(TARG.STOG3),
        STRIP(AUTHID),
        ADDJOB CARD,
        STRIP(SRC.SCHEMA),
        DATASET_PREFIX

CALL JCLGENTB STRIP(SRC.DB),
             STRIP(SRC.DB2REL),
             STRIP(SRC.SSID),
             STRIP(TARG.DB),
             STRIP(TARG.DB2REL),
             STRIP(TARG.OWNER),
             STRIP(TARG.SQLID),
             STRIP(SRC.STOG3),
             STRIP(TARG.STOG3),
             STRIP(AUTHID),
             ADDJOB CARD,
             STRIP(SRC.SCHEMA),
             DATASET_PREFIX

CALL JCLGENIX STRIP(SRC.DB),
             STRIP(SRC.DB2REL),
             STRIP(SRC.SSID),
             STRIP(TARG.DB),
             STRIP(TARG.DB2REL),
             STRIP(TARG.OWNER),
             STRIP(TARG.SQLID),
             STRIP(SRC.STOG3),
             STRIP(TARG.STOG3),
             STRIP(AUTHID),
             ADDJOB CARD,
             STRIP(SRC.SCHEMA),
             DATASET_PREFIX

loopctr=loopctr+1          /* prevent infinite loop */
IF loopctr = 100000 THEN  /* or just limit to speed up testing */
DO
    SAY 'Stopped after ' loopctr ' databases '
    EXIT(12)
END
ELSE NOP

/* Insert Jobcard every 250 steps (JES limit is 250 steps per job) */
StepsInJob = StepsInJob + 1
IF StepsInJob > 250 THEN
DO
    SAY 'JCLGEN New jobcard being generated after' StepsInJob 'steps'

```

```

        ADDJOBCARD = 'Y'
        StepsInJob = 0
        END
ELSE
    ADDJOBCARD = 'N'

/* Read the next database record */
ADDRESS DSNREXX
"EXECSQL FETCH C1 INTO " || ,
    " :SRC.DB, :SRC.DB2REL, :SRC.SSID, :TARG.DB," || ,
    " :TARG.DB2REL, :TARG.OWNER, :SRC.STOG, :TARG.STOG, :SRC.SCHEMA"
IF (SQLCODE = 0) | (SQLCODE = 100) THEN
    NOP
ELSE
    DO
        SAY "JCLGEN SQLCODE from FETCH NEXT is "SQLCODE SQLERRMC SQLERRD.5
        EXIT(12)
    END
END
/*****/
/* Cleanup and exit */
/*****/
/* Close Cursor */
"EXECSQL CLOSE C1"
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')
EXIT
/*****/
/* SUBROUTINES */
/*****/
/* Subroutine used to extract the first 3 characters of the */
/* current source and target stogroups */
/* This is used in the DB2 Admin JCL for masking. */
/*****/
GETSTOG3:

/* Separate STOG into STOG3 and StogLast3Chars at position 4 */
PARSE VAR SRC.STOG SRC.STOG3 4 SRC.StogLast3Chars
PARSE VAR TARG.STOG TARG.STOG3 4 TARG.StogLast3Chars

RETURN
/*****/

```

JCLGENDB

In Example C-3, we provide the REXX subroutine used to generate JCL for creating DDL for databases. The source of this subroutine (the JCLGENDB.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, "Additional material" on page 265).

Example: C-3 REXX subroutine to generate JCL for creating DDL for databases

```
/******  
/* REXXSQL  
  
REXX routine name: JCLGENDB (subroutine of REXX routine JCLGEN)  
  
Used to generate JCL for invoking the DB2 Administration Tool.  
This routine is called from the JCLGEN REXX routine for each  
database which is being merged. This routine then generates  
JCL based on this information.  
  
NOTE: The generated JCL will be run against the source DB2  
system, so review at least these items in the JCL statements:  
- JOBCARD, including JOBPARM to direct job to source MVS system  
- SDSNLOAD/SDSNEXIT for the source DB2 system  
  
*****  
/* Accept input values */  
ARG SRC.DB,  
SRC.DB2REL,  
SRC.SSID,  
TARG.DB,  
TARG.DB2REL,  
TARG.OWNER,  
TARG.SQLID,  
SRC.STOG3,  
TARG.STOG3,  
AUTHID,  
ADDJOB CARD,  
SRC.SCHEMA,  
DATASET_PREFIX  
  
/* Write a jobcard if requested */  
IF ADDJOB CARD = 'Y' THEN  
DO  
QUEUE '//SAPRES41 JOB (999,KEL), 'GEN DB DDL', CLASS=A, MSGCLASS=T, "  
QUEUE '// NOTIFY=&SYSUID, TIME=1440, REGION=OM, TYPRUN=HOLD "  
QUEUE '//JOB LIB DD DSN=DB7S7.SDSNEXIT, DISP=SHR "  
QUEUE '// DD DSN=DB7S7.SDSNLOAD, DISP=SHR "  
QUEUE '// DD DISP=SHR, DSN=ADB.V4R1MO.SADBLLIB "DO
```

```

        QUEUE "/*JOBPARM SYSAFF=SC42,L=9999"
        QUEUE "/******"
        QUEUE "/*"
        QUEUE "/* JCL TO CALL DB2 ADMINISTRATION TOOL TO GENERATE DDL FOR"
        QUEUE "/* CREATING DATABASES"
        QUEUE "/*"
        QUEUE "/******"
        END
    ELSE NOP

/* Put JCL into a queue (fifo stack) and write them to database JCL */
    QUEUE "/******"
    QUEUE "/* CREATE DATABASE STATEMENT FOR DATABASE "TARG.DB"
    QUEUE "/******"
    QUEUE "/*"SRC.DB" EXEC PGM=IKJEFT01,DYNAMNBR=100
    QUEUE "/*SYSTSPT DD SYSOUT=*
    QUEUE "/*SYSTSIN DD *
    QUEUE " DSN SYSTEM("SRC.SSID")
    QUEUE " RUN PROG(ADB2GEN) PLAN(ADB2GEN) PARMS('/MASK')
    QUEUE " END
    QUEUE "/*
    QUEUE "/*MASKS DD *
    QUEUE " SGMNAME:"SRC.STOG3"*,"TARG.STOG3"*
    QUEUE "/*
    QUEUE "/*SQL DD DISP=MOD,
    QUEUE "/* DSN="DATASET_PREFIX".DDLDB.UNTAIL
    QUEUE "/*SYSPRINT DD SYSOUT=*
    QUEUE "/*IN DD *
    QUEUE " DB2SYS = "SRC.SSID"
    QUEUE " DB2ALOC = ' ',
    QUEUE " DB2SERV = "SRC.SSID",
    QUEUE " DB2AUTH = "AUTHID",
    QUEUE " DB2REL = "SRC.DB2REL",
    QUEUE " RUNSQLID = "TARG.OWNER",
    QUEUE " GENSG = 'N',
    QUEUE " GENDB = 'Y',
    QUEUE " GENTS = 'N',
    QUEUE " GENTABLE = 'N',
    QUEUE " GENVIEW = 'N',
    QUEUE " GENINDEX = 'N',
    QUEUE " GENSYN = 'N',
    QUEUE " GENALIAS = 'N',
    QUEUE " GENLABEL = 'N',
    QUEUE " GENCOMM = 'N',
    QUEUE " GENRELS = 'N',
    QUEUE " GENTRIG = 'N',
    QUEUE " GRANTDB = 'N',
    QUEUE " GRANTTS = 'N',
    QUEUE " GRANTTAB = 'N',

```



```

QUEUE " GRANTVW = 'N',           "
QUEUE " GRANTSG  = 'N',           "
QUEUE " NEWDB    = '"TARG.DB"',    "
QUEUE " NEWTSSG  = '',            "
QUEUE " NEWXSG   = '',            "
QUEUE " NEWSQLID = '"TARG.OWNER"',  "
QUEUE " SPCALLOC = 'DEFINED',     "
QUEUE " COMMITFR = 'N',           "
QUEUE " DEFAULTS = 'R',           "
QUEUE " TGTDB2   = '"TARG.DB2REL"', "
QUEUE " DB='SRC.DB', TS='';       "
QUEUE "/*                          "

/* Write the JCL to the output file */
"EXECIO * DISKW JCLDB "
IF rc < 0 THEN
  DO
    SAY 'Error writing to JCLDB ' rc
  EXIT
  END
ELSE NOP
RETURN

```

JCLGENTS

In Example C-4, we provide the REXX subroutine used to generate JCL for creating DDL for tablespaces. The source of this subroutine (the JCLGENTS.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, "Additional material" on page 265).

Example: C-4 REXX subroutine to generate JCL for creating DDL for tablespaces

```

/*****
/* REXXSQL

```

REXX routine name: JCLGENTS (subroutine of REXX routine JCLGEN)

Used to generate JCL for invoking the DB2 Administration Tool. This routine is called from the JCLGEN REXX routine for each database which is being merged. This routine then generates JCL for invoking the DB2 Administration Tool to create the tablespace contained within that database.

NOTE: The generated JCL will be run against the source DB2 system, so review at least these items before generating the JCL:

- JOBCARD, including JOBPARM to direct job to source MVS system
- SDSNLOAD/SDSNEXIT for the source DB2 system

```

*****/

/* Accept input values */
ARG SRC.DB,
SRC.DB2REL,
SRC.SSID,
TARG.DB,
TARG.DB2REL,
TARG.OWNER,
TARG.SQLID,
SRC.STOG3,
TARG.STOG3,
AUTHID,
ADDJOB CARD,
SRC.SCHEMA,
DATASET_PREFIX

/* Write a jobcard if requested */
IF ADDJOB CARD = 'Y' THEN
DO
QUEUE "//SAPRES42 JOB (999,KEL), 'GEN TS DDL', CLASS=A, MSGCLASS=T, "
QUEUE "// NOTIFY=&SYSUID, TIME=1440, REGION=OM, TYPRUN=HOLD "
QUEUE "//JOB LIB DD DSN=DB7S7.SDSNEXIT, DISP=SHR "
QUEUE "// DD DSN=DB7S7.SDSNLOAD, DISP=SHR "
QUEUE "// DD DISP=SHR, DSN=ADB.V4R1MO.SADBL LIB "
QUEUE "/*JOBPARM SYSAFF=SC42, L=9999 "
QUEUE "/******"
QUEUE "/*"
QUEUE "/* JCL TO CALL DB2 ADMINISTRATION TOOL TO GENERATE DDL FOR "
QUEUE "/* CREATING TABLESPACES FOR A GIVEN DATABASE "
QUEUE "/*"
QUEUE "/******"
END
ELSE NOP

/* Put lines into a queue (fifo stack) and write them to outdd */
QUEUE "/******"
QUEUE "/* CREATE TABLESPACE STATEMENTS FOR DATABASE "TARG.DB" "
QUEUE "/******"
QUEUE "/*"SRC.DB" EXEC PGM=IKJEFT01, DYNAMNBR=100 "
QUEUE "/*SYSTSPRT DD SYSOUT=* "
QUEUE "/*SYSTSIN DD * "
QUEUE " DSN SYSTEM("SRC.SSID") "
QUEUE " RUN PROG(ADB2GEN) PLAN(ADB2GEN) PARS('/MASK') "
QUEUE " END "
QUEUE "/* "
QUEUE "/*MASKS DD * "
QUEUE " SNAME:"SRC.STOG3"*, "TARG.STOG3"* "

```

```

QUEUE /*
QUEUE //SQL DD DISP=MOD,
QUEUE // DSN="DATASET_PREFIX".DDLTS.UNTAIL "
QUEUE //SYSPRINT DD SYSOUT=*
QUEUE //IN DD *
QUEUE " DB2SYS = '"SRC.SSID"'
QUEUE " DB2ALOC = ''
QUEUE " DB2SERV = '"SRC.SSID"',
QUEUE " DB2AUTH = '"AUTHID"',
QUEUE " DB2REL = '"SRC.DB2REL"',
QUEUE " RUNSQLID = '"TARG.OWNER"',
QUEUE " GENSG = 'N',
QUEUE " GENDB = 'N',
QUEUE " GENTS = 'Y',
QUEUE " GENTABLE = 'N',
QUEUE " GENVIEW = 'N',
QUEUE " GENINDEX = 'N',
QUEUE " GENSYN = 'N',
QUEUE " GENALIAS = 'N',
QUEUE " GENLABEL = 'N',
QUEUE " GENCOMM = 'N',
QUEUE " GENRELS = 'N',
QUEUE " GENTRIG = 'N',
QUEUE " GRANTDB = 'N',
QUEUE " GRANTTS = 'N',
QUEUE " GRANTTAB = 'N',
QUEUE " GRANTVW = 'N',
QUEUE " GRANTSG = 'N',
QUEUE " NEWDB = '"TARG.DB"',
QUEUE " NEWTSSG = ''
QUEUE " NEWIXSG = ''
QUEUE " NEWSQLID = '"TARG.OWNER"',
QUEUE " SPCALLOC = 'DEFINED',
QUEUE " COMMITFR = 'N',
QUEUE " DEFAULTS = 'R',
QUEUE " TGTDB2 = '"TARG.DB2REL"';
QUEUE " DB='"SRC.DB"', TS='';
QUEUE /*

```

/* Write the JCL to the output file */
"EXECIO * DISKW JCLTS "
IF rc <> 0 THEN
DO
SAY 'Error writing to JCLTS ' rc
EXIT
END
ELSE NOP
RETURN

JCLGENTB

In Example C-5, we provide the REXX subroutine used to generate JCL for creating DDL for tables. The source of this subroutine (the JCLGENTB.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-5 REXX subroutine to generate JCL for creating DDL for tables

```

/*****/
/* REXXSQL

      REXX routine name: JCLGENTB (subroutine of REXX routine JCLGEN)

      Used to generate JCL for invoking the DB2 Administration Tool.
      This routine is called from the JCLGEN REXX routine for each
      database which is being merged. This routine then generates JCL
      for invoking the DB2 Administration Tool to create the
      the tables contained within this tablespace.

