Siebel 7 with DB2 for z/OS: Database Implementation and Administration Guide

- Installing and customizing Siebel database for production
- Database administration
- Monitoring for performance

Viviane Anavi-Chaput
Steve Baker
Richard Corrihons
Rich Dale
Lou Doran
Debra Scott, Lee Siegmund
Xuan Nhut Tran, Marion Wilmot

ibm.com/redbooks
First Edition (March 2003)

This edition applies to z/OS 1.2, DB2 UDB for OS/390 and z/OS V7.1, DB2 Connect EE V7.2, AIX V5L, Windows 2000, and Siebel 704.

© 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

Notices ................................................................. vii
Trademarks ......................................................... viii

Preface ............................................................... ix
The team that wrote this redbook. ................................. ix
Become a published author ........................................ xi
Comments welcome ................................................ xi

Chapter 1. Overview of Siebel 7 with DB2 for z/OS database .... 1
1.1 Siebel 7 system architecture overview ........................ 2
1.2 Siebel 7 database implementation overview .................... 3
1.3 Siebel 7 database administration overview ..................... 3
1.4 DB2 for z/OS V7 functions that benefit Siebel 7 .............. 4
  1.4.1 Correlated subquery for UPDATE joins .................... 5
  1.4.2 Self-referencing subselect with UPDATE/DELETE ................ 5
  1.4.3 Online system parameters ................................ 6
  1.4.4 Scrollable cursors ......................................... 6
  1.4.5 MEMBER ID for sysplex scalability ....................... 6
  1.4.6 Realtime statistics ....................................... 7
  1.4.7 FETCH FIRST n ROWS .................................. 7
  1.4.8 Asynchronous preformatting ................................ 7
  1.4.9 Catalog row-level locking ................................ 8
  1.4.10 REORG enhancements ................................... 8
  1.4.11 Statistics HISTORY .................................... 9
  1.4.12 New monitoring indicators ............................... 10
1.5 DB2 Connect EE V7 database connectivity ...................... 10
  1.5.1 DB2 database connection types ............................ 11
  1.5.2 DB2 Connect EE setup overview ............................ 13
1.6 Siebel 7 database connection pooling ......................... 18

Chapter 2. Installing the Siebel 7 system environment .......... 21
2.1 System configuration ........................................... 22
  2.1.1 Siebel 7 client tier ...................................... 22
  2.1.2 Siebel 7 mid-tier ........................................ 23
  2.1.3 Siebel 7 database tier ................................... 23
2.2 z/OS security setup ........................................... 24
  2.2.1 RACF - userids and groups ............................... 25
  2.2.2 DB2 UDB for z/OS and OS/390 authorization .............. 25
2.3 DB2 for OS/390 and z/OS setup ................................ 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5 Inactivation of unused indexes</td>
<td>94</td>
</tr>
<tr>
<td>Chapter 6. Partitioning method using DB2 utilities</td>
<td>97</td>
</tr>
<tr>
<td>6.1 Planning steps for partitioning Siebel tables</td>
<td>98</td>
</tr>
<tr>
<td>6.2 Identifying candidate tables and partition columns</td>
<td>98</td>
</tr>
<tr>
<td>6.3 Planning for REXX procedures</td>
<td>99</td>
</tr>
<tr>
<td>6.4 Unloading the tables</td>
<td>100</td>
</tr>
<tr>
<td>6.4.1 Reference table</td>
<td>100</td>
</tr>
<tr>
<td>6.4.2 Unloading the data</td>
<td>101</td>
</tr>
<tr>
<td>6.5 Creating new tables with partitions</td>
<td>102</td>
</tr>
<tr>
<td>6.6 Loading the tables</td>
<td>103</td>
</tr>
<tr>
<td>6.7 Reorganizing the tables</td>
<td>104</td>
</tr>
<tr>
<td>Chapter 7. Monitoring dynamic SQL</td>
<td>105</td>
</tr>
<tr>
<td>7.1 Monitoring dynamic SQL in the DSC</td>
<td>106</td>
</tr>
<tr>
<td>7.1.1 DB2 PM Workstation Online Monitor tool</td>
<td>106</td>
</tr>
<tr>
<td>7.1.2 Explaining SQL statement from the cache</td>
<td>109</td>
</tr>
<tr>
<td>7.1.3 Identifying inefficient SQL</td>
<td>112</td>
</tr>
<tr>
<td>7.1.4 Considerations on parameter markers</td>
<td>113</td>
</tr>
<tr>
<td>7.2 Siebel SQL tracing</td>
<td>114</td>
</tr>
<tr>
<td>7.2.1 Object Manager</td>
<td>114</td>
</tr>
<tr>
<td>7.2.2 Dedicated Web client</td>
<td>119</td>
</tr>
<tr>
<td>7.2.3 EIM performance considerations</td>
<td>119</td>
</tr>
<tr>
<td>Chapter 8. Monitoring the Siebel 7 database</td>
<td>121</td>
</tr>
<tr>
<td>8.1 DB2 installation considerations</td>
<td>122</td>
</tr>
<tr>
<td>8.1.1 Database objects</td>
<td>122</td>
</tr>
<tr>
<td>8.1.2 Free space</td>
<td>124</td>
</tr>
<tr>
<td>8.1.3 EDM pool size</td>
<td>124</td>
</tr>
<tr>
<td>8.1.4 Buffer pools</td>
<td>125</td>
</tr>
<tr>
<td>8.2 DB2 statistics</td>
<td>126</td>
</tr>
<tr>
<td>8.2.1 Why collect statistics</td>
<td>126</td>
</tr>
<tr>
<td>8.2.2 What statistics should you gather for Siebel</td>
<td>126</td>
</tr>
<tr>
<td>8.2.3 Monitoring VSAM data set extents</td>
<td>126</td>
</tr>
<tr>
<td>8.3 DB2 history statistics</td>
<td>127</td>
</tr>
<tr>
<td>8.3.1 Monitoring space growth</td>
<td>128</td>
</tr>
<tr>
<td>8.3.2 Monitoring the rebuild of the compression dictionary</td>
<td>129</td>
</tr>
<tr>
<td>8.4 DB2 realtime statistics</td>
<td>129</td>
</tr>
<tr>
<td>8.5 Database reorganization</td>
<td>130</td>
</tr>
<tr>
<td>8.6 Siebel-DB2 database recovery</td>
<td>132</td>
</tr>
<tr>
<td>8.6.1 Recovery to currency</td>
<td>132</td>
</tr>
<tr>
<td>8.6.2 Disaster recovery</td>
<td>132</td>
</tr>
<tr>
<td>8.6.3 Point-in-time recovery</td>
<td>133</td>
</tr>
</tbody>
</table>
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:

- AIX®
- AIX 5LTM
- CICS®
- DB2®
- DB2 Connect™
- DRDA®
- FlashCopy®
- Hummingbird®
- IBM®
- @serverTM
- IMS™
- MVS™
- Netfinity®
- OS/390®
- Parallel Sysplex®
- QMF™
- RACF®
- Redbooks™
- Redbooks (logo)™
- RS/6000®
- S/390®
- z/OS™
- zSeries

The following terms are trademarks of other companies:

Siebel and the Siebel Logo are trademarks of Siebel Systems, Inc., and may be registered in certain jurisdictions.