      NOTE: The generated JCL will be run against the source DB2
      system, so review at least these items before generating the JCL:
      - JOBCARD, including JOBPARM to direct job to source MVS system
      - SDSNLOAD/SDSNEXIT for the source DB2 system

*****/

/* Accept input values                                     */
ARG SRC.DB,
    SRC.DB2REL,
    SRC.SSID,
    TARG.DB,
    TARG.DB2REL,
    TARG.OWNER,
    TARG.SQLID,
    SRC.STOG3,
    TARG.STOG3,
    AUTHID,
    ADDJOBCARD,
    SRC.SCHEMA,
    DATASET_PREFIX

/* Write a jobcard if requested                             */
IF ADDJOBCARD = 'Y' THEN
DO
    QUEUE "//SAPRES43 JOB (999,KEL), 'GEN TB DDL', CLASS=A, MSGCLASS=T, "
    QUEUE "// NOTIFY=&SYSUID, TIME=1440, REGION=OM, TYPRUN=HOLD      "
    QUEUE "//JOBLIB DD DSN=DB7S7.SDSNEXIT, DISP=SHR                  "
    QUEUE "//          DD DSN=DB7S7.SDSNLOAD, DISP=SHR                "
```

```

        QUEUE "//          DD DISP=SHR,DSN=ADB.V4R1M0.SADBLLIB          "
        QUEUE "/*JOBPARM SYSAFF=SC42,L=9999                             "
        QUEUE "/******"
        QUEUE "/*"
        QUEUE "/* JCL TO CALL DB2 ADMINISTRATION TOOL TO GENERATE DDL FOR "
        QUEUE "/* CREATING TABLES FOR A GIVEN DATABASE                 "
        QUEUE "/*"
        QUEUE "/******"
        END
ELSE NOP

/* Put lines into a queue (fifo stack) and write them to outdd */
QUEUE "/******"
QUEUE "/* CREATE TABLE STATEMENTS FOR DATABASE "TARG.DB"          "
QUEUE "/******"
QUEUE "/*SRC.DB" EXEC PGM=IKJEFT01,DYNAMNBR=100                    "
QUEUE "/*SYSTSPRT DD SYSOUT=*"
QUEUE "/*SYSTSIN DD *"
QUEUE " DSN SYSTEM("SRC.SSID")"
QUEUE " RUN PROG(ADB2GEN) PLAN(ADB2GEN) PARS('/MASK')"
QUEUE " END"
QUEUE "/*"
QUEUE "/*MASKS          DD *"
QUEUE "   SGMNAME:"SRC.STOG3"*,"TARG.STOG3"*"
QUEUE "/*"
QUEUE "/*SQL          DD DISP=MOD,"
QUEUE "/*          DSN="DATASET_PREFIX".DDLTB.UNTAIL          "
QUEUE "/*SYSPRINT DD SYSOUT=*"
QUEUE "/*IN          DD *"
QUEUE " DB2SYS      = 'SRC.SSID'"
QUEUE " DB2ALOC     = ','"
QUEUE " DB2SERV     = 'SRC.SSID',"
QUEUE " DB2AUTH     = 'AUTHID',"
QUEUE " DB2REL      = 'SRC.DB2REL',"
QUEUE " RUNSQLID    = 'TARG.OWNER',"
QUEUE " GENSG       = 'N',"
QUEUE " GENDB       = 'N',"
QUEUE " GENTS       = 'N',"
QUEUE " GENTABLE    = 'Y',"
QUEUE " GENVIEW     = 'N',"
QUEUE " GENINDEX   = 'N',"
QUEUE " GENSYN     = 'N',"
QUEUE " GENALIAS   = 'N',"
QUEUE " GENLABEL   = 'N',"
QUEUE " GENCOMM    = 'N',"
QUEUE " GENRELS    = 'N',"
QUEUE " GENTRIG    = 'N',"
QUEUE " GRANTDB    = 'N',"
QUEUE " GRANTTS    = 'N',"

```

```

QUEUE " GRANTTAB = 'N',           "
QUEUE " GRANTVW  = 'N',           "
QUEUE " GRANTSG  = 'N',           "
QUEUE " NEWDB    = '"TARG.DB"',    "
QUEUE " NEWTSSG  = '',            "
QUEUE " NEWXSG   = '',            "
QUEUE " NEWSQLID = '"TARG.OWNER"',  "
QUEUE " SPCALLOC = 'DEFINED',     "
QUEUE " COMMITFR = 'N',           "
QUEUE " DEFAULTS = 'R',           "
QUEUE " TGTDB2   = '"TARG.DB2REL"; "
QUEUE " DB='"SRC.DB"', TS='';     "
QUEUE "/*                          "

/* Write the JCL to the output file */
"EXECIO * DISKW JCLTB "
IF rc < 0 THEN
  DO
    SAY 'Error writing to JCLTB ' rc
  EXIT
  END
ELSE NOP
RETURN

```

JCLGENIX

In Example C-6, we provide the REXX subroutine used to generate JCL for creating DDL for indexes. The source of this subroutine (the JCLGENIX.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-6 REXX subroutine to generate JCL for creating DDL for indexes

```

/*****
/* REXXSQL

```

REXX routine name: JCLGENIX (subroutine of REXX routine JCLGEN)

Used to generate JCL for invoking the DB2 Administration Tool. This routine is called from the JCLGEN REXX routine for each database which is being merged. This routine then generates JCL for invoking the DB2 Administration Tool to create all of the indexes within this database.

NOTE: The generated JCL will be run against the source DB2 system, so review at least these items before generating the JCL:

- JOBCARD, including JOBPARM to direct job to source MVS system
- SDSNLOAD/SDSNEXIT for the source DB2 system

```

*****/

/* Accept input values                                     */
ARG SRC.DB,
SRC.DB2REL,
SRC.SSID,
TARG.DB,
TARG.DB2REL,
TARG.OWNER,
TARG.SQLID,
SRC.STOG3,
TARG.STOG3,
AUTHID,
ADDJOB CARD,
SRC.SCHEMA,
DATASET_PREFIX

/* Write a jobcard if requested                           */
IF ADDJOB CARD = 'Y' THEN
DO
QUEUE "//SAPRES44 JOB (999,KEL), 'GEN IX DDL', CLASS=A, MSGCLASS=T, "
QUEUE "// NOTIFY=&SYSUID, TIME=1440, REGION=OM, TYPRUN=HOLD "
QUEUE "//JOB LIB DD DSN=DB7S7.SDSNEXIT, DISP=SHR "
QUEUE "// DD DSN=DB7S7.SDSNLOAD, DISP=SHR "
QUEUE "// DD DISP=SHR, DSN=ADB.V4R1MO.SADBLLIB "
QUEUE "/*JOBPARM SYSAFF=SC42, L=9999 "
QUEUE "/******"
QUEUE "/*"
QUEUE "/* JCL TO CALL DB2 ADMINISTRATION TOOL TO GENERATE DDL FOR "
QUEUE "/* CREATING INDEXES FOR A GIVEN DATABASE "
QUEUE "/*"
QUEUE "/******"
END
ELSE NOP

/* Put JCL into a queue (fifo stack) and write them to outdd */
QUEUE "/******"
QUEUE "/* CREATE INDEX STATEMENTS FOR DATABASE "TARG.DB" "
QUEUE "/******"
QUEUE "/*SRC.DB" EXEC PGM=IKJEFT01, DYNAMNBR=100 "
QUEUE "/*SYSTSPRT DD SYSOUT=* "
QUEUE "/*SYSTSIN DD * "
QUEUE " DSN SYSTEM("SRC.SSID") "
QUEUE " RUN PROG(ADB2GEN) PLAN(ADB2GEN) PARS('/MASK') "
QUEUE " END "
QUEUE "/* "
QUEUE "/*MASKS DD * "
QUEUE " SGMNAME:"SRC.STOG3"*, "TARG.STOG3"* "

```

```

QUEUE /*
QUEUE //SQL DD DISP=MOD,
QUEUE // DSN="DATASET_PREFIX".DDLIX.UNTAIL
QUEUE //SYSPRINT DD SYSOUT=*
QUEUE //IN DD *
QUEUE " DB2SYS = '"SRC.SSID"'
QUEUE " DB2ALOC = ',
QUEUE " DB2SERV = '"SRC.SSID"',
QUEUE " DB2AUTH = '"AUTHID"',
QUEUE " DB2REL = '"SRC.DB2REL"',
QUEUE " RUNSQLID = '"TARG.OWNER"',
QUEUE " GENSG = 'N',
QUEUE " GENDB = 'N',
QUEUE " GENTS = 'N',
QUEUE " GENTABLE = 'N',
QUEUE " GENVIEW = 'N',
QUEUE " GENINDEX = 'Y',
QUEUE " GENSYN = 'N',
QUEUE " GENALIAS = 'N',
QUEUE " GENLABEL = 'N',
QUEUE " GENCOMM = 'N',
QUEUE " GENRELS = 'N',
QUEUE " GENTRIG = 'N',
QUEUE " GRANTDB = 'N',
QUEUE " GRANTTS = 'N',
QUEUE " GRANTTAB = 'N',
QUEUE " GRANTVW = 'N',
QUEUE " GRANTSG = 'N',
QUEUE " NEWDB = '"TARG.DB"',
QUEUE " NEWTSSG = ',
QUEUE " NEWIXSG = ',
QUEUE " NEWSQLID = '"TARG.OWNER"',
QUEUE " SPCALLOC = 'DEFINED',
QUEUE " COMMITFR = 'N',
QUEUE " DEFAULTS = 'R',
QUEUE " TGTDB2 = '"TARG.DB2REL"';
QUEUE " DB='"SRC.DB"', TS='';
QUEUE /*

/* Write the JCL to the output file */
"EXECIO * DISKW JCLIX "
IF rc <> 0 THEN
DO
SAY 'Error writing to JCLIX ' rc
EXIT
END
ELSE NOP
RETURN

```


JCL to run generation REXX procedure and submit the generated JCL

In Example C-7, we provide the JCL used to run the generation REXX procedure and submit the generated JCL.

Keep in mind that this JCL must run on the target system, because the REXX procedures use the metadata tables in the target system, and the generated JCL invoking DB2 Admin must run on the source subsystem to extract information from the source DB2 catalog.

The source of this JCL is also part of the additional material (the JCLGEN\$.JCL file) that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-7 JCL used to run the generation REXX procedure

```
//SAPRES4J JOB (999,KEL), 'JCLGEN$', CLASS=A, MSGCLASS=T,
// NOTIFY=SAPRES4, REGION=OM
/*JOBPARM SYSAFF=SC49, L=9999
//*****
//*
/** JCL JOBSTREAM NAME: JCLGEN$
//*
//*****
/** THIS JCL USES REXX TO CREATE JOBS FOR INVOKING THE
/** THE DB2 ADMINISTRATION TOOL. THESE GENERATED JOBS ARE
/** THEN SUBMITTED TO THE INTRDR.
/**
/** THIS JOBSTREAM NEEDS TO BE RUN ON THE TARGET DB2 SYSTEM BECAUSE
/** THE JCLGEN REXX ROUTINE USES THE METADATA TABLES.
/** ALTER PARAMETERS PASSED TO THE JCLGEN REXX ROUTINE TO
/** SPECIFY THIS.
//*****
//*
/** DELETE AND REDEFINE DSETS WHICH JCL AND DDL WILL BE GENERATED INTO.
//*
//DSNTIC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
DELETE (SAPRES4.MERGBLU.REXXFILE.JCLDB)
DELETE (SAPRES4.MERGBLU.REXXFILE.JCLTS)
DELETE (SAPRES4.MERGBLU.REXXFILE.JCLTB)
DELETE (SAPRES4.MERGBLU.REXXFILE.JCLIX)
DELETE (SAPRES4.MERGBLU.REXXFILE.DDLDB.UNTAIL)
DELETE (SAPRES4.MERGBLU.REXXFILE.DDLTS.UNTAIL)
DELETE (SAPRES4.MERGBLU.REXXFILE.DDLTB.UNTAIL)
DELETE (SAPRES4.MERGBLU.REXXFILE.DDLIX.UNTAIL)
SET MAXCC=0
```

```

/*
//ALLOCDSN EXEC PGM=IEFBR14
//JCLDB DD DSN=SAPRES4.MERGBLU.REXXFILE.JCLDB,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(30,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//JCLTS DD DSN=SAPRES4.MERGBLU.REXXFILE.JCLTS,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(30,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//JCLTB DD DSN=SAPRES4.MERGBLU.REXXFILE.JCLTB,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(30,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//JCLIX DD DSN=SAPRES4.MERGBLU.REXXFILE.JCLIX,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(30,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DLLDB DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLDB.UNTAIL,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DLLTS DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLTS.UNTAIL,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DLLTB DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLTB.UNTAIL,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DLLIX DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLIX.UNTAIL,
// DISP=(,CATLG,DELETE),
// UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
/*
/* GENERATE JOBS FOR INVOKING THE DB2 ADMINISTRATION TOOL
/*
/* UPDATE THE PARAMETERS FOR CALLING JCLGEN REXX ROUTINE:
/* 1. TARGET DB2 SUBSYSTEM CONTAINING METADATA TABLES
/* 2. OWNER OF THE METADATA TABLES IN TARGET SYSTEM
/* 3. PREFIX FOR THE DATASET NAMES TO GENERATE DDL & JCL INTO
/* (THIS PREFIX NEEDS TO MATCH THE NAMES USED IN THIS JCL)
/* 3. AUTHID FOR ADMIN TOOL TO USE
/* UPDATE SDSNLOAD/SDSNEXIT FOR TARGET DB2 SUBSYSTEM.
/*
//JCLGEN EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(0,LT)
//STEPLIB DD DSN=DB7T7.SDSNEXIT,DISP=SHR
// DD DSN=DB7T7.SDSNLOAD,DISP=SHR
//SYSEXEC DD DISP=SHR,DSN=SAPRES4.MERGBOOK.REXX

```

```

//JCLDB DD DISP=MOD,DSN=SAPRES4.MERGBLU.REXXFILE.JCLDB
//JCLTS DD DISP=MOD,DSN=SAPRES4.MERGBLU.REXXFILE.JCLTS
//JCLTB DD DISP=MOD,DSN=SAPRES4.MERGBLU.REXXFILE.JCLTB
//JCLIX DD DISP=MOD,DSN=SAPRES4.MERGBLU.REXXFILE.JCLIX
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%JCLGEN DB7T SAPR3 SAPRES4.MERGBLU.REXXFILE SAPRES1
//*
/** SUBMIT THE JOBS TO THE INTERNAL READER BECAUSE THEY ARE TOO LARGE
/** TO EDIT AND SUBMIT IN ISPF
/**
//INTRDRB EXEC PGM=IEBGENER,REGION=4096K,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.JCLDB
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN DD DUMMY
/**
//INTRDRTS EXEC PGM=IEBGENER,REGION=4096K,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.JCLTS
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN DD DUMMY
/**
//INTRDRTB EXEC PGM=IEBGENER,REGION=4096K,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.JCLTB
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN DD DUMMY
/**
//INTRDRIX EXEC PGM=IEBGENER,REGION=4096K,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.JCLIX
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=(LRECL=80,RECFM=FB,BLKSIZE=80)
//SYSIN DD DUMMY

```

Tailoring the output from DB2 Admin

The DDL generated by DB2 Admin needs additional tailoring before it can be executed.

TAILORTS

In Example C-8 on page 218, we provide the REXX procedure used to reduce the primary and secondary quantity allocations in the CREATE TABLESPACE statements generated by DB2 Admin. The source of this procedure (the

TAILORTS.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, "Additional material" on page 265).

Example: C-8 REXX procedure to tailor CREATE TABLESPACE statements

```

/*****/
/* REXXSQL

      REXX routine name: TAILORTS

      This routine will read through the untailed DDL file which
      contains CREATE TABLESPACE statements.

      For the CREATE TABLESPACE statement it will reduce the PRIQTY
      and SECQTY to small values.

      The tailored DDL is then written to the output file.

*****/
/* TRACE(R) */

SAY ' '
SAY 'TAILORTS executing.'
SAY ' '

/* Read the first record */
"EXECIO 0 DISKR DDLTSIN (OPEN"
"EXECIO 1 DISKR DDLTSIN"
IF rc <> 0 THEN
  DO
    SAY 'TAILORTS Error opening DDLTSIN input file. rc=' rc
    EXIT(12)
  END
ELSE NOP

/* Open output file which will contain tailored DDL */
"EXECIO 0 DISKW DDLTSOUT (OPEN"
IF rc <> 0 THEN
  DO
    SAY 'TAILORTS Error opening DDLTSOUT file. rc=' rc
    EXIT(12)
  END
ELSE NOP

DO WHILE RC = 0
  PULL currline

      /* If this line contains PRIQTY then it's for CREATE TABLESPACE.
```

```

Reduce the PRIQTY and SECQTY to small values. These values provide
adequate space for creating the empty tablespaces  */

IF WORD(currline,1) = 'PRIQTY' THEN
  DO
    currline = "PRIQTY 240 SECQTY 3600"
  END
ELSE NOP

PUSH currline
"EXECIO 1 DISKW DDLTSOUT"      /* Write curr record */
IF rc <> 0 THEN
  DO
    SAY 'TAILORTS Error writing to DDLTSOUT output file. rc=' rc
    SAY 'current line is:' currline
    EXIT(12)
  END
ELSE NOP

"EXECIO 1 DISKR DDLTSIN"      /* Read next record */
END

"EXECIO 0 DISKR DDLTSIN (FINIS" /* Close the files */
"EXECIO 0 DISKW DDLTSOUT (FINIS"

```

TAILORIX

In Example C-9, we provide the REXX procedure used to reduce the primary and secondary quantity allocations in the CREATE INDEX statements generated by DB2 Admin. The source of this procedure (the TAILORIX.RXX file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-9 REXX procedure to tailor CREATE INDEX statements

```

/*****
/* REXXSQL

```

```

REXX routine name: TAILORIX

```

```

This routine will read through and untailed DDL file which
contains CREATE INDEX statements.

```

```

For each CREATE INDEX statement it will reduce the PRIQTY
and SECQTY to small values.

```

```

The tailored DDL is then written to the output file.

```

```

*****/
/* TRACE(R) */

SAY ' '
SAY 'TAILORIX executing.'
SAY ' '

/* Read the first record */
"EXECIO 0 DISKR DDLIXIN (OPEN"
"EXECIO 1 DISKR DDLIXIN"
  IF rc <> 0 THEN
    DO
      SAY 'TAILORIX Error opening DDLIXIN input file. rc=' rc
      EXIT(12)
    END
  ELSE NOP

/* Open output file which will contain tailored DDL */
"EXECIO 0 DISKW DDLIXOUT (OPEN"
IF rc <> 0 THEN
  DO
    SAY 'TAILORIX Error opening DDLIXOUT file. rc=' rc
    EXIT(12)
  END
ELSE NOP

DO WHILE RC = 0
  PULL currline

  /* If this line contains PRIQTY then it's for CREATE INDEX.
  Reduce the PRIQTY and SECQTY to small values. These values provide
  adequate space for creating the empty indexes */
  IF WORD(currline,1) = 'PRIQTY' THEN
    DO
      currline = " PRIQTY 240 SECQTY 3600"
    END
  ELSE NOP

  PUSH currline
  "EXECIO 1 DISKW DDLIXOUT"          /* Write curr record */
  IF rc <> 0 THEN
    DO
      SAY 'TAILORIX Error writing to DDLIXOUT output file. rc=' rc
      EXIT(12)
    END
  ELSE NOP

  "EXECIO 1 DISKR DDLIXIN"          /* Read next record */
END

```

```
"EXECIO 0 DISKR DDLIXIN (FINIS"          /* Close the files */
"EXECIO 0 DISKW DDLIXOUT (FINIS"
```

TAILORTB

In Example C-10, we provide the REXX procedure used to append the OBID clause to the CREATE TABLE statements generated by DB2 Admin. The source of this procedure (the TAILORTB.RXX file) is also part of the additional material that be downloaded from the Internet (see Appendix E, "Additional material" on page 265).

Example: C-10 REXX procedure to tailor CREATE TABLE statements

```

/*****/
/* REXXSQL

REXX routine name: TAILORTB

This routine will read through the untailored DDL file which
contains CREATE TABLE statements.