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

In this IBM® Redbook, we describe the implementation, customization, and administration of the Siebel 7 database with DB2® UDB for z/OS™ and OS/390® V7 using AIX® V5L or Win2000 Siebel Servers, and we discuss the following topics:

- Installation tasks for setting up a three-tier configuration, with the application middle tier residing on AIX/Win2000 and the database tier residing on z/OS. The DB2 Connect™ EE V7 setup required to connect the Siebel application server to the DB2 database on z/OS is also discussed.
- Customization of the Siebel schema for a production environment.
- Migration from a development system to a production system.
- Database administration tasks, including:
  - Identifying and inactivating unused indexes
  - Table space partitioning
  - Monitoring dynamic SQL
  - Monitoring the Siebel 7 database
  - Statistics gathering
  - Reorg, backup, and recovery

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Viviane Anavi-Chaput** is a Senior IT Specialist for BI and ERP at the IBM ITSO Poughkeepsie Center. She writes extensively, teaches worldwide, and presents at international conferences on all areas of Business Intelligence and DB2 for OS/390. She was also an ITSO Specialist for DB2 at the San Jose Center from 1990 to 1994.

**Steve Baker** is a Senior Technical Consultant for Baker Consultants Ltd. in the United Kingdom. He has 22 years of experience working with databases and middleware in a variety of environments including AIX, MVS™, and Linux. Steve's current focus is Siebel support, legacy system integration, and data migration.
Richard Corrihons is a DB2 for z/OS Specialist with the IBM Products and Solutions Support Center (PSSC) in Montpellier, France. Richard specializes in customer proofs of concept and benchmarks mainly in the CRM and ERP areas. He holds an Electronic Engineering degree from the Conservatoire National des Arts et Metiers.

Rich Dale is a Senior IT Specialist in the IBM Americas Data Management organization. He has been with IBM for 28 years and has extensive experience in DB2 on MVS, AIX, and Intel-based operating systems. He has specialized in Siebel systems running on DB2 since 1999.

Lou Doran is a Consulting IT Specialist in the Americas e-business Technical Sales organization. He has 30 years of experience in the IT field, and has spent the last six years working with the ERP/CRM Solutions on the IBM @server™ zSeries with DB2.

Debra Scott is a DB2 specialist working on the Technical Alliance Management team in the IBM Data Management Support & Services organization at Silicon Valley Lab in California. She has 15 years of experience in the field of IT, including DB2 for OS/390. Her current focus is on Siebel support and Technical Support Marketing for Siebel Data Management.

Lee Siegmund is a Consulting Marketing Support Representative and Certified IT Specialist from the IBM DB2 Relational Support Unit of the IBM Dallas Systems Center. He has been with IBM for 33 years, and for the past 19 years he has supported DB2, from Version 1 Release 1 to the current Version 7.

Xuan Nhut Tran is a lead DBA at Siebel Systems in California. He has 17 years of experience in IT, with extensive experience in DB2 for OS/390. His primary responsibilities are in the areas of high availability, performance, and disaster recovery.

Marion Wilmot is a member of the Technical Support group at Siebel Systems in UK. She has many years of experience with database and mainframe technologies, including DB2 for OS/390.

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

Richard Conway, Vasilis Karras
IBM International Technical Support Organization, Poughkeepsie, NY, USA

Miguel de Andrade
Siebel, Inc., UK

Blaine Lucyk, Svetlana Sicular
Siebel, Inc., CA, USA
James Guo, Eva Hu, Gopal Krishnan, Rick Levin, James Pickel, Hugh Smith, Annie Tsang, Yun Wang, Casey Young
IBM Silicon Valley Laboratory, CA, USA

Rick Kumar
IBM Competency Center for Siebel, CA, USA

Eric Cicchiello, Jean-Marc Darees, Stefano Delle Chiaie, Noel Richard
IBM Products and Solutions Support Center (PSSC), Montpellier, France

Namik Hrle, Norbert Jenninger
IBM Boeblingen Laboratory, Germany

Kent Burel, Franck Hatchett, Vincent Patruno, Jeff Wiese
IBM IT Specialist, ATS and FTSS, USA

Thanks also to Terry Barthel, Alison Chandler, and Alfred Schwab for their editorial assistance, and Ella Buslovich for her graphics assistance.

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
Overview of Siebel 7 with DB2 for z/OS database

In this section we give an overview of Siebel 7 architecture and introduce the implementation tasks and administration issues to consider. We also summarize DB2 V7 enhancements used by Siebel 7.

- Overview of Siebel 7 architecture, implementation and administration
- DB2 for z/OS V7 functions used by Siebel 7
- Siebel 7 database connectivity overview
1.1 Siebel 7 system architecture overview

Figure 1-1 describes Siebel 7 system architecture on IBM AIX and DB2 UDB for z/OS and OS/390.

For its client connectivity, Siebel uses Resonate as a distributor of messages received from the Web gateway servers to the application servers via the Siebel gateway.

For its database connectivity, Siebel uses the DB2 Connect Enterprise Edition (EE) database connector to connect the application server to the database server on DB2 UDB for z/OS and OS/390. DB2 Connect EE servers provide transparent database connections for many simultaneous requests from application servers.

Using connection concentrator (also called transaction pooling), DB2 Connect EE provides connection load balancing to the DB2 data sharing group in a Parallel Sysplex® configuration, as well as failover routing in case of failure of a sysplex member.
This book focuses on Siebel 7 database implementation with DB2 for z/OS and OS/390, including connectivity to the DB2 database using DB2 Connect EE V7.

We also discuss DB2 database administration issues in the context of Siebel 7.

1.2 Siebel 7 database implementation overview

The following sections of the book describe Siebel 7 installation using the AIX V5/Windows 2000 application server and DB2 for z/OS and OS/390 V7 database on series, which is the most popular three-tier configuration implemented for Siebel with a database on IBM® zSeries.

- Siebel 7 basic three-tier installation is described in Chapter 2, “Installing the Siebel 7 system environment” on page 21.
  
  We describe the client-tier, mid-tier, and database-tier implementations.

- Schema modifications done after the basic installation are described in Chapter 3, “Modifying the Siebel schema for a production environment” on page 67.
  
  For this exercise, we show how to partition tables that were originally created as non-partitioned during Siebel 7 basic installation. We discuss partitioning techniques for small and large tables.

- Migrating schema modifications from development to production is described in Chapter 4, “Migration from development to the production system” on page 75.

1.3 Siebel 7 database administration overview

The administration sections of the book consider the major database administration issues a DBA will have to handle in the context of Siebel 7 using DB2 on zSeries.

- Partitioning techniques for small and large tables are described in Chapter 3, “Modifying the Siebel schema for a production environment” on page 67.

- Managing unused indexes is described in Chapter 5, “Identifying and removing unused, non-unique indexes” on page 89.

- Dynamic SQL tuning and monitoring are described in Chapter 7, “Monitoring dynamic SQL” on page 105.

- DB2 database administration issues are described in Chapter 8, “Monitoring the Siebel 7 database” on page 121, including:
  
  - Initial tuning of DB2 objects and subsystem parameters
– How to collect and use DB2 statistics to monitor the database environment
– Database reorganization
– Database backup and recovery techniques

We gathered many application and SQL tuning recommendations at the IBM zSeries Benchmark Centers in Montpellier, France, based on lessons learned from the proof of concept and performance benchmarks they run for Siebel customers.

We consulted the DB2 Silicon Valley Laboratory to get their recommendations for the initial tuning of the DB2 database for optimal performance with Siebel applications. Those recommendations were based on the performance benchmarks the DB2 Lab runs specifically for the Siebel-DB2 database.

Finally, we had excellent support from Siebel developers, expert services, and system administrators to constantly fuel the project with the best practices they would recommend to technical professionals having to implement, customize, tune and monitor a Siebel-DB2 environment—they are all documented in this book!

Before discussing Siebel 7 database implementation and administration, however, we first introduce the new features of DB2 for z/OS and OS/390 that benefit Siebel 7. Understanding these enhancements will help you to understand the implementation options and administration decisions suggested and explained further in this book.

1.4 DB2 for z/OS V7 functions that benefit Siebel 7

DB2 V7 brings several enhancements used by Siebel 7. The following describes the DB2 V7 functions that are beneficial to Siebel on z/OS and OS/390:

- Correlated subquery for UPDATE joins
- Self-referencing subselect with UPDATE/DELETE
- Online system parameters
- Scrollable cursors
- MEMBER-ID for sysplex scalability
- Realtime statistics
- FETCH FIRST N ROWS
- Asynchronous preformatting
- Catalog row-level locking
- REORG enhancements
- Statistics HISTORY
- New monitoring indicators
In the following sections, we explain these in more detail and offer examples of usage.

1.4.1 Correlated subquery for UPDATE joins

Application systems, such as Siebel, require certain stage 2 subquery predicates to be transformed into join operations. The requirement is for UPDATE, DELETE, and SELECT queries with IN, =ANY, or EXISTS subquery predicates that are currently evaluated as stage 2 predicates.

By transforming the query into a join of the outer query table and the subquery table, DB2 can significantly improve performance in many cases because it can choose the most efficient join order for the tables. Users can rewrite SELECT statements of this type as joins in some cases, but they cannot do the same for UPDATE or DELETE.

Example

UPDATE T1 A
SET A.C1 = 1
WHERE A.C2 IN (SELECT B.C2
FROM T2 B
WHERE A.C3 = B.C3);

Siebel applications greatly benefit from this enhancement. The transaction router, EIM, and some client SQL use this function.

1.4.2 Self-referencing subselect with UPDATE/DELETE

Allow DB2 UPDATE/DELETE FROM TABLE with a subselect from the same table.

Example 1

Give each employee whose salary is below the average a salary increase of 10%.

UPDATE EMP X
SET SALARY = 1.10
WHERE SALARY < (SELECT AVG(SALARY)
FROM EMP Y
WHERE X.WORKDEPT = Y.WORKDEPT);

Example 2

For each department, delete the employee with the most absences.

DELETE FROM EMP X
WHERE ABSENT = (SELECT MAX(ABSENT)
FROM EMP Y
WHERE X.WORKDEPT = Y.WORKDEPT);
This enhancement makes SQL in DB2 more powerful, thus reducing the need for temporary tables and application programming. Siebel's EIM import and delete use this function.

1.4.3 Online system parameters

DB2 V7 makes it possible to change the subsystem parameters dynamically when DB2 is currently up. The new \texttt{-SET SYSPARM} command allows the reloading of the DSNZPxxx (subsystem parameters) load module.

With previous DB2 versions, you had to stop Siebel application servers to recycle (stop and start) DB2, and then restart Siebel application servers, in order to take the new subsystem parameters into account. This is not needed with DB2 V7. Thus, this function provides more availability and flexibility to Siebel applications.

Use the DB2 command \texttt{SET SYSPARM} to change the system parameters values.

1.4.4 Scrollable cursors

Scrollable cursors provide application programs with a mechanism to have random access to a result table with simple SQL and simple logic. Scrollable cursors are most useful to screen-based applications in which the user scrolls backwards and forwards through the data.

Previous DB2 versions provided forward-only cursors, which force screen-based applications to cache the result table locally and perform scrolling themselves. Alternatively, to access a previously fetched row, applications had to close the cursor, reopen the cursor, and fetch forward until the desired row was fetched.

The scrollable cursors provide a solution that enables applications to use a powerful new set of SQL to fetch data via cursor in random order—with optimal performance, no extra cache requirements in the application, and simple logic, while also providing a means to indicate whether the fetched row must be current.