For the CREATE TABLE statements it will add the OBID we want to
use when defining the table in the target system.

The tailored DDL is then written to the output file.

Parameters to call this procedure:
METADATA_SSID  Target DB2 subsystem containing metadata tables
METADATA_OWNER Owner of the metadata tables in target system

Note that this routine needs to run against the metadata
tables located on the target DB2 system. Review the parameters
to ensure this.

*****/
/* TRACE(R) */

PARSE ARG METADATA_SSID METADATA_OWNER
SAY ' '
SAY 'TAILORTB executing in DB2 subsystem' METADATA_SSID
SAY '      using metadata from table ' METADATA_OWNER'.ZMCOB_TABLES'
SAY ' '

/*****/
/* Add DSNREXX to the host command environment table if not there. */
/*****/
'SUBCOM DSNREXX'
```

```

IF RC THEN ,
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
ADDRESS DSNREXX
/*****
/* Connect to TARGET DB2 subsystem containing metadata tables */
*****/
'CONNECT' METADATA_SSID

/*****
/* Setup the cursor for reading from the ZMCOB_TABLES table. */
/* The ZMCOB_TABLES table contains one row for every database */
/* being merged. */
*****/
SQLSTMT1 = "SELECT SRCTABOBID FROM " METADATA_OWNER".ZMCOB_TABLES "
SQLSTMT1 = SQLSTMT1 " WHERE TRGSCHEMA = " ? " AND SRCTABLE = " ?
"EXECSQL DECLARE C1 CURSOR FOR S1"
IF SQLCODE <> 0 THEN
    DO
        SAY "TAILORTB from DECLARE is " SQLCODE SQLERRMC SQLERRD.5
        EXIT(12)
    END
ELSE NOP

/* Return to the TSO environment so EXECIO will work */
ADDRESS TSO

/* Open input file which contains untailed DDL and read the
first record */
"EXECIO 0 DISKRU DDLTBIN (OPEN"
"EXECIO 1 DISKRU DDLTBIN"
IF rc <> 0 THEN
    DO
        SAY 'TAILORTB Error opening DDLTBIN input file. rc=' rc
        EXIT(12)
    END
ELSE NOP

/* Open output file which will contain tailed DDL */
"EXECIO 0 DISKW DDLTBOUT (OPEN"
IF rc <> 0 THEN
    DO
        SAY 'TAILORTB Error opening DDLTBOUT file. rc=' rc
        EXIT(12)
    END
ELSE NOP

/* Read through each line of the input file */
DO WHILE RC = 0
    PULL currline

```



```

newline = currline

/* If this line contains the CREATE TABLE statements then
extract table name and creator. These will be used to determine the
OBID to be used to create the table */

IF (WORD(currline,1) = 'CREATE') & (WORD(currline,2) = 'TABLE') THEN
DO
/* SAY 'Found the CREATE TABLE LINE ' */
currcreatortable = WORD(currline,3) /*extract CREATOR.TABLE */
PARSE VAR currcreatortable currcreator '.' currtable
/* SAY 'Extracted ' currcreator 'and' currtable */
END
ELSE NOP

/* If this line starts with 'IN' and contains a ';' then it is
the final line of the CREATE TABLE statements. Call subroutine to
lookup the OBID and append the OBID clause to the line */

IF (WORD(currline,1) = 'IN') & (POS(';',currline) <> 0) THEN
DO
currobid = 'AA'
CALL OBIDLookup currcreator, currtable, currobid
/* Separate the line at the semicolon */
PARSE VAR currline startofline ';' endofline
newline = startofline ' OBID ' currobid ';'
END
ELSE NOP

/* Write the record, which may or may not have been changed
to the output file */
PUSH newline
"EXECIO 1 DISKW DDLTBOUT"
IF rc <> 0 THEN
DO
SAY 'TAILORTB Error writing DDLTBOUT file. rc=' rc
EXIT(12)
END
ELSE NOP

"EXECIO 1 DISKRU DDLTBIN" /* Read next input record */
END

"EXECIO 0 DISKRU DDLTBIN (FINIS" /* Close the files */
"EXECIO 0 DISKW DDLTBOUT (FINIS"
EXIT
/*****
/* SUBROUTINES */
*****/

```

```

/* This subroutine will query the SAPR3.ZMCOB_TABLES table to */
/* obtain the OBID to be used to create the table */
/*****/
OBIDLOOKUP:
ARG CURRCREATOR, CURRTABLE
  CURROBID = 'BB'

/* Transfer to DSNREXX environment */
ADDRESS DSNREXX

/* Prepare the SQL statement */
"EXECSQL PREPARE S1 FROM :SQLSTMT1"
IF SQLCODE <> 0 THEN
  DO
    SAY "TAILORTB from PREPARE is " SQLCODE SQLERRMC SQLERRD.5
    SAY "TAILORTB CURRCREATOR " CURRCREATOR " CURRTABLE "CURRTABLE
    EXIT(12)
  END
ELSE NOP

/* Setup the host variables.
The creator and table names may have single or double quotes around
them. The SQL query requires single quotes. So, use the STRIP command
to remove whatever quotes may be on it. Then add the single quotes
we require. */

CREATORNOQUOTES = STRIP(CURRCREATOR,B,'"') /* remove double quotes */
CREATORNOQUOTES = STRIP(CREATORNOQUOTES,B,'"') /* remove single quotes */
CREATORINQUOTES = "'"CREATORNOQUOTES"'" /* add single quotes */

TABLENOQUOTES = STRIP(CURRTABLE,B,'"') /* remove double quotes */
TABLENOQUOTES = STRIP(TABLENOQUOTES,B,'"') /* remove single quotes */
TABLEINQUOTES = "'"TABLENOQUOTES"'" /* add single quotes */

/* Open the cursor */
"EXECSQL OPEN C1 USING :CREATORINQUOTES, :TABLEINQUOTES "
IF SQLCODE <> 0 THEN
  DO
    SAY "TAILORTB from OPEN C1 is " SQLCODE SQLERRMC SQLERRD.5
    SAY "TAILORTB CREATORINQUOTES " CREATORINQUOTES
    SAY "TAILORTB TABLEINQUOTES " TABLEINQUOTES
    EXIT(12)
  END
ELSE NOP

/* Fetch the cursor */
"EXECSQL FETCH C1 INTO :CURROBID"
IF SQLCODE <> 0 THEN
  DO

```

```

        SAY "TAILORTB from FETCH C1 is " SQLCODE SQLERRMC SQLERRD.5
        SAY "TAILORTB CURRCREATOR " CURRCREATOR
        SAY "TAILORTB CURRTABLE " CURRTABLE
        SAY "TAILORTB CREATORINQUOTES " CREATORINQUOTES
        SAY "TAILORTB TABLEINQUOTES " TABLEINQUOTES
        EXIT(12)
        END
ELSE NOP

/* Close the cursor */
"EXECSQL CLOSE C1"
IF SQLCODE <> 0 THEN
    DO
        SAY "TAILORTB from CLOSE C1 is " SQLCODE SQLERRMC SQLERRD.5
        SAY "TAILORTB CREATORINQUOTES " CREATORINQUOTES
        SAY "TAILORTB TABLEINQUOTES " TABLEINQUOTES
        EXIT(12)
    END
ELSE NOP

/* Return to the TSO environment and exit subroutine */
ADDRESS TSO
RETURN(CURROBID)

```

JCL to run the tailoring REXX procedures

In Example C-11, we provide the JCL used to run the tailoring REXX procedures. The source of this JCL (the TAILOR\$.JCL file) is also part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example: C-11 JCL to run the tailoring REXX procedures

```

// NOTIFY=SAPRES4,REGION=0M
/*JOBPARM SYSAFF=SC49,L=9999
//*****
//*
/* JCL JOBSTREAM NAME: TAILOR$
/*
//*****
/* THIS JCL USES REXX TO TAILOR DDL WHICH WAS GENERATED BY THE DB2
/* ADMINISTRATION TOOL. TAILORING DONE INCLUDES:
/* - REDUCE THE PRIMARY AND SECONDARY QUANTITY ALLOCATIONS
/* ON CREATE TABLESPACE AND CREATE INDEX STATEMENTS
/* - INCLUDE THE OBID CLAUSE ON EACH CREATE TABLE STATEMENT
/*
/* THIS JOBSTREAM NEEDS TO BE RUN ON THE TARGET DB2 SYSTEM BECAUSE
/* THE TAILOR REXX ROUTINE USES THE METADATA TABLES TO DETERMINE

```

```

/** THE OBID FOR THE NEW TABLES.
/**
/** ALTER PARAMETERS PASSED TO THE TAILOR REXX ROUTINE TO
/** SPECIFY THIS.
/*******
/**
/** DELETE AND DEFINE DATASETS WHICH THE DDL WILL BE TAILORED INTO
/**
//DSNTIC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
        DELETE (SAPRES4.MERGBLU.REXXFILE.DDLDB.TAILOR)
        DELETE (SAPRES4.MERGBLU.REXXFILE.DDLTS.TAILOR)
        DELETE (SAPRES4.MERGBLU.REXXFILE.DDLTB.TAILOR)
        DELETE (SAPRES4.MERGBLU.REXXFILE.DDLIX.TAILOR)
        SET MAXCC=0
/*
//ALLOCDSN EXEC PGM=IEFBR14
//DDLDB DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLDB.TAILOR,
//      DISP=(,CATLG,DELETE),
//      UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DDLTS DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLTS.TAILOR,
//      DISP=(,CATLG,DELETE),
//      UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DDLTB DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLTB.TAILOR,
//      DISP=(,CATLG,DELETE),
//      UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
//DDLIX DD DSN=SAPRES4.MERGBLU.REXXFILE.DDLIX.TAILOR,
//      DISP=(,CATLG,DELETE),
//      UNIT=3390,VOL=SER=TOTDB7,SPACE=(CYL,(15,15)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400)
/**
/** DATABASE DDL REQUIRES NO TAILORING SO JUST COPY TO THE OUTPUT FILE
/**
//COPYDB EXEC PGM=IEBGENER,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.DDLDB.UNTAIL
//SYSUT2 DD DISP=SHR,DSN=SAPRES4.MERGBLU.REXXFILE.DDLDB.TAILOR
//SYSIN DD DUMMY
/**
/** RUN THE REXX ROUTINES TO TAILOR THE DDL
/**
/** UPDATE SDSNLOAD/SDSNEXIT FOR TARGET DB2 SUBSYSTEM.
/** UPDATE THE PARAMETERS FOR CALLING TAILORTB REXX ROUTINE:
/** 1. TARGET DB2 SUBSYSTEM CONTAINING METADATA TABLES

```

```

/** 2. OWNER OF THE METADATA TABLES IN TARGET SYSTEM
/** (THE OTHER REXX ROUTINES DON'T ACCESS DB2 AND DONT REQUIRE
/** AND PARAMETERS)
/**
//TAILOR EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(0,LT)
//STEPLIB DD DSN=DB7T7.SDSNEXIT,DISP=SHR
// DD DSN=DB7T7.SDSNLOAD,DISP=SHR
//SYSEXEC DD DISP=SHR,DSN=SAPRES4.MERGBOOK.REXX
//DDLTSIN DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLTS.UNTAIL
//DDLTSOUT DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLTS.TAILOR
//DDLTBIN DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLTB.UNTAIL
//DDLTBOUT DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLTB.TAILOR
//DDLIXIN DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLIX.UNTAIL
//DDLIXOUT DD DISP=OLD,DSN=SAPRES4.MERGBLU.REXXFILE.DDLIX.TAILOR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
%TAILORTS
%TAILORTB DB7T SAPR3
%TAILORIX
/*

```

Sample DDL for creating objects

This section contains samples of the DDL that is generated and tailored. We use this to define the source objects in the target DB2 system.

Example C-12 is a sample of the CREATE STOGROUP DDL for two stogroups. These stogroups are SAPSTD and SAPBTD in the source system. They are created as BLUSTD and BLUBTD in the target system, according to the MCODE naming convention for storage groups.

Example: C-12 DDL for CREATE STOGROUP

```

CREATE STOGROUP BLUSTD VOLUMES('*') VCAT MCDBLU ;
CREATE STOGROUP BLUBTD VOLUMES('*') VCAT MCDBLU ;

```

Example C-13 on page 228 is a sample of the CREATE DATABASE DDL. The database name is changed during the generation process to avoid name clashes with databases that already exist in the target system.

Example: C-13 DDL for CREATE DATABASE

```
-----  
-- Database=A000X8X4 Stogroup=BLUSTI  
-----  
--  
CREATE DATABASE A000X8X4  
  BUFFERPOOL BP3  
  INDEXBP BP0  
  CCSID ASCII  
  STOGROUP BLUSTI ;  
--  
COMMIT;
```

Example C-14 is a sample of the CREATE TABLE DDL. Note that the primary and secondary quantities have been reduced to conserve disk space during the merge procedure.

Example: C-14 DDL for CREATE TABLESPACE

```
-----  
-- DATABASE=A000X8X4 STOGROUP=BLUSTD  
-- TABLESPACE=A000X8X4.ABAPTREE  
-----  
--
```

```
CREATE TABLESPACE ABAPTREE  
  IN A000X8X4  
  USING STOGROUP BLUSTD  
PRIQTY 240 SECQTY 3600  
  FREEPAGE 3 PCTFREE 20  
  SEGSIZE 4  
  BUFFERPOOL BP2  
  LOCKSIZE ROW  
  CCSID ASCII;  
--  
COMMIT;
```

Example C-15 is a sample of the CREATE TABLE DDL. We include the OBID clause so that the table will retain its OBID during the merge procedure.

Example: C-15 DDL for CREATE TABLE

```
-----  
-- TABLE=SAPBLU.ABAPTREE IN A000X8X4.ABAPTREE  
-----  
--  
CREATE TABLE SAPBLU.ABAPTREE  
  (ID VARCHAR(6) NOT NULL WITH DEFAULT '000000' ,  
  REPNAME VARCHAR(40) NOT NULL WITH DEFAULT ' ' ,  
  OBID OBID);
```

```

        CONSTRAINT ID PRIMARY KEY (ID))
    IN A000X8X4.ABAPTREE  OBID 3 ;
--
COMMIT;

```

Example C-16 is a sample of the CREATE INDEX DDL. We reduce the primary and secondary quantity to conserve disk space during the merge procedure.

Example: C-16 DDL for CREATE INDEX