1.4.5 MEMBER ID for sysplex scalability

The sysplex MEMBER ID is now utilized by Siebel 7 for partitioning transaction logging table \texttt{S\_DOCK\_TXN\_LOG} so the logging will take place on the same member of the data sharing group. This is beneficial for Siebel remote mobile users because all transactions are being logged and stored in the \texttt{S\_DOCK\_TXN\_LOG} table for the mobile users’ synchronization.
Siebel table S_DOCK_TXN_LOG is used to keep track of all changed records for Siebel 7 remote mobile users. Note that, by default, transaction logging is ON.

If the user is not using remote mobile users, this table is not used and the logging activity should be disabled by turning Siebel Transaction logging OFF.

1.4.6 Realtime statistics

Most DBA operations are triggered by analyzing catalog statistics collected and updated by RUNSTATS. However, this approach gives no indication when RUNSTATS itself should be run, so the statistics can quickly become outdated and misleading.

The realtime statistics tables and standard catalog tables include new columns that should be used when deciding when a table space or index needs reorganization:

- Number of extents
- Number of data sets
- Number of pseudo-deleted index entries
- Number of near-off and far-off index page splits

An index split page is considered “near-off” the present page if the difference is less than 16, and “far off” if the difference is 16 or more. If 10 percent of all pages are near-off or far-off, consider performing an index reorg.

1.4.7 FETCH FIRST n ROWS

The new FETCH FIRST n ROWS option speeds up fetch processing in cases in which the number of rows that need to be fetched is known in advance. Another usage is to check if a table is empty.

Use the following query:

```
SELECT 1 FROM TAB FETCH FIRST 1 ROW.
```

Unless explicitly specified, OPTIMIZE FOR n ROWS is implied.

1.4.8 Asynchronous preformatting

Prior to DB2 V7, mass insert processing is affected by space preformatting. Namely, when the formatted space is exhausted, a synchronous task for formatting the additional space is scheduled. Inserts must wait until this task completes its processing. During heavy insert processing, these situations can appear often and significantly slow down the overall process.
With DB2 V7, the preformatting is an asynchronous process that gets triggered before the formatted space is exhausted. When a new page for insert is close to the end of formatted pages and there is more allocated and unformatted space in the data set, an asynchronous format task will format the next range of pages. This will ensure that insert rarely has to wait for the page to be formatted.

Additionally, the space search performance is improved because the search will not happen at each logical extent, but only at each physical extent. This feature is internal to DB2, and no user specification is needed.

### 1.4.9 Catalog row-level locking

Earlier DB2 versions do not allow ALTER LOCKSIZE against catalog table spaces. In most cases, the catalog table spaces use page-level locking and that can create lock contention, especially during massive, parallel DDL executions.

In DB2 V7, ALTER TABLESPACE LOCKSIZE is supported for table spaces of the DB2 catalog that do not contain links—all the table spaces except SYSDATABASE, SYSPLAN, SYSVIEWS, SYSDBAUT, and SYSGROUP.

Attempts to ALTER TABLESPACE LOCKSIZE for one of those table spaces still returns the SQL code -607.

### 1.4.10 REORG enhancements

DB2 V7 brings the following REORG enhancements.

**Avoid data set renames in the switch phase**

Before DB2 V7, online REORG tried to fully preserve the underlying data set names for table space and indexspaces via a series of AMS RENAMEs that took place during the REORG switch phase. But this process can cause significant contention in the system because the duration of the switch phase (during which the object being reorganized cannot be accessed) is directly proportional to the number of underlying data sets. At least one data set exists for each table space and each index.

With DB2 V7, you can avoid data set renames in the switch phase. The fifth data set qualifier (called the instance node) changes from a fixed value (I0001) to a variable value alternating between I0001 and J0001.

In the UTILINIT phase, if the original object's instance node starts with an “I”, the shadow object's instance node starts with a “J”, and vice versa. In the switch phase, the catalog is updated (the IPREFIX column in SYSTABLEPART and SYSINDEXPART), and the shadow object becomes the active object. In the UTILTERM phase, the original object is deleted.
Specifying the FASTSWITCH option on the REORG statement activates the new no-rename behavior in the REORG switch phase.

**REORG timeout and drain retry options**

Before DB2 V7, the timeout value for REORG was at least equal to the value specified as the transactions timeout. It was calculated as a product of the system parameters IRLMRWT (the transactions timeout value) and UTIMOUT, the utility timeout factor (greater or equal to 1) that allows REORG to wait longer for a resource than a regular statement would.

In DB2 V7, the new REORG option DRAIN_WAIT (with possible values of 1 to 1,800 seconds) specifies how long REORG will wait to acquire drains. (Draining is a mechanism used to take over an object and serialize access to it.)

The new option applies to drains in the UNLOAD, LOG, and SWITCH phases. When multiple objects need draining (for example, a table space and its index), the DRAIN_WAIT value is the aggregate time.

Use the DRAIN_WAIT option in conjunction with the new retry logic that preserves REORG processing done before DRAIN_WAIT was exhausted. Otherwise, after a timeout, the REORG process would terminate without accomplishing anything.

With the retry logic, the failed draining can be periodically retried. The new option RETRY_DELAY specifies how long REORG will wait before retrying the failed drain process. Another option, RETRY, specifies how many times this whole cycle would be repeated. During the RETRY_DELAY period, online transactions continue to access the data.

REORG will have more opportunities (RETRY times) to drain the resources and complete successfully. If the switch phase drain fails, online REORG will go back into the log apply phase, allowing the writers to resume. After the RETRY_DELAY time, writers will have to be drained again, followed by the last log apply phase and, finally, a readers drain.

**1.4.11 Statistics HISTORY**

DB2 V7 RUNSTATS includes a new option, HISTORY. If specified, every time the corresponding catalog tables are inserted or updated, the following fields are copied to the new catalog history tables: SYSCOLDIST_HIST, SYSCOLUMNS_HIST, SYSINDEXES_HIST, SYSINDEXPART_HIST, SYSINDEXSTATS_HIST, SYSLOBSTATS_HIST, SYSTABLEPART_HIST, SYSTABLES_HIST, and SYSTABSTATS_HIST.
Data can now be stored in the DB2 catalog on a historical basis that will enable you to do trend analysis. When RUNSTATS stores new information in the DB2 catalog tables, it will also store that information in the DB2 Version 7 catalog history tables, so that the catalog will hold the current statistics, and the catalog history tables will hold historical (including current) statistics. The MODIFY STATISTICS command will be used to delete the outdated statistics in the history tables.

This results in preserving the relevant statistics data across multiple RUNSTATS executions, creating a basis for monitoring the database object growth and providing assistance in determining optimal allocation quantities. Additionally, the historical data could be used for reinstating old catalog statistics in case of access-path degradation.

RUNSTATS can run from the Siebel server, or natively.

1.4.12 New monitoring indicators

New performance indicators for faster identification of potential tuning areas include:

- Improved accounting class 3 granularity for data sharing global contention for locks. The “global contention for locks” bucket is broken out into six separate buckets: Parent L-lock; Child L-lock; other L-lock; Pageset/partition P-lock; Page P-lock; other P-lock.
- Page P-lock counters reported at the group bufferpool (GBP) level for accounting and statistics. Because page P-locks are such a critical performance factor in data sharing environments, reporting the P-lock activity at the GBP level will enable more effective monitoring and tuning of P-locking activity. The new indicators include page P-locking requests, suspensions, and negotiations for space map pages, data pages, and index leaf pages.
- Number of times pages are added to the logical page list per buffer pool.
- Internal resource lock manager settings such as PC, DEADLOCK time, and cycle and actual TIMEOUT are now available via IFI.
- Faster detection of lock conflicts in the system.
- Synchronizing DB2 statistics trace across data sharing members (system parameter SYNCVAL).

1.5 DB2 Connect EE V7 database connectivity

We describe, in this section, the DB2 Connect EE V7.2 database connectivity options you have between the application server and the DB2 for z/OS V7 data
sharing environment on z/OS. We also describe the new DB2 Connect EE V7 functions available for building connections between the Siebel application server and the DB2 database on z/OS.

1.5.1 DB2 database connection types

Figure 1-2 provides an overview of the different types of DB2 database connections.

Now let's discuss the following connection types in some detail:

- Dedicated connections
- Sysplex-aware connections
- Connection concentrator also called transaction pooling

Dedicated connections

Prior to connection concentrators, each time a client needed a connection to DB2, an agent task would be created on the DB2 Connect EE server. This agent serviced the client's connection requests to DB2 until the client issued a disconnect.

During the entire connection time, all system resources (for example, CPU and memory) are tied up until disconnect occurs, regardless of whether or not the client is actively using the target DB2 database.

All connection requests from a given application server are always routed to the same DB2 member. This is reflected in the example of App Server1, in Figure 1-2 on page 11, which is configured to always connect to D7R1.
Sysplex-aware connections

Connection requests coming from a given application server can be handled more intelligently. When DB2 Connect EE is sysplex-aware, requests can be routed to any available DB2 data sharing group member. The most suitable member gets the connection.

DB2 Connect EE in this case is aware of the sysplex and data sharing configuration, and receives information from both DB2 and Workload Manager (WLM) to enable connection load balancing between the members of the data sharing group. DB2 Connect EE receives this information at the creation of each new connection. This is the example shown for App Server2, in Figure 1-2 on page 11, whose database connection requests can go either to D7R1 or D7R2.

Connection concentrator (or transaction pooling)

DB2 Connect EE V7 uses connection concentrator (which is also called transaction pooling). This technique provides reuse of DB2 Connect server agents, and avoids repeated connect/disconnect processing.

In this case, after a client commits, DB2 Connect EE leaves the agent task with its corresponding host connection in place. When another client connection request is received, DB2 Connect EE checks to see if there is a free pooled agent task available. If there is, DB2 Connect EE then signs on the new user over the existing host connection, and processing continues from there with a lower cost than a full sign-on. The example of App Server3 in Figure 1-2 on page 11 illustrates connection concentrator.

DB2 Connect EE V7 connection concentrator provides improved load balancing in sysplex data sharing configurations. When DB2 Connect EE is accessing DB2 data sharing groups in connection concentrator mode, it is frequently receiving sysplex status information on the members of the group. DB2 Connect EE uses this information to continually balance the number of connections it has to each member of the group, as well as to determine to which member a client's next transaction should be routed.

The main benefit of the connection concentrator is that when a connection drops, only those logical agents that are processing transactions in the failed data sharing group member experience difficulty; all other clients remain connected to DB2 or continue their transactions via other members of the data sharing group as identified in the server list.

Connection concentrator is not recommended by Siebel.
1.5.2 DB2 Connect EE setup overview

It is important to have a good naming convention defined for the DB2 data sharing group components because after installation, some names (such as the group name and member names) cannot be changed.

In the mid-tier AIX environment where the DB2 Connect EE setup takes place, the same consideration should be applied to the DB2 Connect EE catalog entries and ODBC settings used. To avoid confusion, it is best to use the same names defined at the z/OS level platform whenever possible. The DB2 and Siebel component names we used for this discussion are as follows:

- **ClientA**: First Siebel application server (AIX)
- **ClientB**: Second Siebel application server (AIX)
- **DB7R**: DB2 location name (z/OS)
- **wtsc04oe.itso.ibm.com**: First host name (z/OS)
- **D7R1**: DB2 data sharing group member on z/OS host wtsc04oe.itso.ibm.com
- **wtsc67oe.itso.ibm.com**: Second host name (z/OS)
- **D7R2**: DB2 data sharing group member on z/OS host wtsc67oe.itso.ibm.com

Table 1-1 shows the DB2 database manager configuration parameters which need to be set for the DB2 Connect EE instance.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQRIOBLK</td>
<td>Maximum size network I/O blocks</td>
<td>32767</td>
</tr>
<tr>
<td>DIR_CACHE</td>
<td>Cache directory information</td>
<td>YES</td>
</tr>
<tr>
<td>DRDA_HEAP_SZ</td>
<td>DRDA® heap size</td>
<td>512</td>
</tr>
<tr>
<td>MON_HEAP_SZ</td>
<td>Database system monitor heap size</td>
<td>1024</td>
</tr>
</tbody>
</table>

A Siebel component parameter named OM Read Config from Server for the Object Manager should be set to False. When set to True, settings are taken from the Siebel Enterprise Server and settings from Siebel configuration files are ignored.

With the Siebel application server using DB2 Connect agents, we must pass SQL to DB2 agents in DB2 Connect by using the following command:

```
db2set DB2CONNECT_IN_APP_PROCESS=NO
```
This tells DB2 Connect EE to use DB2 connection agents and reference the DB2 database manager configuration for parameter settings.