```

-----
-- DATABASE=A000X8X4
--   INDEX=SAPBLU.ABAPTREE~0           ON SAPBLU.ABAPTREE
-----
--
CREATE UNIQUE INDEX SAPBLU."ABAPTREE~0"
  ON SAPBLU.ABAPTREE
  (ID          ASC)
  USING STOGROUP BLUSTI
  PRIQTY 240 SECQTY 3600
  CLUSTER
  BUFFERPOOL BP3
  COPY NO;
--
COMMIT;

```

Example C-17 is a sample of the CREATE VIEW DDL.

Example: C-17 DDL for CREATE VIEW

```

CREATE
VIEW
"APPL_DOMA"
(
  "DEVCLASS",
  "OBJ_NAME"
)
AS
SELECT
T0001."DEVCLASS",
T0001."OBJ_NAME"
FROM
  "TADIR"
T0001
WHERE
T0001."PGMID"
=
'R3TR'
AND

```

```
T0001."OBJECT"  
=  
'DOMA'  
;
```

Using filler tablespaces to reserve OBIDs

Because an OBID cannot be specified on CREATE TABLESPACE, and we know that we want specific OBIDS to be available when creating tables, we need to influence which OBIDs are used when tablespaces are created. We use the following method:

1. Determine the largest OBID that is used within a database for tables. Call that n.
2. Create enough “filler” tablespaces in the database to use n OBIDs.
3. Then, create the real tablespaces, knowing their OBIDs will be greater than n.
4. Drop the filler tablespaces, making n the lowest OBIDs available.
5. We can then create the tables knowing that the OBIDs we require will be available.

The REXX procedure FILLERTS generates the DDL to create the correct number of filler tablespaces in each database. The source of this REXX procedure (the FILLERTS.RXX file) and the JCL to execute it (the FILLERS\$.JCL file), are part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Example C-18 is a sample of the CREATE TABLESPACE DDL we use to create filler tablespaces for reserving OBIDs within a database. Creating the nine tablespaces, as shown, reserve 18 OBIDs because each tablespace occupies two OBIDs.

Example: C-18 DDL for CREATE TABLESPACE for filler tablespaces

```
CREATE TABLESPACE FIL1 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL2 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL3 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL4 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL5 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL6 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL7 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL8 IN A000XAAA DEFINE NO;  
CREATE TABLESPACE FIL9 IN A000XAAA DEFINE NO;
```

The FILLERTS routine also generates DDL to drop the filler tablespaces, as follows. Example C-19 is a sample of the DROP TABLESPACE DDL used to drop filler tablespaces.

Example: C-19 DDL for DROP TABLESPACE for filler tablespaces

```
DROP TABLESPACE A000XAAA . FIL1 ;  
DROP TABLESPACE A000XAAA . FIL2 ;  
DROP TABLESPACE A000XAAA . FIL3 ;  
DROP TABLESPACE A000XAAA . FIL4 ;  
DROP TABLESPACE A000XAAA . FIL5 ;  
DROP TABLESPACE A000XAAA . FIL6 ;  
DROP TABLESPACE A000XAAA . FIL7 ;  
DROP TABLESPACE A000XAAA . FIL8 ;  
DROP TABLESPACE A000XAAA . FIL9 ;
```

Archived

Merge in place: Migrating the data

This appendix contains material that supports the migration steps of the merge in place process. This includes SQL statements, REXX procedures, and JCL to perform the following tasks:

- ▶ Issuing DB2 START and STOP DATABASE commands
- ▶ Executing REPAIR on page set header pages
- ▶ Deleting created target data sets and renaming source data sets
- ▶ Executing REPAIR LEVELID on tablespaces and indexes
- ▶ Generating DDL for views
- ▶ Generating DDL for altering object sizes

The source of all these SQL statements, REXX procedures, and JCL, as well as parameter files and JCL skeletons, are part of the additional material that can be downloaded from the Internet (see Appendix E, “Additional material” on page 265).

Running the migration procedures

In this section, we describe the environment in which the migration procedures run.

PDS libraries used

The migration procedures refer to a list of partitioned data set (PDS) libraries:

SAPRES5.REXX.BLUCNTL	Source of the REXX procedures and JCL to execute them
SAPRES5.REXX.BLUJOBS	JCL generated by the REXX procedures
SAPRES5.REXX.BLUPARM	Input parameters
SAPRES5.REXX.BLUQURS	Output generated by the PQUERYTB procedure
SAPRES5.REXX.BLUQUERY	SQL queries used by PQUERYTB
SAPRES5.REXX.BLUSKEL	Skeletons for JCL generation
SAPRES5.REXX.BLUSTIN	Statements generated by the REXX procedures

In Example D-1, we provide a sample JCL to allocate these PDS libraries.

Example: D-1 Allocating PDS libraries for migration procedures

```
//ALLOCPDS JOB (999,P0K),CLASS=A,MSGCLASS=T,  
//          NOTIFY=&SYSUID,TIME=1440,REGION=0M  
//IKJEFT01 EXEC PGM=IKJEFT1A  
//BLUCNTL DD DSN=SAPRES5.REXX.BLUCNTL,  
//          UNIT=SYSDA,DISP=(,CATLG),  
//          SPACE=(TRK,(50,10,10)),  
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD DUMMY  
//*  
//BLUJOBS DD DSN=SAPRES5.REXX.BLUJOBS,  
//          UNIT=SYSDA,DISP=(,CATLG),  
//          SPACE=(TRK,(50,10,10)),  
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD DUMMY  
//*  
//BLUPARM DD DSN=SAPRES1.REXX.BLUPARM,  
//          UNIT=SYSDA,DISP=(,CATLG),  
//          SPACE=(TRK,(50,10,10)),  
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD DUMMY
```

```

//*
//BLUQURY DD DSN=SAPRES1.REXX.BLUQURY,
//          UNIT=SYSDA,DISP=(,CATLG),
//          SPACE=(TRK,(50,10,10)),
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//*
//BLUQURS DD DSN=SAPRES1.REXX.BLUQURS,
//          UNIT=SYSDA,DISP=(,CATLG),
//          SPACE=(CYL,(100,10,10)),
//          DSORG=PO,RECFM=FB,LRECL=1540,BLKSIZE=6160
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//*
//BLUSKEL DD DSN=SAPRES1.REXX.BLUSKEL,
//          UNIT=SYSDA,DISP=(,CATLG),
//          SPACE=(TRK,(50,10,10)),
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//*
//BLUSTIN DD DSN=SAPRES1.REXX.BLUSTIN,
//          UNIT=SYSDA,DISP=(,CATLG),
//          SPACE=(CYL,(20,10,10)),
//          DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6160
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD DUMMY
//*
```

Input parameters for all migration procedures

The migration procedures use a list of input parameters specified by keywords during the invocation or in the following:

- ▶ A PDS member specifically named during the invocation LOCAL=
- ▶ A PDS member containing parameters globally valid for all procedures GLOBAL=

The precedence of parameters is as follows:

1. Provided during invocation.
2. If not provided, some default values are used.
3. Provided by LOCAL=<local>.
4. Provided by GLOBAL=<global>.

In Example D-2, we provide a sample global parameter file.

Example: D-2 Global parameter file BLUGLOB

```
/* comments have to start with '/*' */
sid=BLU /* SAP SYSTEM ID */
tssid=DB7T /* target DB2 SUBSYSTEM id */
sssid=DB7S /* source DB2 SUBSYSTEM id */
tsysid=SC49 /* target MVS system id */
sssysid=SC42 /* source MVS system id */
tvcat=MCDBLU /* target VCAT name */
svcat=SAPBLU /* source VCAT name */
tschema=SAPBLU /* target schema name */
njvsalt=1 /* # of IDCAMS ALTER jobs */
njrepob=1 /* # of REPAIR OBID jobs */
njreplv=1 /* # of REPAIR LVLID jobs */
njdbcom=1 /* # of REPAIR DB2COMjobs */
njdbddl=1 /* # of REPAIR DB2DDLjobs */
dsnload=DB7S7.SDSNLOAD /* DSN Loadlib */
dsnextit=DB7S7.SDSNEXIT /* DSN Exitlib */
inpparm=SAPRES5.REXX.BLUPARM /* inplib for REXX parms */
inpqurs=SAPRES5.REXX.BLUQURS /* inplib for QUERY result */
inpskel=SAPRES5.REXX.BLUSKEL /* inplib for JCL skeleton */
outparm=SAPRES5.REXX.BLUSTIN /* outlib for utility parms*/
outjobs=SAPRES5.REXX.BLUJOBS /* outlib for jcl */
prognam=SAPRES5 /* name of Programmer for */
/* JOB card */
outqurs=SAPRES5.REXX.BLUQURS /* outlib for QUERY results*/
outskel=SAPRES5.REXX.BLUSKEL /* outlib for JCL skeletons*/
skjjobc=SKGJJJBC /* name of JOBCARD skeleton*/
ksidca=SKGSIDCA /* name of IDCAMS skeleton*/
ksdsnu=SKGSDSNU /* name of DSNU skeleton*/
ksdbco=SKGSDBCO /* name of skeleton*/
ksdbdd=SKGSDBDD /* name of DSNTIAD skeleton*/
```

Extracting information from the metadata tables

The migration procedures work on information extracted from the metadata tables by the REXX procedure PQUERYTB and different SQL query statements.

\$ZMCDBSR

The SQL query \$ZMCDBSR, as shown in Example D-3 on page 237, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the databases.

Example: D-3 SQL query to retrieve information about databases

```
-- Query against SAPR3.ZM_COD_DATABASE
-- used in SG24-6914
select
  ZM.SRCSID as ZM_SRCSID ,
  ZM.SRCDATABASE as ZM_SRCDATABASE ,
  ZM.SRCSHEMA as ZM_SRCSHEMA ,
  ZM.SRCSSTOGROUP as ZM_SRCSSTOGROUP ,
  ZM.SRCSRELEASE as ZM_SRCSRELEASE ,
  ZM.TRGSID as ZM_TRGSID ,
  ZM.TRGDATABASE as ZM_TRGDATABASE ,
  ZM.TRGSHEMA as ZM_TRGSHEMA ,
  ZM.TRGSTOGROUP as ZM_TRGSTOGROUP ,
  ZM.TRGRELEASE as ZM_TRGRELEASE
from
  SAPR3.ZM_COD_DATABASE as ZM
order by
  ZM_SRCDATABASE
```

\$ZMCINSR

The SQL query \$ZMCINSR, as shown in Example D-4, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the indexes.

Example: D-4 SQL query to retrieve information about indexes

```
-- Query against SAPR3.ZM_COD_DATABASE and SAPR3.ZM_COD_INDEXES
-- used in SG24-6914
select
  IN.SRCDATABASE as ZM_SRCDATABASE ,
  IN.SRCINDEXSPACE as ZM_SRCINDEXSPACE ,
  IN.SRCPART as ZM_SRCPART ,
  IN.SRCINDEX as ZM_SRCINDEX ,
  IN.SRCSHEMA as ZM_SRCSHEMA ,
  IN.SRCSID as ZM_SRCSID ,
  IN.SRCSBID as ZM_SRCSBID ,
  IN.SRCSPACE as ZM_SRCSPACE ,
  IN.SRCISOBID as ZM_SRCISOBID ,
  IN.SRCIPREFIX as ZM_SRCIPREFIX ,
  IN.SRCPRIQTY as ZM_SRCPRIQTY ,
  IN.SRCSSTOGROUP as ZM_SRCSSTOGROUP ,
  IN.SRSCAT as ZM_SRSCAT ,
  IN.TRGINDEX as ZM_TRGINDEX ,
  IN.TRGINDEXSPACE as ZM_TRGINDEXSPACE ,
  IN.TRGSHEMA as ZM_TRGSHEMA ,
  DB.TRGDATABASE as ZM_TRGDATABASE ,
```

```

        IN.TRGSSID      as ZM_TRGSSID ,
        IN.TRGDBID     as ZM_TRGDBID ,
        IN.TRGISOBID   as ZM_TRGISOBID ,
        IN.TRGIPREFIX  as ZM_TRGIPREFIX ,
        IN.TRGSTOGROUP as ZM_TRGSTOGROUP ,
        IN.TRGVCAT     as ZM_TRGVCAT
from
        SAPR3.ZMCOINDEXES as IN,
        SAPR3.ZMCO_DATABASE as DB
where
        IN.SRCDBNAME = DB.SRCDBNAME
order by
        ZM_SRCDBNAME ,
        ZM_SRCINDEXSPACE ,
        ZM_SRCPART

```

\$ZMCTSSR

The SQL query \$ZMCTSSR, as shown in Example D-5, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the tablespaces.

Example: D-5 SQL query to retrieve information about tablespaces

```

-- Query against ZMCO_TABLESPACE and ZMCO_DATABASE
-- used in SG24-6914
select
        TS.SRCSSID      as ZM_SRCSSID ,
        TS.SRCTSNAME    as ZM_SRCTSNAME ,
        TS.SRCDBNAME    as ZM_SRCDBNAME ,
        TS.SRCDBID     as ZM_SRCDBID ,
        TS.SRCPSID     as ZM_SRCPSID ,
        TS.SRCRELEASE  as ZM_SRCRELAESE ,
        TS.SRCPART     as ZM_SRCPART ,
        TS.SRCSPACE    as ZM_SRCSPACE ,
        TS.SRCIPREFIX  as ZM_SRCIPREFIX ,
        TS.SRCSHEMA    as ZM_SRCSHEMA ,
        TS.SRCSTOGROUP as ZM_SRCSTOGROUP ,
        TS.SRCVCAT     as ZM_SRCVCAT ,
        TS.TRGSSID     as ZM_TRGSSID ,
        TS.TRGPSID     as ZM_TRGPSID ,
        TS.TRGTSNAME   as ZM_TRGTSNAME ,
        DB.TRGDBNAME   as ZM_TRGDBNAME ,
        TS.TRGDBID     as ZM_TRGDBID ,
        TS.TRGRELEASE  as ZM_TRGRELAESE ,
        TS.TRGIPREFIX  as ZM_TRGIPREFIX ,
        TS.TRGSHEMA    as ZM_TRGSHEMA ,
        TS.TRGSTOGROUP as ZM_TRGSTOGROUP ,

```



```

    TS.TRGVCAT      as ZM_TRGVCAT
  from
    SAPR3.ZMCOB_TABLESPACE TS,
    SAPR3.ZMCOB_DATABASE   DB
  where
    DB.SRCDBNAME = TS.SRCDBNAME
  order by
    ZM_SRCDBNAME,
    ZM_SRCPSID,
    ZM_SRCPART

```

PQUERYTB

The REXX procedure PQUERYTB can extract information from the metadata tables. It takes as input SQL queries located by default in SAPRES5.REXX.BLUQURY. The output is generated in SAPRES5.REXX.BLUQURS.

Once the metadata tables are populated with all the information from the source and the target system, we use the following JCL, as shown in Example D-6, for invoking the REXX procedure PQUERYTB. Note the PQUERYTB is run against the *target* DB2 subsystem.

Example: D-6 Invocation of PQUERYTB

```

//SAPRES5A JOB (999,P0K), 'RUNREXX      ',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM,RESTART=HERE
/*JOBPARM  SYSAFF=SC42,L=9999
// JCLLIB ORDER=(DB7SU.PROCLIB)
//HERE EXEC PGM=IEFBR14
//*
//RUNREXX EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DB7S7.SDSNEXIT,DISP=SHR
//          DD DSN=DB7S7.SDSNLOAD,DISP=SHR
//SYSEXEC  DD DISP=SHR,DSN=SAPRES5.REXX.BLUCNTL
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
%PQUERYTB INPMEMB=$ZMCDBSR DB2=T
%PQUERYTB INPMEMB=$ZMCINSR DB2=T
%PQUERYTB INPMEMB=$ZMCTSSR DB2=T

```

Example D-7 on page 240 shows the log of the REXX procedure PQUERYTB when we use as the input member (parameter INPMEMB) the SQL query \$ZMCTSSR.

Example: D-7 Log of PQUERYTB

```
READY
%PQUERYTB INPMEMB=$ZMCTSSR DB2=T
MCD0014I Procedure PQUERYTB Version 1.1.0
MCD0009I Procedure PQUERYTB in Lib SYSEXEC
MCD0021W Input Arg parmlib replaced by SAPRES5.REXX.BLUPARM
MCD0021W Input Arg globlal replaced by BLUGLOB
MCD0021W Input Arg outmemb replaced by $ZMCTSSR
MCD0021W Input Arg inpqry replaced by SAPRES5.REXX.BLUQUERY
MCD0022I Input Arg db2 is T
MCD0022I Input Arg inpqry is SAPRES5.REXX.BLUQUERY
MCD0022I Input Arg inpmemb is $ZMCTSSR
MCD0022I Input Arg outqurs is SAPRES5.REXX.BLUQRS
MCD0022I Input Arg outmemb is $ZMCTSSR
MCD0022I Input Arg sid is BLU
MCD0022I Input Arg sssid is DB7S
MCD0022I Input Arg tssid is DB7T
MCD0040I SQLCMD connect SUBSYSTEM DB7T
MCD0041I SQLCMD connect RC 0
MCD0046I Records fetched = 3347
MCD0007I Procedure PQUERYTB RC = 0
MCD0013I Elapsed Time 38 in seconds
READY
END
```

Example D-8 shows the output generated by the REXX procedure PQUERYTB.

Example: D-8 Output generated by PQUERYTB

```
EDIT      SAPRES5.REXX.BLUQRS($ZMCTSSR) - 01.00          Columns 00001 00072
Command ==>                                           Scroll ==> CSR
000001 -- Generated by Procedure PQUERYTB in Lib SYSEXEC --
000002 -- Date 16/12/02 Time 07:05:26 --
000003 -- using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM --
000004 -- using QUERY $ZMCTSSR in library SAPRES5.REXX.BLUQUERY --
000005 -- using DB2 Subsystem DBSSN --
000006 -- Query against ZMCOB_TABLESPACE and ZMCOB_DATABASE --
000007 -- used in SG24-6914 --
000008 -- select --
000009 -- TS.SRCSSID as ZM_SRCSSID , --
000010 -- TS.SRCTSNAME as ZM_SRCTSNAME , --
000011 -- TS.SRCDBNAME as ZM_SRCDBNAME , --
000012 -- TS.SRCDBID as ZM_SRCDBID , --
000013 -- TS.SRCPSID as ZM_SRCPSID , --
000014 -- TS.SRCRELEASE as ZM_SRCRELAESE , --
000015 -- TS.SRCPART as ZM_SRCPART , --
000016 -- TS.SRCSPACE as ZM_SRCSPACE , --
000017 -- TS.SRCIPREFIX as ZM_SRCIPREFIX , --
```

```

000018 -- TS.SRCSHEMA as ZM_SRCSHEMA , --
000019 -- TS.SRCSTOGROUP as ZM_SRCSTOGROUP , --
000020 -- TS.SRCVCAT as ZM_SRCVCAT , --
000021 -- TS.TRGSSID as ZM_TRGSSID , --
000022 -- TS.TRGPSID as ZM_TRGPSID , --
000023 -- TS.TRGTSNAME as ZM_TRGTSNAME , --
000024 -- DB.TRGDBNAME as ZM_TRGDBNAME , --
000025 -- TS.TRGDBID as ZM_TRGDBID , --
000026 -- TS.TRGRELEASE as ZM_TRGRELAESE , --
000027 -- TS.TRGIPREFIX as ZM_TRGIPREFIX , --
000028 -- TS.TRGSCHEMA as ZM_TRGSCHEMA , --
000029 -- TS.TRGSTOGROUP as ZM_TRGSTOGROUP , --
000030 -- TS.TRGVCAT as ZM_TRGVCAT --
000031 -- from --
000032 -- SAPR3.ZMCD_TABLESPACE TS, --
000033 -- SAPR3.ZMCD_DATABASE DB --
000034 -- where --
000035 -- DB.SRCDBNAME = TS.SRCDBNAME --
000036 -- order by --
000037 -- ZM_SRCDBNAME, --
000038 ; ZM_SRCSSID ; ZM_SRCTSNAME ; ZM_SRCDBNAME ; ZM_SRCDBID ; ZM_SRCPSID ; ...
000039 ; DB7S ; ABAPTREE ; A000X000 ; 2395 ; 2 ; 710 ; 0 ; 144 ; I ; SAPR3 ; ...
000040 ; DB7S ; TLANTEST ; A000X019 ; 2394 ; 2 ; 710 ; 0 ; -1 ; I ; SAPR3 ; ...
000041 ; DB7S ; ADR12S ; A000X01I ; 2393 ; 2 ; 710 ; 0 ; -1 ; I ; SAPR3 ; ...
000042 ; DB7S ; TRESEPR ; A000X04Z ; 2392 ; 2 ; 710 ; 0 ; -1 ; I ; SAPR3 ; ...
.....
.....

```

Issuing DB2 START and STOP DATABASE commands

Before, during, and after migration, we must issue DB2 START and STOP DATABASE commands, with or without a special access mode.

PDSNDBST

The REXX procedure PDSNDBST generates the JCL and control statements to issue DB2 START and STOP DATABASE commands.

Example D-9 on page 242 shows the invocation of the REXX procedure PDSNDBST. By default, it uses \$ZMCDBSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCDBSR as the input member).

Example: D-9 Invocation of PDSNDBST

```
%PDSNDBST LOCAL=BLUSDA
%PDSNDBST LOCAL=BLUSDU
%PDSNDBST LOCAL=BLUSDZ
%PDSNDBST LOCAL=BLUTDA
%PDSNDBST LOCAL=BLUTDU
%PDSNDBST LOCAL=BLUTDZ
```

Example D-10 gives an example of the parameter file BLUSDA.

Example: D-10 Parameter file BLUSDA used by PDSNDBST

```
/* comments have to start with '/*' */
db2=S /* DB2 source or target */
type=A /* A=Start Database */
dsnload=DB7S7.SDSNLOAD /* DSN Loadlib */
dsnexit=DB7S7.SDSNEXIT /* DSN Exitlib */
```

The other files are for the different types:

- ▶ T stands for the target database subsystem.
- ▶ S stands for the source database system.
- ▶ A stands for -START DATABASE(dbname).
- ▶ U stands for -START DATABASE(dbname) ACCESS(UT).
- ▶ Z stands for -STOP DATABASE(dbname).

Example D-11 shows the log of the REXX procedure PDSNDBST when we use BLUSDA as the local parameter file.

Example: D-11 Log of PDSNDBST

```
READY
%PDSNDBST LOCAL=BLUSDA
MCD0008I PDSNDBST 1.1.0
MCD0009I Procedure SYSEXEC in Lib
MCD0021W Input Arg inpmemb replaced by $ZMCDBSR
MCD0021W Input Arg parmllib replaced by SAPRES5.REXX.BLUPARM
MCD0021W Input Arg global replaced by BLUGLOB
MCD0021W Input Arg infix replaced by SDA
MCD0021W Input Arg program replaced by SAPRES5
MCD0022I Input Arg db2 is S
MCD0022I Input Arg dsnexit is DB7S7.SDSNEXIT
MCD0022I Input Arg dsnload is DB7S7.SDSNLOAD
MCD0022I Input Arg infix is SDA
MCD0022I Input Arg inpmemb is $ZMCDBSR
MCD0022I Input Arg inpqurs is SAPRES5.REXX.BLUQRS
```

```

MCD0022I Input Arg inpskel is SAPRES5.REXX.BLUSKEL
MCD0022I Input Arg njdbcom is 1
MCD0022I Input Arg outjobs is SAPRES5.REXX.BLUJOBS
MCD0022I Input Arg outparm is SAPRES5.REXX.BLUSTIN
MCD0022I Input Arg program is SAPRES5
MCD0022I Input Arg sid is BLU
MCD0022I Input Arg skeldbc is SKGSDBCO
MCD0022I Input Arg skeljob is SKGJJJOB
MCD0022I Input Arg sssid is DB7S
MCD0022I Input Arg ssysid is SC42
MCD0022I Input Arg svcat is SAPBLU
MCD0022I Input Arg tssid is DB7T
MCD0022I Input Arg tsysid is SC49
MCD0022I Input Arg tvcat is MCDBLU
MCD0022I Input Arg type is A
MCD0037I # databases 3337 processed and written to BLUSDA01
MCD0038I # databases 3337 are read
MCD0007I Procedure PDSNDBST RC = 0
MCD0013I Elapsed Time 2 in seconds
READY
END

```

Example D-12 shows the JCL generated by the REXX procedure PDSNDBST.

Example: D-12 JCL generated by PDSNDBST

```

EDIT          SAPRES5.REXX.BLUJOBS(BLUSDA01) - 01.00          Columns 00001 00072
Command ==>                                          Scroll ==> CSR
000001 //BLUSDA01 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=0M,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC42,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PDSNDBST in Lib SYSEXEC
000013 /* Date 17/12/02 Time 10:36:28
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCDBSR in library SAPRES5.REXX.BLUQRS
000016 //BLUSDA01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
000017 //STEPLIB DD DSN=DB7S7.SDSNLOAD,
000018 //          DISP=SHR
000019 //          DD DSN=DB7S7.SDSNEXIT,
000020 //          DISP=SHR
000021 //SYSTSPRT DD SYSOUT=*

```

```
000022 //SYSPRINT DD SYSOUT=*
000023 //SYSUDUMP DD SYSOUT=*
000024 //SYSYSIN DD DSN=SAPRES5.REXX.BLUSTIN(BLUSDA01),
000025 // DISP=SHR
```

Example D-13 shows sample statements created by the REXX procedure PDSNDBST.

Example: D-13 Sample statements created by PDSNDBST

```
EDIT          SAPRES5.REXX.BLUSTIN(BLUSDA01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 DSN SYSTEM(DB7S)
000002 -START DATABASE(A000X000)
000003 -START DATABASE(A000X019)
000004 -START DATABASE(A000X01I)
.....
002617 -START DATABASE(A223XQZQ)
.....
003338 -START DATABASE(U020X8AV)
```

Executing REPAIR on page set header pages

For a source page set (tablespace or indexspace) to be accessed by the target DB2 subsystem, the DBID and PSID on the header page of each page set must be changed to match the DBID and PSID that are in the target DB2 catalog. REXX procedures PREPTSOB and PREPINOB generate the REPAIR statements to achieve this.

PREPTSOB

The REXX procedure PREPTSOB generates REPAIR statements for tablespaces. By default, it uses \$ZMCTSSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCTSSR as the input member).

Example D-14 shows the invocation of the REXX procedure PREPTSOB.

Example: D-14 Invocation of PREPTSOB

```
%PREPTSOB
```

Note that no parameters are given on the invocation. Looking at the log of the REXX procedure PREPTSOB, as shown in Example D-15, we see that the following parameters are automatically replaced by their default values before looking at the values specified in BLUGLOB:

- ▶ inpmemb
- ▶ infix
- ▶ db2
- ▶ parmlib
- ▶ global
- ▶ program

Example: D-15 Log of PREPTSOB

```

READY
%PREPTSOB
MCD0008I PREPTSOB 1.1.0
MCD0009I Procedure PREPTSOB in Lib SYSEXEC
MCD0021W Input Arg inpmemb replaced by $ZMCTSSR
MCD0021W Input Arg infix replaced by TSO
MCD0021W Input Arg db2 replaced by S
MCD0021W Input Arg parmlib replaced by SAPRES5.REXX.BLUPARM
MCD0021W Input Arg global replaced by BLUGLOB
MCD0021W Input Arg program replaced by SAPRES5
MCD0022I Input Arg db2 is S
MCD0022I Input Arg infix is TSO
MCD0022I Input Arg inpmemb is $ZMCTSSR
MCD0022I Input Arg inpqurs is SAPRES5.REXX.BLUQRS
MCD0022I Input Arg inpskel is SAPRES5.REXX.BLUSKEL
MCD0022I Input Arg njrepob is 1
MCD0022I Input Arg outjobs is SAPRES5.REXX.BLUJOBS
MCD0022I Input Arg outparm is SAPRES5.REXX.BLUSTIN
MCD0022I Input Arg program is SAPRES5
MCD0022I Input Arg sid is BLU
MCD0022I Input Arg skeldsn is SKGSDSNU
MCD0022I Input Arg skeljob is SKGJJJBC
MCD0022I Input Arg sssid is DB7S
MCD0022I Input Arg ssysid is SC42
MCD0022I Input Arg svcat is SAPBLU
MCD0022I Input Arg tssid is DB7T
MCD0022I Input Arg tsysid is SC49
MCD0022I Input Arg tvcat is MCDBLU
MCD0030A Partitioned Tablespace A223XQZQ.USOBXX manual FUP required
MCD0033I # Tablespace 1105 processed and written to BLUTS001
MCD0034I # Tablespace 3346 are read
MCD0007I Procedure PREPTSOB RC = 0
MCD0013I Elapsed Time 8 in seconds

```

READY
END

Let us look at messages MCD0030A, MCD0033I, and MCD0034I in Example D-15 on page 245:

- ▶ MCD0030A states that a partitioned tablespace is found, the statements produced, and will need a manual follow up treatment.
- ▶ MCD033I states that 1105 tablespaces have output generated.
- ▶ MCD034I states that 3346 tablespaces are read.

The difference $3346 - 1105 = 2241$ tablespaces are not materialized so far, because they are created with the DEFINE NO parameter, and no table that resides in this tablespace contains a row.

Example D-16 shows the JCL generated by the REXX procedure PREPTS0B. Note that the PDS member names are generated by <SID><INFIX>nn, where nn is a number between 1 and NJREPOB.

Example: D-16 JCL generated by PREPTS0B

```
EDIT          SAPRES5.REXX.BLUJOBS(BLUTS001) - 01.00          Columns 00001 00072
Command ==>                                           Scroll ==> CSR
000001 //BLUTS001 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=0M,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC42,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PREPTS0B in Lib SYSEXEC
000013 /* Date 16/12/02 Time 08:23:32
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQRS
000016 //BLURIO01 EXEC DSNUPROC,
000017 // SYSTEM='DB7S',
000018 // UID='BLUTS001',
000019 // COND=(4,LT)
000020 //SYSIN DD DSN=SAPRES5.REXX.BLUSTIN(BLUTS001),
000021 // DISP=SHR
000022 //SYSPRINT DD SYSOUT=*
```

Example D-17 shows sample statements created by the REXX procedure PREPTSOB.

Example: D-17 Sample statements created by PREPTSOB