The traditional DB2 API establishes connectivity to DB2 directly via application executables. For example, when we write a traditional COBOL program to access DB2, the DB2 API is used. The program is built and the executable module communicates directly with DB2. The DB2 Connect is in the application process. DB2CONNECT_IN_APP_PROCESS=YES represents this situation.

Dedicated connections setup
This option is the simplest form of DB2 connection between Siebel Application Servers and the DB2 database. Each Siebel application server client uses a dedicated DB2 Connect EE (real agent). The real agent references a dedicated data sharing group member to gain access to the shared database.

Figure 1-3 on page 15 illustrates a dedicated connection.

- The Siebel App Server Client A uses DB2 connection (DB7R) to access the z/OS database (location DB7R) via the data group member D7R1 on host wtsc04oe.itso.ibm.com, port 33750.
- The Siebel App Server Client B uses DB2 connection (DB7R) to access the z/OS database (location DB7R) via the data group member D7R2 on host wtsc67oe.itso.ibm.com, port 33750.

The DB2 connection is named DB7R on each instance of DB2 Connect EE to reference the physical database. This is a convention; the names could be anything.
The DB2 Connect setup for dedicated connections is as follows:

- The DB2 registry variable DB2CONNECT_IN_APP_PROCESS must be disabled (do not set it to NO, for example, because you would be wasting memory and consuming extra CPU for no benefit).
- As db2admin on the Siebel Client A machine, use DB2 Connect EE to issue catalog entries:
  
  ```
  catalog tcpip node d7r1 remote wtsc04oe.itso.ibm.com server 33750
  catalog dcs db db7r as db7r at node d7r1 authentication dcs
  ```

  **Note:** The port number (33750) must match the port number defined for the data group member. This must be the same for all DB2 hosts defined to all environments participating in data sharing. We have named the node d7r1 to match the data group member; this is a convention, but not required.

  ```
  catalog db db7r as db7r at node d7r1 authentication dcs
  ```

  **Note:** The first DB7R names the dcs db and can be any name you wish; the second must reference the DB2 location for the z/OS database.
As db2admin, connect to DB2 for z/OS and bind the utilities and applications:

- connect to db7r
- bind $HOME/bnd/@ddcsmvs.lst blocking all grant public

As db2admin on the Siebel client B machine, use DB2 Connect EE to issue catalog entries:

- catalog tcpip node d7r2 remote wtsc67oe.itso.ibm.com server 33750
- catalog db db7r as db7r at node d7r2 authentication dcs
- catalog dcs db db7r as db7r

This configuration has no load balancing capability. Every Siebel process or user that is routed through Siebel App Server Client A will access DB2 via its connection through data sharing group member d7r1. App Server Client B will access DB2 via its connection through data sharing group member d7r2.

This configuration has no failover possibilities, either. If a member fails, all connections routed through this member will also fail.

**Sysplex-aware connections setup**

The configuration for this option is the same as a dedicated connection, except the catalog command includes the "sysplex" parameter. This syntax makes the catalog entry “sysplex-aware”.

- catalog tcpip node d7r2 remote wtsc67oe.itso.ibm.com server 33750
- catalog db db7r as db7r at node d7r2 authentication dcs
- catalog dcs db db7r as db7r parms ',,,,,,sysplex'

Sysplex-aware means that DB2 Connect EE gets sysplex information and is capable of load balancing the connections between the different members of the data sharing group participating in the sysplex configuration.

Also, the DB2 registry variable DB2CONNECT_IN_APP_PROCESS must be enabled and set to NO if local processes to the DB2 Connect EE server code want to be sysplex-aware.

Figure 1-4 illustrates a dedicated and a sysplex-aware connection.
Client B has two host connections. When the Siebel Client B requests information from the database DB7R, DB2 Connect EE has the choice of using data sharing group member D7R1 or D7R2 to access the database. Siebel Client A is not aware of sysplex, and all connect activity is accommodated by data sharing group member D7R1.

When Client B initially contacts DB7R, it is passed a listing of all connections available for use from the Workload Manager (WLM). The Workload Manager has technology in the z/OS environment for maintaining a list of servers in a ready state.

The Workload Manager also sets weights for each server according to configurable policies like workload and capacity. The sysplex routes requests between datasharing group members according to these policies; refer to OS/390 Workload Manager Implementation and Exploitation, SG24-5326, for details.

When this facility is used, the hostname defined in the `catalog tcpip node...` command establishes the primary data sharing group member for the connection. If this connection fails, or if the data sharing group member is otherwise unavailable, then the connection fails and cannot be reestablished until these are reinstated.
Connection concentrator setup
In order to use connection concentrator (or transaction pooling), the DB2 registry variable DB2CONNECT_IN_APP_PROCESS must be set to NO.

Table 1-2 lists the DB2 Connect parameters to define connection concentrator.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_LOGICAGENTS</td>
<td>Maximum number of logical agents</td>
<td>Set to maximum application client agents</td>
</tr>
<tr>
<td>(new parm for transaction pooling)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXAGENTS</td>
<td>Maximum number of worker agents</td>
<td>Concurrent connections to DB2 for z/OS</td>
</tr>
<tr>
<td>NUM_POOLAGENTS</td>
<td>Maximum number of idle pool agents</td>
<td>Concurrent connections to DB2 for z/OS</td>
</tr>
<tr>
<td>MAXCOORDAGENTS</td>
<td>Maximum number of coordinating agents</td>
<td>Concurrent connections to DB2 for z/OS</td>
</tr>
</tbody>
</table>

Normally, MAX_LOGICAGENTS is set to the number of Siebel clients, and the other three parameters are set to the number of connections to DB2. The MAXAGENTS, MAXCOORDAGENTS, and NUM_POOLAGENTS should all be the same; this is the number of real agents that are retained in the pool.

1.6 Siebel 7 database connection pooling

Siebel 7 connection pooling (Database Connection Multiplexing) saves the overhead of opening and closing a new connection for every query. If you do not enable this option, each query sent to the underlying database will open a new connection.

To allow a single database connection to remain open for the specified time for use by future query requests, Siebel connection pooling must be enabled. Siebel recommends a ratio of 10 users per database connection. In other words, set the three DB Multiplex settings to number of concurrent users/10. This is also assuming 30-second think time between user operations. Adjust the ratio proportional to the actual user think time for a particular implementation.