```
EDIT          SAPRES5.REXX.BLUSTIN(BLUST001) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001  -- Generated by Procedure PREPTSOB  in Lib SYSEXEC
000002  -- Date 16/12/02 Time 08:23:32
000003  -- using GLOBAL member BLUGLOB  in library SAPRES5.REXX.BLUPARM
000004  -- using INPUT member $ZMCTSSR  in library SAPRES5.REXX.BLUQRS
000005  REPAIR OBJECT LOG YES
000006  LOCATE TABLESPACE A000X000.ABAPTREE  PAGE 0
000007  VERIFY OFFSET 12 DATA X'095B'
000008  REPLACE OFFSET 12 DATA X'095B'
000009  VERIFY OFFSET 14 DATA X'0002'
000010  REPLACE OFFSET 14 DATA X'0002'
000011  VERIFY OFFSET 40 DATA 'DB7S'
000012  REPLACE OFFSET 40 DATA 'DB7T'
000013  VERIFY OFFSET 82 DATA 'SAPSTD '
000014  REPLACE OFFSET 82 DATA 'BLUSTD '
000015  VERIFY OFFSET 90 DATA 'SAPBLU '
000016  REPLACE OFFSET 90 DATA 'MCDBLU '
000017  REPAIR OBJECT LOG YES
.....
009390  -- A223XQZQ.USOBXX PARTITIONED MAN FUP REQ
009391  -- LOCATE TABLESPACE A223XQZQ.USOBXX PAGE X'00000000'
009392  -- VERIFY OFFSET 12 DATA X'0E0E'
009393  -- REPLACE OFFSET 12 DATA X'0E0E'
009394  -- VERIFY OFFSET 14 DATA X'0005'
009395  -- REPLACE OFFSET 14 DATA X'0005'
009396  -- VERIFY OFFSET 40 DATA 'DB7S'
009397  -- REPLACE OFFSET 40 DATA 'DB7T'
009398  -- VERIFY OFFSET 82 DATA 'SAPP0D '
009399  -- REPLACE OFFSET 82 DATA 'BLUP0D '
009400  -- VERIFY OFFSET 90 DATA 'SAPBLU '
009401  -- REPLACE OFFSET 90 DATA 'MCDBLU '
009402  -- REPAIR OBJECT LOG YES
009403  -- LOCATE TABLESPACE A223XQZQ.USOBXX
009404  -- VERIFY OFFSET 12 DATA X'0E0E'
.....
```

If we look at rows 009390 to 009404 in Example D-17, we see that the partitioned tablespace A223XQZQ.USOBXX has all the REPAIR statements created. However the PAGE X'.....' of partitions 2 to n cannot be calculated manually with the data provided in the metadata tables. That is why the statements are generated as comments.

PREPINOB

The REXX procedure PREPINOB generates REPAIR statements for indexespaces. It works very much like PREPTSOB, but uses \$ZMCINSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCINSR as the input member).

Deleting created target data sets and renaming source data sets

The VSAM data sets created when running the DDL on the target system must be deleted, and the source data sets must be renamed to match the definitions in the target DB2 catalog. REXX procedures PVSATSDE and PVSAINDE are used to generate the DELETE statements. REXX procedures PVSATSAL and PVSAINAL are used to generate the ALTER statements.

PVSATSDE

The REXX procedure PVSATSDE generates the DELETE statements for tablespaces. By default, it uses \$ZMCTSSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCTSSR as the input member).

Example D-18 shows the invocation of the REXX procedure PVSATSDE.

Example: D-18 Invocation of PVSATSDE

```
%PVSATSDE
```

Example D-19 shows the log of the REXX procedure PVSATSDE.

Example: D-19 Log of PVSATSDE

```
READY
%PVSATSDE
MCD0008I PVSATSDE 1.1.0
MCD0009I in Lib SYSEXEC
MCD0021W Arg inpmemb not specified $ZMCTSSR used
MCD0021W Arg infix not specified TSD used
MCD0021W Arg db2 not specified S used
MCD0021W Arg parmlib not specified SAPRES5.REXX.BLUPARM used
MCD0021W Arg global not specified BLUGLOB used
MCD0021W Arg progname not specified SAPRES5 used
MCD0022I Arg db2 = S
MCD0022I Arg infix = TSD
MCD0022I Arg inpmemb = $ZMCTSSR
```

```

MCD0022I Arg inpqurs = SAPRES5.REXX.BLUQRS
MCD0022I Arg inpskel = SAPRES5.REXX.BLUSKEL
MCD0022I Arg njvsalt = 1
MCD0022I Arg outjobs = SAPRES5.REXX.BLUJOBS
MCD0022I Arg outparm = SAPRES5.REXX.BLUSTIN
MCD0022I Arg progname = SAPRES5
MCD0022I Arg sid = BLU
MCD0022I Arg skelidca = SKGSIDCA
MCD0022I Arg skeljobc = SKGJJBOC
MCD0022I Arg sssid = DB7S
MCD0022I Arg ssysid = SC42
MCD0022I Arg svcat = SAPBLU
MCD0022I Arg tssid = DB7T
MCD0022I Arg tsysid = SC49
MCD0022I Arg tvcat = MCDBLU
MCD0035I # Tablespaces 1105 processed and written to BLUTSD01
MCD0036I # Tablespaces 3346 are read
MCD0007I Procedure PVSATSDE RC = 0
MCD0013I Elapsed Time 4 in seconds
READY
END

```

Example D-20 shows the JCL generated by the REXX procedure PVSATSDE.

Example: D-20 JCL generated by PVSATSDE

```

EDIT          SAPRES5.REXX.BLUJOBS(BLUTSD01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 //BLUTSD01 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=0M,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC42,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PVSATSDE in Lib SYSEXEC
000013 /* Date 16/12/02 Time 09:29:31
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQRS
000016 //BLUVI001 EXEC PGM=IDCAMS
000017 //SYSPRINT DD SYSOUT=*
000018 //SYSIN DD DSN=SAPRES5.REXX.BLUSTIN(BLUTSD01),
000019 // DISP=SHR

```

Example D-21 shows sample statements created by the REXX procedure PVSATSDE. In addition, note that it deals with partitions.

Example: D-21 Sample statements created by PVSATSDE

```
EDIT          SAPRES5.REXX.BLUSTIN(BLUTSD01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 /* Generated by Procedure PVSATSDE in Lib  SYSEXEC          */
000002 /* Date 16/12/02 Time 09:29:15                      */
000003 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM */
000004 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQURS */
000005 DELETE MCDBLU.DSNDBC.A000X02Y.BAPIT00L.I0001.A001
000006 DELETE MCDBLU.DSNDBC.A000X03C.SEUDEPTT.I0001.A001
000007 DELETE MCDBLU.DSNDBC.A000X0DR.TER13.I0001.A001
000008 DELETE MCDBLU.DSNDBC.A000X0TS.DSVASREP.I0001.A001
000009 DELETE MCDBLU.DSNDBC.A000X15F.SXBNFRUL.I0001.A001
000010 DELETE MCDBLU.DSNDBC.A000X1L4.SAASYSTT.I0001.A001
000011 DELETE MCDBLU.DSNDBC.A000X2MI.USR41.I0001.A001 003925
.....
```

PVSAINDE

The REXX procedure PVSAINDE generates the DELETE statements for indexspaces. It works very much like PVSATSDE, but uses \$ZMCINSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCINSR as the input member).

PVSATSAL

The REXX procedure PVSATSAL generates the ALTER statements for tablespaces. By default, it uses \$ZMCTSSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCTSSR as the input member).

Example D-22 shows the invocation of the REXX procedure PVSATSAL.

Example: D-22 Invocation of PVSATSAL

```
%PVSATSAL
```

Example D-23 on page 251 shows the log of the REXX procedure PVSATSAL.

Example: D-23 Log of PVSATSAL

```
READY
%PVSATSAL
MCD0008I PVSATSAL 1.1.0
MCD0009I in Lib SYSEXEC
MCD0021W Arg inpmemb not specified $ZMCTSSR used
MCD0021W Arg infix not specified TSA used
MCD0021W Arg db2 not specified S used
MCD0021W Arg parmlib not specified SAPRES5.REXX.BLUPARM used
MCD0021W Arg global not specified BLUGLOB used
MCD0021W Arg progname not specified SAPRES5 used
MCD0022I Arg db2 = S
MCD0022I Arg infix = TSA
MCD0022I Arg inpmemb = $ZMCTSSR
MCD0022I Arg inpqurs = SAPRES5.REXX.BLUQRS
MCD0022I Arg inpskel = SAPRES5.REXX.BLUSKEL
MCD0022I Arg njvsalt = 1
MCD0022I Arg outjobs = SAPRES5.REXX.BLUJOBS
MCD0022I Arg outparm = SAPRES5.REXX.BLUSTIN
MCD0022I Arg progname = SAPRES5
MCD0022I Arg sid = BLU
MCD0022I Arg skelidca = SKGSIDCA
MCD0022I Arg skeljobc = SKGJJ0BC
MCD0022I Arg sssid = DB7S
MCD0022I Arg ssysid = SC42
MCD0022I Arg svcat = SAPBLU
MCD0022I Arg tssid = DB7T
MCD0022I Arg tsysid = SC49
MCD0022I Arg tvcat = MCDBLU
MCD0035I # Tablespace 1105 processed and written to BLUTSA01
MCD0036I # Tablespace 3346 are read
MCD0007I Procedure PVSATSAL RC = 0
MCD0013I Elapsed Time 7 in seconds
READY
END
```

Example D-24 shows the JCL generated by the REXX procedure PVSATSAL.

Example: D-24 JCL generated by PVSATSAL

```
EDIT          SAPRES5.REXX.BLUJOBS(BLUTSA01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 //BLUTSA01 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
```

```

000007 // REGION=OM,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC42,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PVSATSAL in Lib SYSEXEC
000013 /* Date 16/12/02 Time 09:29:31
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQURS
000016 //BLUVI001 EXEC PGM=IDCAMS
000017 //SYSPRINT DD SYSOUT=*
000018 //SYSIN DD DSN=SAPRES5.REXX.BLUSTIN(BLUTSA01),
000019 // DISP=SHR

```

Example D-25 shows sample statements created by the REXX procedure PVSATSAL. Note that the procedure deals with the cluster *and* data components (see lines 003916 to 003919 in Example D-25). In addition, note that it deals with partitions.

Example: D-25 Sample statements created by PVSATSAL

```

EDIT          SAPRES5.REXX.BLUSTIN(BLUTSA01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 /* Generated by Procedure PVSATSAL in Lib SYSEXEC          */
000002 /* Date 16/12/02 Time 09:29:31                               */
000003 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM */
000004 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQURS */
.....
003915
003916 ALTER SAPBLU.DSNDBC.A223XQZQ.TABL.VALU.ZM_SS.I0001.A001 -
003917         NEWNAME (MCDBLU.DSNDBC.A223XQZQ.TABL.VALU.ZM_TS.I0001.A001)
003918 ALTER SAPBLU.DSNDBD.A223XQZQ.TABL.VALU.ZM_SS.I0001.A001 -
003919         NEWNAME (MCDBLU.DSNDBD.A223XQZQ.TABL.VALU.ZM_TS.I0001.A001)
003920
003921 ALTER SAPBLU.DSNDBC.A223XQZQ.TABL.VALU.ZM_SS.I0001.A002 -
003922         NEWNAME (MCDBLU.DSNDBC.A223XQZQ.TABL.VALU.ZM_TS.I0001.A002)
003923 ALTER SAPBLU.DSNDBD.A223XQZQ.TABL.VALU.ZM_SS.I0001.A002 -
003924         NEWNAME (MCDBLU.DSNDBD.A223XQZQ.TABL.VALU.ZM_TS.I0001.A002)
003925
.....

```

PVSAINAL

The REXX procedure PVSAINAL generates the ALTER statements for indexspaces. It works very much like PVSATSAL, but uses \$ZMCINSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCINSR as the input member).

Executing REPAIR LEVELID on tablespaces and indexes

Once that the source VSAM data sets have been renamed and repaired for use in the target system, the target DB2 subsystem will recognize the VSAM data sets. However, the LEVELID may not be compatible. REXX procedures PREPTSLV and PREPINLV generate the REPAIR LEVELID statements to achieve this.

PREPTSLV

The REXX procedure PREPTSLV generates REPAIR LEVELID statements for tablespaces. By default, it uses \$ZMCTSSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCTSSR as the input member).

Example D-26 shows the invocation of the REXX procedure PREPTSLV.

Example: D-26 Invocation of PREPTSLV

```
%PREPTSLV
```

Example D-27 shows the log of the REXX procedure PREPTSLV.

Example: D-27 Log of PREPTSLV

```
READY
%PREPTSLV
MCD0008I PREPTSLV 1.1.0
MCD0009I Procedure PREPTSLV in Lib SYSEXEC
MCD0021W Input Arg inpmemb replaced by $ZMCTSSR
MCD0021W Input Arg infix replaced by TSL
MCD0021W Input Arg db2 replaced by T
MCD0021W Input Arg parmlib replaced by SAPRES5.REXX.BLUPARM
MCD0021W Input Arg global replaced by BLUGLOB
MCD0021W Input Arg progname replaced by SAPRES5
MCD0022I Input Arg db2 is T
MCD0022I Input Arg infix is TSL
MCD0022I Input Arg inpmemb is $ZMCTSSR
MCD0022I Input Arg inpqurs is SAPRES5.REXX.BLUQRS
MCD0022I Input Arg inpskel is SAPRES5.REXX.BLUSKEL
MCD0022I Input Arg njreplv is 1
MCD0022I Input Arg outjobs is SAPRES5.REXX.BLUJOBS
MCD0022I Input Arg outparm is SAPRES5.REXX.BLUSTIN
MCD0022I Input Arg progname is SAPRES5
MCD0022I Input Arg sid is BLU
MCD0022I Input Arg skeldsnu is SKGSDSNU
MCD0022I Input Arg skeljobc is SKGJJ0BC
MCD0022I Input Arg sssid is DB7S
```

```

MCD0022I Input Arg  ssysid   is   SC42
MCD0022I Input Arg  svcat    is   SAPBLU
MCD0022I Input Arg  tssid    is   DB7T
MCD0022I Input Arg  tsysid   is   SC49
MCD0022I Input Arg  tvcat    is   MCDBLU
MCD0031I # Tablespaces 1105 processed and written to BLUTSL01
MCD0032I # Tablespaces 3346 are read
MCD0007I Procedure PREPTSLV RC = 0
MCD0013I Elapsed Time 6 in seconds
READY
END

```

Example D-28 shows the JCL generated by the REXX procedure PREPTSLV.

Example: D-28 JCL generated by PREPTSLV

```

EDIT          SAPRES5.REXX.BLUJ0BS(BLUTSL01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 //BLUTSL01 JOB (999,P0K),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=0M,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC49,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PREPTSLV in Lib SYSEXEC
000013 /* Date 16/12/02 Time 09:03:09
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCTSSR in library SAPRES5.REXX.BLUQRS
000016 //BLUTSL01 EXEC DSNUPROC,
000017 // SYSTEM='DB7T',
000018 // UID='BLUTSL01',
000019 // COND=(4,LT)
000020 //SYSIN DD DSN=SAPRES5.REXX.BLUSTIN(BLUTSL01),
000021 // DISP=SHR
000022 //SYSPRINT DD SYSOUT=*

```

Example D-29 on page 255 shows sample statements created by the REXX procedure PREPTSLV. Note that the procedure deals with partitioned tablespaces.

Example: D-29 Sample statements created by PREPTSLV

```
EDIT          SAPRES5.REXX.BLUSTIN(BLUTSL01) - 01.00          Columns 00001 00072
Command ==>                               Scroll ==> CSR
000001  -- Generated by Procedure PREPTSLV  in Lib SYSEXEC      --
000002  -- Date 16/12/02 Time 09:03:09      --
000003  -- using GLOBAL member BLUGLOB  in library SAPRES5.REXX.BLUPARM  --
000004  -- using INPUT member $ZMCTSSR  in library SAPRES5.REXX.BLUQRS    --
000005  REPAIR LEVELID TABLESPACE A000X8X4.ABAPTREE
000006  REPAIR LEVELID TABLESPACE A000X0XG.STXRDIR
000007  REPAIR LEVELID TABLESPACE A000X0ER.SHIEXPRO
000008  REPAIR LEVELID TABLESPACE A000X0F8.OJDEFS
000009  REPAIR LEVELID TABLESPACE A000X1AP.SDOKMESA
000010  REPAIR LEVELID TABLESPACE A000X1HD.MSSSPVER
000011  REPAIR LEVELID TABLESPACE A000X1VE.DB2IFIT
.....
000787  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 1
000788  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 2
000789  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 3
000790  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 4
000791  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 5
000792  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 6
000793  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 7
000794  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 8
000795  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 9
000796  REPAIR LEVELID TABLESPACE A223XQZQ.USOBXX PART 10
000797  REPAIR LEVELID TABLESPACE A230X18G.LTDX
.....
```

PREPINLV

The REXX procedure PREPINLV generates the REPAIR LEVELID statements for indexes. It works very much like PREPINDEX, but uses \$ZMCINSR as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$ZMCINSR as the input member).

Generating DDL for views

The same logic is also used by the REXX procedure that generates DDL for views. The procedure works on information extracted from the source DB2 catalog tables by the REXX procedure PQUERYTB and a specific SQL query statement (for more information about PQUERYTB, refer to “Extracting information from the metadata tables” on page 236).

\$IBMVIST

The SQL query \$IBMVIST, as shown in Example D-30, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the views. In that case, PQUERYTB must be run against the *source* DB2 subsystem.

Example: D-30 SQL query to retrieve information about views