Set the parameters MinSharedDbConns, MaxSharedDbConns, MinTrxDBConns for each component, using srvmgr to <target#users>/10.
Table 1-3  Siebel database connection multiplex settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTrxDBConns</td>
<td>Minimum number of dedicated database connections</td>
<td>Minimum number of dedicated database connections used for write/update/delete operations</td>
</tr>
<tr>
<td>MinSharedDbConns</td>
<td>Minimum number of shared database connections</td>
<td>Minimum number of shared database connections used for read-only operations</td>
</tr>
<tr>
<td>MaxSharedDbConns</td>
<td>Maximum number of shared database connections</td>
<td>Maximum number of shared database connections used for read-only operations</td>
</tr>
</tbody>
</table>

For example, set the DBConn parameters to 100 for 1000 users, using a ratio of 10 users/DbConns. To disable database connection pooling, set parameters back to -1.

For more information on setting up Siebel database connection pooling, refer to Technical Note 388: How to configure Siebel Object Manager (SOM) in Siebel 7 on Siebel Support Web.
Chapter 2. Installing the Siebel 7 system environment

In this chapter, we describe the installation tasks required for Siebel 7 using AIX V5L application servers and a DB2 UDB for z/OS and OS/390 V7 database server. The z/OS environment was a Parallel Sysplex, and we used a two-member DB2 data sharing group. We considered this system to be our “development” environment.

This chapter discusses the following:

- System configuration
- z/OS security setup
- DB2 UDB for z/OS and OS/390 V7 setup
- Mid-tier setup
- Installation of Siebel 7 servers
- Installation of Siebel 7 Web clients
- Validation of Siebel 7 database
2.1 System configuration

Figure 1-1 outlines the configuration we used at the ITSO for the installation of Siebel 7.

2.1.1 Siebel 7 client tier

The Siebel 7 client tier includes two types of clients, with the following hardware and software.

**Siebel 7 Web client**

- **Hardware**
  - Netfinity® A40 PIII 1.0 GHz processor
- **Software**
  - Windows 2000 - Service Pack 3
  - IE5.5 Service Pack 2
  - JVM3805, QFE Q303750 and QFE Q320721
Siebel 7 Tools Client

- **Hardware**
  - Netfinity A40 PIII 1.0 GHz processor
- **Software**
  - Windows 2000 - Service Pack 3
  - IE5.5 Service Pack 2
  - JVM3805, QFE Q303750 and QFE Q320721
  - DB2 Connect PE V7.2 Fix Pack 5s or DB2 V7.2 Runtime Client Fix Pack 5s with an available DB2 Connect EE V7.2 Gateway Server

2.1.2 Siebel 7 mid-tier

The Siebel 7 mid-tier includes the following.

- **Hardware**
  - RS/6000® S7A with 2 GB Memory, (3) 18.2 GB Drives, (2) 9.1 GB Drives, and a 1 Gb Enet Adapter
- **Software**
  - AIX 5.1.0.25 (Siebel requires minimum maintenance level 1)
  - Samba 2.2.3a
  - IBMIHS 2.0.30 with Siebel 7 Web Server Plug-in
  - DB2 Connect EE V7.2 Fix Pack 5s
  - Siebel 7 Gateway Server (minimum patch level 7.0.4.100)
  - Siebel 7 Enterprise Server (minimum patch level 7.0.4.100)
  - Siebel 7 Application Server (minimum patch level 7.0.4.100)
  - Siebel 7 Web Server Extension (minimum patch level 7.0.4.100)

2.1.3 Siebel 7 database tier

The DB2 UDB for z/OS and OS/390 environment includes the following:

- **Hardware**
  - 9672-x77 G6 (two LPARs)
  - 1 Gb OSA Adapter
  - 6 GB DASD
  - Sysplex environment
  - Cross-System Coupling Facility (XCF)
- **Software**
  - z/OS 1.2
  - DB2 UDB for z/OS and OS/390 V7 with maintenance from APAR II13012
  - Two-member data sharing group
2.2 z/OS security setup

The Siebel server, Siebel dedicated Web client, Siebel Mobile client, and the Siebel Tools client will use the DB2 Connect middleware to pass the user name and password used to access the Siebel eBusiness applications to DB2 UDB for z/OS and OS/390 to establish a connection. The user name and password can be user credentials supplied by the external security, if this option has been implemented.

All user names and passwords must have been preregistered with the OS/390 security package. The user name can be associated by the OS/390 security administrator with a group ID within the security package to simplify database security privilege administration.

To run the installation tasks and tests, RACF® user IDs and groups were set up with DB2 authorizations assigned to them. The authorizations, userids, and groups are defined in 2.2.1, “RACF - userids and groups” on page 25. User authorizations are managed by RACF, DB2, and Siebel security options.

Regarding client user authorizations, there can be many approaches to implementing them in a Siebel environment. Their granularity should be based on business requirements. The greater the granularity, the greater the complexity of the administration tasks.

The Siebel DB2 subsystem consists of many databases, table spaces and tables. It is difficult to know exactly which tables are used in each application. Thus, it is difficult to control access to the data by allowing users selective access at the table level.

There is a sample file /sieb704/dbsrvr/DB2390/gen_grantstat.sql, which gives an example of how to generate GRANTs to the interface (EIM) tables. Using this example, different types of grant statements with other criteria could be generated. Our approach was to let Siebel security functions control user access to the various applications. These steps are described in detail in this chapter.

Because of security constraints, RACF user IDs/passwords will periodically expire. Siebel eBusiness Applications support the ability to change passwords from the login screen once they expire without administrative intervention.

To be able to change the OS/390 passwords from the remote client machines, DB2 for OS/390 and z/OS must have the Extended Security option (extsec) set to YES. The default DB2 value is NO. This can be accomplished by setting the DSNZPARM EXTSEC=YES in the DSN6SYSP macro.

When LDAP or a similar security package is used, it is possible to have a single userid to log on to DB2. LDAP checks for the necessary userid authorization.
(password), and sends to DB2 a different userid that is common to the entire enterprise. In this case, EXTSEC should be set to NO as only the administrator should be able to change the password.

2.2.1 RACF - userids and groups

We created RACF group IDs SIEBEIM and SIEBGRP:

- The SIEBGRP group is the Siebel users group.
- The SIEBEIM group is the Siebel administrator group.

We created RACF user IDs SIEBTBO and SIEBADM:

- SIEBTBO is the Siebel table owner ID.
- SIEBADM is the Siebel administrator ID.
- Both SIEBTBO and SIEBADM user IDs were connected to the RACF group SIEBEIM.