```
-- Query against SYSIBM.SYSVIEWS
-- used in SG24-6914
select
  VI.NAME           as VI_NAME,
  VI.CREATOR       as VI_CREATOR,
  VI.SEQNO         as VI_SEQNO,
  VI.CHECK         as VI_CHECK,
  VI.PATHSCHEMAS  as VI_PATHSCHEMAS,
  VI.TEXT          as VI_TEXT
from
  SYSIBM.SYSVIEWS as VI
where
  VI.CREATOR = 'SAPR3'
order by
  VI_NAME,
  VI_SEQNO ;
```

PCREATVI

The REXX procedure PCREATVI generates DDL for views. By default, it uses \$IBMVIST as the input member (member generated by the REXX procedure PQUERYTB, using the SQL query \$IBMVIST as the input member).

Example D-31 shows the invocation of the REXX procedure PCREATVI.

Example: D-31 Invocation of PCREATVI

```
%PCREATEVI
```

Example D-33 on page 257 shows the log of the REXX procedure PCREATVI.

Example: D-32 Log of PCREATVI

```
READY
%PCREATVI
MCD0008I PCREATVI 1.1.0
MCD0009I Procedure PCREATVI in Lib SYSEXEC
MCD0021W Input Arg inpmemb replaced by $IBMVIST
MCD0021W Input Arg local replaced by BLUTVI
```

```

MCD0021W Input Arg infix replaced by VDD
MCD0021W Input Arg db2 replaced by T
MCD0021W Input Arg parmlib replaced by SAPRES5.REXX.BLUPARM
MCD0021W Input Arg global replaced by BLUGLOB
MCD0021W Input Arg program replaced by SAPRES5
MCD0022I Input Arg db2 is T
MCD0022I Input Arg infix is VDD
MCD0022I Input Arg inpmemb is $IBMVIST
MCD0022I Input Arg inpqurs is SAPRES5.REXX.BLUQURS
MCD0022I Input Arg inpskel is SAPRES5.REXX.BLUSKEL
MCD0022I Input Arg njdb2dd is 1
MCD0022I Input Arg outjobs is SAPRES5.REXX.BLUJOBS
MCD0022I Input Arg outparm is SAPRES5.REXX.BLUSTIN
MCD0022I Input Arg program is SAPRES5
MCD0022I Input Arg sid is BLU
MCD0022I Input Arg skelddb is SKGSDBDD
MCD0022I Input Arg skeljob is SKGJJ0BC
MCD0022I Input Arg dsnload is DB7T7.SDSNLOAD
MCD0022I Input Arg dsnext is DB7T7.SDSNEXT
MCD0022I Input Arg dsrunl is DB7TU.RUNLIB.LOAD
MCD0022I Input Arg tschema is SAPBLU
MCD0022I Input Arg sssid is DB7S
MCD0022I Input Arg ssysid is SC42
MCD0022I Input Arg svcat is SAPBLU
MCD0022I Input Arg tssid is DB7T
MCD0022I Input Arg tsysid is SC49
MCD0022I Input Arg tvcat is MCDBLU
MCD0037I £ views 623 processed, written to BLUVIE01
MCD0038I £ views 623 are read from $IBMVIST
MCD0007I Procedure PCREATVI RC = 0
MCD0013I Elapsed Time 4 in seconds
READY
END

```

Example D-33 shows the JCL generated by the REXX procedure PCREATVI.

Example: D-33 JCL generated by PCREATVI

```

EDIT          SAPRES5.REXX.BLUJOBS(BLUVDD01) - 01.00          Columns 00001 00072
Command ===>                                          Scroll ===> CSR
000001 //BLUVIE01 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=0M,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)

```

```

000010 /*JOBPARM SYSAFF=SC49,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PCREATVI in Lib ?
000013 /* Date 10/01/03 Time 12:05:35
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $IBMVIS in library SAPRES5.REXX.BLUQURS
000016 //BLUVIE01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
000017 //STEPLIB DD DSN=DB7T7.SDSNLOAD,
000018 // DISP=SHR
000019 //          DD DSN=DB7T7.SDSNEXIT,
000020 // DISP=SHR
000021 //SYSTSIN DD *
000022   DSN SYSTEM(DB7T)
000023   RUN PROGRAM(DSNTEP2) PLAN(DSNTEP71) -
000024       LIB('DB7TU.RUNLIB.LOAD') PARMS('/ALIGN(MID)')
000025 //SYSTSPRT DD SYSOUT=*
000026 //SYSPRINT DD SYSOUT=*
000027 //SYSUDUMP DD SYSOUT=*
000028 //SYSIN   DD DSN=SAPRES5.REXX.BLUSTIN(BLUVDD01),
000029 // DISP=SHR

```

Example D-34 shows sample statements created by the REXX procedure PCREATVI.

Example: D-34 Sample statements created by PCREATVI

```

EDIT      SAPRES5.REXX.BLUSTIN(BLUVDD01) - 01.00          Columns 00001 0007
Command ===>                                         Scroll ===> CSR
000001 -- Generated by Procedure PCREATVI in Lib SYSEXEC --
000002 -- Date 10/01/03 Time 12:05:35 --
000003 -- using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM --
000004 -- using INPUT member $IBMVIS in library SAPRES5.REXX.BLUQURS --
000005 set current sqlid='SAPBLU' ;
000006 commit;
000007
000008 CREATE
000009 VIEW
000010 "APPL_DEVC"
000011 (
000012 "DEVCLASS",
000013 "OBJ_NAME"
000014 )
000015 AS
000016 SELECT
000017 T0001."DEVCLASS",
000018 T0001."DEVCLASS"
000019 FROM
000020 "TDEV"
000021 T0001

```

```
000022 ;
.....
```

Generating DDL for altering object sizes

The same logic is also used by the REXX procedures that generate DDL for altering object sizes after the migration is completed. The procedures work on information extracted from the source DB2 catalog tables and the metadata tables by the REXX procedure PQUERYTB and specific SQL queries (for more information about PQUERYTB, refer to “Extracting information from the metadata tables” on page 236).

\$IBMTSSR

The SQL query \$IBMTSSR, as shown in Example D-35, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the source tablespaces. In that case, PQUERYTB must be run against the *source* DB2 subsystem.

Example: D-35 SQL query to retrieve information about source tablespaces

```
-- GET DATA
--
select
  TS.DBNAME      as TS_DBNAME ,
  TS.NAME        as TS_NAME ,
  TS.CREATOR     as TS_CREATOR ,
  TS.DBID        as TS_DBID ,
  TS.PSID        as TS_PSID ,
  TS.OBID        as TS_OBID ,
  TS.TYPE        as TS_TYPE ,
  TS.PARTITIONS  as TS_PARTITIONS ,
  TS.CLOSERULE   as TS_CLOSERULE ,
  TS.ERASERULE   as TS_ERASERULE ,
  TS.BPOOL       as TS_BPOOL ,
  TS.LOCKRULE    as TS_LOCKRULE ,
  TS.LOCKMAX     as TS_LOCKMAX ,
  TS.LOCKPART    as TS_LOCKPART ,
  TS.MAXROWS     as TS_MAXROWS ,
  TS.DSSIZE      as TS_DSSIZE ,
  TS.LOCKRULE    as TS_LOCKRULE ,
  TS.LOG         as TS_LOG ,
  TS.ENCODING_SCHEME as TS_ENCODING ,
  TS.SEGSIZE     as TS_SEGSIZE ,
  TP.SPACE       as TP_SPACE ,
  TP.COMPRESS    as TP_COMPRESS,
```

```

TP.PARTITION      as TP_PARTITION ,
TP.IXNAME         as TP_IXNAME    ,
TP.GBPCACHE      as TP_GBPCACHE,
TP.PQTY          as TP_PQTY      ,
TP.SQTY          as TP_SQTY      ,
TP.SECQTYI       as TP_SECQTYI   ,
TP.STORTYPE      as TP_STORTYPE   ,
TP.FREEPAGE      as TP_FREEPAGE  ,
TP.PCTFREE       as TP_PCTFREE   ,
TP.TRACKMOD      as TP_TRACKMOD  ,
TP.LIMITKEY      as TP_LIMITKEY  ,
from
  SYSIBM.SYSTABLESPACE as TS,
  SYSIBM.SYSTABLEPART as TP
where
  TS.CREATOR = 'SAPR3'
and
  TP.DBNAME  = TS.DBNAME
and
  TP.TSNAME = TS.NAME
order by
  TS_DBNAME,
  TS_PSID,
  TP_PARTITION

```

\$IBMINSR

The SQL query \$IBMINSR, as shown in Example D-36, can be used as the input member of the REXX procedure PQUERYTB to retrieve information related to the source indexes. In that case, PQUERYTB must be run against the *source* DB2 subsystem.

Example: D-36 SQL query to retrieve information about source indexes

```

-- Query against SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXPART
-- used in SG24-6914
select
  IX.DBNAME      as IX_DBNAME      ,
  IX.INDEXSPACE  as IX_INDEXSPACE  ,
  IX.NAME        as IX_NAME        ,
  IX.TBNAME      as IX_TBNAME      ,
  IP.PARTITION   as IP_PARTITION   ,
  IX.CREATOR     as IX_CREATOR     ,
  IX.UNIQUERULE  as IX_UNIQUERULE  ,
  IX.COLCOUNT   as IX_COLCOUNT   ,
  IX.CLUSTERING  as IX_CLUSTERING  ,
  IX.CLUSTERED   as IX_CLUSTERED   ,
  IX.BPOOL       as IX_BPOOL       ,

```

```

IX.ERASERULE      as      IX_ERASERULE      ,
IX.CLOSERULE      as      IX_CLOSERULE      ,
IX.PIECESIZE      as      IX_PIECESIZE      ,
IX.COPY           as      IX_COPY           ,
IP.PQTY           as      IP_PQTY           ,
IP.SQTY           as      IP_SQTY           ,
IP.SECQTYI        as      IP_SECQTYI       ,
IP.STORNAME       as      IP_STORNAME      ,
IP.GBPCACHE       as      IP_GBPCACHE      ,
IP.IPREFIX        as      IP_IPREFIX       ,
from
  SYSIBM.SYSINDEXES as IX ,
  SYSIBM.SYSINDEXPART as IP
where
  IX.CREATOR = 'SAPR3'
and
  IP.IXCREATOR = 'SAPR3'
and
  IP.STORNAME LIKE 'SAP%'
and
  IX.NAME = IP.IXNAME
order by
  IX_DBNAME      ,
  IX_INDEXSPACE ,
  IP_PARTITION   ,
  IX_TBNAME

```

PTEPTSAL

The REXX procedure PTEPTSAL generates DDL for altering tablespace sizes. By default, it uses \$ZMCTSSR and \$IBMTSSR as the input members (members generated by the REXX procedure PQUERYTB, using SQL queries \$ZMCTSSR and \$IBMTSSR as input members).

However, because the metadata tables are defined as ASCII tables, whereas the DB2 catalog tables are EBCDIC tables, the order of rows in \$ZMCTSSR and \$IBMTSSR may be different, and you may receive the following error message:

```
MCD0050E Input files $ZMCINSR and $IBMINSR are out of sync
```

Therefore, we use the member \$ZMCTSSS as input member, instead of \$ZMCTSSR. \$ZMCTSSS is created as follows:

1. Copy the header part of \$ZMCTSSR (lines 1 to 42) in a new member called \$ZMCTSHD.
2. Copy the list part of \$ZMCTSSR (lines 43 and next) in a new member called \$ZMCTSSS.

- Use the following ISPF command to sort \$ZMCTSSS according to the EBCDIC order of the field ZM_SRCDBNAME:

```
SORT 21 29
```

- Append \$ZMCTSHD at the top of \$ZMCTSSS.

Example D-37 shows the invocation of the REXX procedure PTEPTSAL.

Example: D-37 Invocation of PTEPTSAL

```
%PTEPTSAL INPMEMB=$ZMCTSSS
```

Example D-38 shows the log of the REXX procedure PTEPTSAL.

Example: D-38 Log of PTEPTSAL

```
READY
%PTEPTSAL INPMEMB=$ZMCTSSS
MCD0008I PTEPTSAL 1.1.0
MCD0009I in Lib SYSEXEC
MCD0021W Arg inpmem2 not specified $IBMTSSR used
MCD0021W Arg infix not specified TSQ used
MCD0021W Arg db2 not specified T used
MCD0021W Arg parmlib not specified SAPRES5.REXX.BLUPARM used
MCD0021W Arg local not specified BLUTDD used
MCD0021W Arg global not specified BLUGLOB used
MCD0021W Arg progname not specified SAPRES5 used
MCD0021W Arg njdb2dd not specified 1 used
MCD0022I Arg db2 = T
MCD0022I Arg dsnexit = DB7S7.SDSNEXIT
MCD0022I Arg dsnload = DB7S7.SDSNLOAD
MCD0022I Arg dsrunl = DB7SU.RUNLIB.LOAD
MCD0022I Arg infix = TSQ
MCD0022I Arg inpmemb = $ZMCTSSS
MCD0022I Arg inpmem2 = $IBMTSSR
MCD0022I Arg inpqurs = SAPRES5.REXX.BLUQRS
MCD0022I Arg inpskel = SAPRES5.REXX.BLUSKEL
MCD0022I Arg njdb2dd = 1
MCD0022I Arg outjobs = SAPRES5.REXX.BLUJOBS
MCD0022I Arg outparm = SAPRES5.REXX.BLUSTIN
MCD0022I Arg progname = SAPRES5
MCD0022I Arg sid = BLU
MCD0022I Arg skeldbdd = SKGSDBDD
MCD0022I Arg skeljobc = SKGJJJBC
MCD0022I Arg sssid = DB7X
MCD0022I Arg ssysid = SC42
MCD0022I Arg svcat = SAPBLU
MCD0022I Arg tschema = SAPBLU
MCD0022I Arg tssid = DB7T
```



```

MCD0022I Arg tsysid   = SC49
MCD0022I Arg tvcat    = MCDBLU
MCD0035I £ Tablespace 3331 processed and written to BLUTSQ01
MCD0036I £ Tablespace 3331 are read from $ZMCTSSS
MCD0036I £ Tablespace 3331 are read from $IBMTSSR
MCD0007I Procedure PTEPTSAL RC = 0
MCD0013I Elapsed Time 17 in seconds
READY
END

```

Example D-39 shows the JCL generated by the REXX procedure PTEPTSAL.

Example: D-39 JCL generated by PTEPTSAL

```

EDIT          SAPRES1.REXX.BLUJOBS(BLUTSQ01) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
000001 //BLUTSQ01 JOB (999,POK),
000002 // 'SAPRES5',
000003 // CLASS=A,
000004 // MSGCLASS=T,
000005 // NOTIFY=&SYSUID,
000006 // TIME=1440,
000007 // REGION=OM,
000008 // RESTART=HERE
000009 // JCLLIB ORDER=(DB7SU.PROCLIB)
000010 /*JOBPARM SYSAFF=SC49,L=9999
000011 //HERE EXEC PGM=IEFBR14
000012 /* Generated by Procedure PTEPTSAL in Lib SYSEXEC
000013 /* Date 14/01/03 Time 06:13:58
000014 /* using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
000015 /* using INPUT member $ZMCTSSS in library SAPRES5.REXX.BLUQURS
000016 //BLUTSQ01 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
000017 //STEPLIB DD DSN=DB7S7.SDSNLOAD,
000018 // DISP=SHR
000019 //          DD DSN=DB7S7.SDSNEXIT,
000020 // DISP=SHR
000021 //SYSTSIN DD *
000022   DSN SYSTEM(DB7T)
000023   RUN PROGRAM(DSNTEP2) PLAN(DSNTEP71) -
000024     LIB('DB7SU.RUNLIB.LOAD') PARMS('/ALIGN(MID)')
000025 //SYSTSPRT DD SYSOUT=*
000026 //SYSPRINT DD SYSOUT=*
000027 //SYSUDUMP DD SYSOUT=*
000028 //SYSIN   DD DSN=SAPRES5.REXX.BLUSTIN(BLUTSQ01),
000029 // DISP=SHR

```

Example D-40 on page 264 shows sample statements created by the REXX procedure PTEPTSAL.

Example: D-40 Sample statements created by PTEPTSAL

```
-- Generated by Procedure PTEPTSAL in Lib SYSEXEC
-- Date 14/01/03 Time 06:13:58
-- using GLOBAL member BLUGLOB in library SAPRES5.REXX.BLUPARM
-- using INPUT member $ZMCTSSS in library SAPRES5.REXX.BLUQRS
set current sqlid='SAPBLU';
commit ;
  alter tablespace A000XAAF.BDNODE
    priquty 144 secquty 100
  ;
.....
```

PTEPINAL

The REXX procedure PTEPINAL generates DDL for altering index sizes. It works very much like PTEPINTS, but uses by default \$ZMCINSR and \$IBMINSR as input members (members generated by the REXX procedure PQUERYTB, using SQL queries \$ZMCINSR and \$IBMINSR as input members).

However, we use the member \$ZMCINSS as input member, instead of \$ZMCINSR:

1. Copy the header part of \$ZMCINSR (lines 1 to 43) in a new member called \$ZMCINHD.
2. Copy the list part of \$ZMCINSR (lines 44 and next) in a new member called \$ZMCINSS.
3. Use the following ISPF command to sort \$ZMCINSS according to the EBCDIC order of the fields ZM_SRCDBNAME and ZM_SRCINDEXSPACE:
SORT 4 22
4. Append \$ZMCINHD at the top of \$ZMCINSS.

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246914>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246914.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
MCOD.CAR	SAP transport to create metadata tables

AppendixC.zip	REXX procedures and JCL referred to in Appendix C, “Merge in place: Defining source objects in target system” on page 197
AppendixD.zip	REXX procedures, parameter files, and JCL referred to in Appendix D, “Merge in place: Migrating the data” on page 233

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Abbreviations and acronyms

ACS	access control system	ICF	integrated catalog facility
APAR	authorized program analysis report	ICLI	Integrated Call Level Interface
ASCII	American Standard Code for Information Interchange	ID	identifier
ATS	Advanced Technical Support	IMG	Implementation Guide
BSDS	bootstrap data set	IMIG	incremental migration
CCMS	Computer Center Management System	ISICC	IBM SAP International Competence Center
CPU	central processing unit	ISOBID	indexspace object identifier
DASD	direct access storage device	IT	information technology
DB	database	ITSO	International Technical Support Organization
DBA	database administrator	IX	index
DBMS	database management system	JCL	job control language
DBRM	database request module	KB	kilobyte (1,024 bytes)
DDF	distributed data facility	LOB	large object
DDIC	data dictionary	LPAR	logical partition
DDL	data definition language	LRSN	log record sequence number
DLL	dynamic link library	MB	megabyte (1,048,576 bytes)
DML	data manipulation language	MCOD	Multiple Components in One Database
DRDA	Distributed Relational Database Architecture	OBID	object identifier
ESS	Enterprise Storage Server™	OLAP	online analytical processing
FTP	File Transfer Protocol	OLTP	online transaction processing
GB	gigabyte (1,073,741,824 bytes)	OS	operating system
GUI	graphical user interface	PDS	partitioned data set
HFS	Hierarchical File System	PPT	prior point in time
HLQ	high-level qualifier	PSID	pageset identifier
HSC	homogeneous system copy	PSP	preventive service planning
I/O	input/output	PTF	program temporary fix
IBM	International Business Machines Corporation	QMF	query management facility
		RACF	Resource Access Control Facility
		RBA	relative byte address

RVA	RAMAC® Virtual Array
SAP	Systems, Applications, Products in Data Processing
SAP AG	SAP Aktiengesell
SAP APO	SAP Advanced Planner And Optimizer
SAP BW	SAP Business Information Warehouse
SAP CRM	SAP Customer Relationship Management
SAP SEM	SAP Strategic Enterprise Management
SAP WAS	SAP Web Application Server
SG	storage group
SMS	Storage Management Subsystem
SQL	Structured Query Language
TB	table
TMS	transport management system
TS	tablespace
UDB	Universal Database
UR	unit of recovery
USS	UNIX system services
VSAM	Virtual Storage Access Method
WLM	Workload Manager



Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 272.

- ▶ *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287
- ▶ *SAP R/3 on DB2 for OS/390: Database Availability Considerations*, SG24-5690
- ▶ *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836

Other References

These publications are also relevant as further information sources:

- ▶ *DB2 UDB for OS/390 and z/OS Diagnosis Guide and Reference Version 7*, LY37-3740
- ▶ *DB2 Universal Database for OS/390 Data Sharing: Planning and Administration Version 6*, SC26-9007

- ▶ *DB2 Universal Database for OS/390 and z/OS Data Sharing: Planning and Administration Version 7*, SC26-9935
- ▶ *DB2 Universal Database for OS/390 and z/OS V7: Installation Guide*, GC26-9936
- ▶ *DB2 Universal Database for OS/390 and z/OS: Utility Guide and Reference Version 7*, SC26-9945
- ▶ *SAP R/3 on DB2 UDB for OS/390 and z/OS: Planning Guide, 2nd Edition, SAP R/3 Release 4.6D*, SC33-7966
- ▶ *SAP on DB2 UDB for OS/390 and z/OS: Planning Guide, SAP Web Application Server 6.20*, SC33-7959
- ▶ *Storage Management for SAP and DB2 UDB: Split Mirror Backup/Recovery with IBM's Enterprise Storage Server (ESS)*, white paper by Sanjoy Das, Siegfried Schmidt, Jens Claussen, and BalaSanni Godavari, available at:
<http://www.storage.ibm.com/hardsoft/products/sap/smdb2.pdf>

The following program directories are referenced in this redbook:

- ▶ *Program Directory for DB2 Management Tools Package (JDB661D)*, GI10-8193
- ▶ *Program Directory for IBM DB2 UDB Server for OS/390 and z/OS: DB2 Management Clients Package (JDB771D)*, GI10-8218
- ▶ *Program Directory for DB2 for OS/390 and z/OS: DB2 Administration Server for z/OS (HDAS810)*, GI10-8472

The following PSP buckets are referenced in this redbook:

- ▶ *390 Enablement V6 (Upgrade DB2610, subset JDB661D)*
- ▶ *390 Enablement V7 (Upgrade DB2710, subset JDB771D)*
- ▶ *DAS (for DB2/390 V7) (Upgrade DB2710, subset HDAS810)*

You must be registered as an SAP Service Marketplace user to access the following resources. The registration requires an SAP installation or customer number. To register, go to:

<http://service.sap.com>

- ▶ *Installation Guide – SAP Web Application Server 6.20 on UNIX: IBM DB2 Universal Database for OS/390 and z/OS*, SAP document available at:
<http://service.sap.com/instguides>
- ▶ *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide, SAP Web Application Server, Release 6.20*, SAP document available at:
<http://service.sap.com/instguides>

- ▶ *BC SAP Database Administration Guides: DB2 for OS/390*
 - SAP Release 3.1I: material number 51002788
 - SAP Release 4.0B: material number 51002661
 - SAP Release 4.5B: material number 51006377
 - SAP Release 4.6C: material number 51009638
- ▶ *SAP R/3 Homogeneous System Copy, Release 4.6C SR2*, material number 51013678
- ▶ *SAP Web Application Server 6.20: Homogeneous and Heterogeneous System Copy*, SAP document available at:

<http://service.sap.com/instguides>

SAP Notes

You must be registered as an SAP Service Marketplace user to access the following resources. The registration requires an SAP installation or customer number. To register, go to:

<http://service.sap.com>

The following SAP Notes are referenced in this redbook:

- ▶ *DB2/390: APAR List*, SAP Note 081737
- ▶ *DB2/390: Backup and Recovery Options*, SAP Note 083000
- ▶ *DB2/390: Installing saposcol manually*, SAP Note 103135
- ▶ *DB2/390: PTF check tool*, SAP Note 183311
- ▶ *DB2/390: DDIC corrections (Releases 4.6A, 4.6B, 4.6C, 4.6D)*, SAP Note 184399
- ▶ *DB2/390: Transports for 4.6C*, SAP Note 217093
- ▶ *DB2/390: Newest version of the CCMS 4.6C*, SAP Note 217468
- ▶ *DB2/390: Transports for Release 4.6D*, SAP Note 324739
- ▶ *DB2/390: RSDB2MAS – new version*, SAP Note 330289
- ▶ *DB2/390: Latest version of CCMS 4.6D*, SAP Note 337776
- ▶ *DB2/390: Incremental Migration to DB2/390*, SAP Note 353558
- ▶ *DB2/390: MCOD installation*, SAP Note 399419
- ▶ *DB2/390: New failover support from Release 6.10*, SAP Note 402078
- ▶ *DB2/390: DDIC corrections (6.10, 6.20)*, SAP Note 407663
- ▶ *DB2/390: DB Performance Monitor/IFI Data Collector*, SAP Note 426863

- ▶ *DB2/390: CCMS corrections (6.10, 6.20)*, SAP Note 427748
- ▶ *DB2/390: MCOD installation tools*, SAP Note 580665

Referenced Web sites

These Web sites are also relevant as further information sources.

You must be registered as an SAP Service Marketplace user to access the following resources. The registration requires an SAP installation or customer number. To register, go to:

<http://service.sap.com>

- ▶ SAP Service Marketplace quick link MCOD:
<http://service.sap.com/mcod>
- ▶ SAP Service Marketplace quick link INSTGUIDES:
<http://service.sap.com/instguides>
- ▶ SAP Service Marketplace quick link PATCHES:
<http://service.sap.com/patches>
- ▶ SAP Service Marketplace quick link QUICKSIZER:
<http://service.sap.com/quicksizer>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Symbols

/etc/services 67

Numerics

390 Enablement (JDB661D or JDB771D) 76

A

ACS routines 6, 59, 85
ADB2GEN 56, 198
alerts 139
application owners 117, 119
application-level recovery 118

B

backup 46, 117
 monitor 136
Basis Support Package 128
BIND 36, 67, 129
bufferpools 7, 11, 18

C

cause of error 119, 122
CCMS 128
 monitor set 137
 transports 37, 128
central DBA planning calendar 131, 134
checklist 19, 23, 47
cloning
 process 71
 wizard 75, 78
cold start 22, 64, 71, 105, 111
conditional restart 70
connect.ini 67
consistency 18, 117
consolidation 19, 118
Control Center 72
creator 5, 59

D

Database Administration Server (DAS) 76
database layout 145

database level recovery 118
database performance analysis 138
database user 4
DB2

 Administration Tool for z/OS 44
 catalog and directory 8
 data sharing 10, 12, 17, 43, 73
 maintenance 37

DBA

 planning calendar 135
DBEXPORT.R3S 35
DBID 59, 61
DBSL 128
DD_GET_STORAGE_CLASS 151
DDIC corrections 37, 128
DDL generation 55
DEFAULT.PFL 36, 66, 107
DEFINE NO 46, 52, 63
DFDSS copy/rename 71
disp+work 128
domain controller 66
DRDA 11
DSN1LOGP 102
DSNACCCx 91
DSNACCMO 90
DSNDB07 113
DSNJU003 64, 112
DSNJU004 22, 102, 111
DSNTEP2 58
DSNUTILS 90

E

EMC Timefinder 121
environment variables
 DBS_DB2_SCHEMA 6, 36, 66
 DBS_DB2_SSID 66
 R3_DB2_SSID 66
 SAPDBHOST 66
ESS FlashCopy 70, 73, 117, 121
export 34

F

filler tablespaces 57, 230

G

GRANT 36, 67

H

hard copy 47

homogeneous system copy (HSC) 70

I

ICLI 7, 11, 36, 67, 107

IDCAMS

ALTER 22, 45–46, 63

DELETE 63

Incremental Migration (IMIG) 38

ISOBID 59

ITSO scenarios

add one SAP component 30

clone one SAP component 73

merge in place 49

merge using SAP tools 33

recovery of one component 121

system configuration 24

J

JES2 internal reader 56

L

license key 19–20, 23–24

LOB 43

long-running URs 139

LRSN 111, 120

M

MCOD

availability 4

benefits 2

cloning 72, 96

definition 2

drawbacks 3

implementation 9

motivation 2

sizing 16

MCODCE_UNIX_db2.BASE.R3S 36

MCODCE_UNIX_db2.base.R3S 107

MCODMIG_UNIX_db2.BASE.R3S 36

merge in place 43

limitations 43

reasons 43

metadata 46, 53, 58–60, 151, 153

mirroring 47

mixed layouts 150

MMCGRTS 101

N

naming conflicts resolution 150

naming convention 7

databases 7, 45, 146, 148

packages 7

plans 7

schema 5, 45

storage groups 6, 45, 146, 148

tablespaces 7, 148

VCAT 6, 45

NPGTHRSH 65

O

OBID 55, 57, 230

offline copy 71

OLAP 10

OLTP 10

online copy 70

P

partitioned tablespaces 43, 62

patches 36

points of consistency 116, 120

PQ68650 96

PQ68659 76

primary VCAT 72, 84

profile parameters

db2/db2/hosttcp 66

db2/db2/schema 6, 36, 66

db2/db2/ssid 66

PSID 59, 61

Q

QUIESCE 120, 122

R

R3SETUP 8, 29

R3trans 128

RBA 22, 44, 111, 120

recovery 18

scenarios 118

- steps 123
- Redbooks Web site 272
 - Contact us xvi
- REORG 52
- REPAIR 22, 46
- REPAIR INDEX 45, 61
- REPAIR LEVELID 45, 63
- REPAIR TABLESPACE 45, 61
- rfcoscol 128
- RSDB2MAS 52, 68
- RUNSTATS 23, 60, 65
- RVA Snapshot 121

S

- SAP components 2, 29
 - delete 8
 - install 30
 - maximum number 17
 - upgrade 8
- SAP Kernel patches 36–37, 128
- SAP Notes
 - 081737 37
 - 083000 116
 - 183311 37
 - 184399 37
 - 217468 37
 - 330289 52
 - 337776 37
 - 353558 39
 - 399419 30
 - 402078 67
 - 407663 37
 - 427748 37
 - 580665 29–30, 35
- SAP Service Marketplace quick links
 - MCOD 4, 29
 - PATCHES 35, 37
 - quicksizer 16
- SAP transactions
 - DB02 133
 - DB12 136
 - DB13 135
 - DB13C 131, 134
 - DB2 129
 - DB2J 67
 - RZ20 137
 - SM59 131
 - STMS 66

- SAPinst 8, 30–31
- saposcol 128
- schema 5–6, 20–21, 30–31, 58, 60, 67
- security 9
- SET LOG SUSPEND 70, 123
- SMS 6, 16, 19, 59, 76, 87
- soft copy 47
- source system 20
- split mirror backup 47, 117
- STARTRBA 64, 102, 105, 111
- statement cache statistics 140
- STOSPACE 78, 85, 108

T

- target system 20
- templates 29, 35
- tp 9, 128
- Transport Management System (TMS) 122
- transports 128

U

- Unicode 11
- unique database names 23
- unit of recovery 116

V

- VCAT 6, 16, 59, 72

X

- XMAP member 84, 96

Archived



Redbooks

SAP on DB2 UDB for OS/390 and z/OS: Multiple Components in One Database (MCOB)

Archived



SAP on DB2 Universal Database for OS/390 and z/OS: Multiple Components in One Database (MCOD)



Hands-on scenarios to merge SAP components into an MCOD landscape

How to clone one SAP component using the Control Center

Recovery considerations in an MCOD landscape

The Multiple Components in One Database (MCOD) feature of SAP enables a reduction in the number of DB2 systems that need to be installed and maintained. This significantly simplifies overall database administration and is considered one of the major DB2 competitive advantages.

This IBM Redbook will help systems administrators, database administrators, managers, and operation staff to plan, implement, and administer an SAP MCOD landscape with DB2 Universal Database (UDB) for OS/390 and z/OS as the database management system.

We describe how to merge existing systems into a single DB2 subsystem. Two different methods are developed, each of them addressing different needs. For small-to-medium SAP systems where high availability is not a requirement, we explain how to use SAP tools. For large systems, where the down time needed by SAP standard procedures is not acceptable, we document a technique to merge SAP components without moving the data.

We also provide a cloning procedure using the Control Center. We show how to clone one component out of an MCOD landscape. We address the backup and recovery implications in an MCOD environment, to help database administrators plan accordingly. We also describe how to set up and use the Computer Center Management System (CCMS) in an MCOD landscape.

**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**