We created RACF user IDs SIEBEL1 to SIEBEL5:

- SIEBEL1-5 are the Siebel user IDs.
- We connected SIEBEL1-5 to the RACF group SIEBGRP.

2.2.2 DB2 UDB for z/OS and OS/390 authorization

The installation of Siebel eBusiness Applications does not require the use of a SYSADM ID; however, the creation of some underlying DB2 resources will. Refer to the SIEBEL 7 document “Implementing Siebel eBusiness Applications on DB2 UDB for z/OS and OS/39” for specific details on roles and permissions required for the installation of the Siebel system.

We granted SIEBTBO and SIEBADM SYSADM authority at the ITSO, so we could execute all the required DB2 management tasks. The Siebel administrator “SIEBADM” ID only requires DBADM authority, but a DB2 system administrator will have to participate in the installation process.

The Siebel administrator does not require DBADM in the situation described in this installation, because the DDLs are being generated into files, and applied later on by the DBA.

The Siebel user IDs are all connected to the RACF group SIEBGRP. The DB2 authorizations for these user IDs is granted to the RACF group during the schema build process created by Siebel. The RACF group SIEBGRP was granted the following privileges: select, insert, update, delete.
The use of the DB2 group authorizations by Siebel requires that the DB2 secondary user authorization exit be enabled. For more information on this, refer to DB2 UDB for z/OS and OS/390 - Installation Guide for Version 7.

2.3 DB2 for OS/390 and z/OS setup

As mentioned, we were using a two-member data sharing group at the ITSO. The data sharing group name was D7RU, with members D7R1 and D7R2. These members were in separate Logical Partitions (LPARs) at the ITSO. The location name for the Data Distribution Facility (DDF) was DB7R, and was listening on port 33750. This information is required for DB2 Connect setup.

The DB2 secondary authorization exit was enabled on both systems.

2.3.1 DSNZPARM values

The following DSNZPARMs are required by Siebel when running the Siebel eBusiness Applications:

- **DSN6SPRM MACRO**
  - **CACHEDYN=YES** - turns on dynamic statement caching.
  - **CDSSRDEF=1** - turns off parallelism for dynamic statements.

- **MXTBJOIN=70** (or greater) - maximum number of table joins.
- **PARTKEYU=YES** - allows update of partitioning keys.
- **RETVLCFK=NO** - turns off index only access of varying length characters.

- **DSN6SYSP MACRO**
  - **EXTSEC=YES** - allows DB2 Connect to receive more complete error messages. Allows the changing of passwords via DB2 Connect.

A sample of DSNZPARMs used is in Appendix A, “DB2 parameters” on page 147. The values used are from Siebel recommendations and the work done at the IBM Silicon Valley Lab for Scalability Characterization of Siebel 7 on DB2 UDB for z/OS and OS/390. These values are recommendations and are installation-dependent. The following DSNZPARM values do require some comments:

- **CTHREAD**
  - Example from Appendix A: CTHREAD=70.
  - CTHREAD stands for the maximum number of non-distributed allied threads, that is, TSO, CICS®, IMS™, batch (including utilities), and CAF/RRSAF. Siebel uses distributed threads so it is not affected by this value.

- **CONDBAT**
  - Example from Appendix A: CONDBAT=25000.
  - Controls the number of connected clients.
Chapter 2. Installing the Siebel 7 system environment

- **MAXDBAT**
  - Example from Appendix A: MAXDBAT=500.
  - The accepted standard definition for the “active users” from a Siebel perspective is the number of logged-on users (to the Siebel application) and also connected (via DB2 Connect) to DB2.

As long as the Siebel user is viewing information within the application, this user is considered just connected to DB2. The maximum number of this kind of user to a DB2 subsystem is 150,000. There can be up to 32 DB2 instances in a data sharing configuration. So, the maximum active Siebel users could be 32 X 150,000.

However, if the Siebel user is actively doing processing that requires database accesses, then this connected DB2 user requires the use of a database access thread (DBAT). The theoretical limit to the maximum number of concurrent database active threads is 1999.

A DBAT utilizes a certain amount of DB2 database manager storage, which is acquired from the same storage area as that used for database manager buffers. To minimize the impact of concurrent DBATs on the use of this storage, the maximum number of concurrent DBATs should be limited to about 500.

- **EDMPOOL**
  - Example from Appendix A: EDMPOOL=40000.
  - Due to large Siebel DBDs, this needs to be large. The correct value really depends on the Siebel installation.

- **SRTPOOL**
  - Example from Appendix A: SRTPOOL=12000.
  - This field specifies the amount of storage needed for the sort pool. Siebel does many sorts.

- **POOLINAC**
  - Example from Appendix A: POOLINAC=120.
  - The approximate time, in seconds, that a database access thread (DBAT) can remain idle in the pool before it is terminated. Since maximum concurrency is desired for Siebel, a lower amount is recommended.

- **STARJOIN**
  - Example from Appendix A: STARJOIN=DISABLE.
  - In general, Siebel applications will not do well with star join because they do not have the suitable indexes on their tables to allow efficient star join processing. It is wise to use the system-wide DSNZPARM to disable star join for now.
2.3.2 Buffer pools and storage group

In this section, we describe buffer pool and storage group definitions.

**Buffer pool definitions**

The buffer pools used at the ITSO for the Siebel 7 installation are as follows, and were defined on both members:

- ALTER BUFFERPOOL (BP1) VPSIZE (20000) 'Used for 4 K table spaces
- ALTER BUFFERPOOL (BP2) VPSIZE (20000) 'Used for Indexes
- ALTER BUFFERPOOL (BP7) VPSIZE (60000) 'Used for Sort
- ALTER BUFFERPOOL (BP16K1) VPSIZE (12000) 'Used for 16 K table spaces
- ALTER BUFFERPOOL (BP32K1) VPSIZE (1000) 'Used for 32 K table spaces

The following commands were used to grant use of the bufferpools:

- GRANT USE OF BUFFERPOOL BP1 TO PUBLIC;
- GRANT USE OF BUFFERPOOL BP2 TO PUBLIC;
- GRANT USE OF BUFFERPOOL BP16K1 TO PUBLIC;
- GRANT USE OF BUFFERPOOL BP32K1 TO PUBLIC;

**Storage group definition**

The following describes the commands used to create the DB2 Storage Group.

- CREATE STOGROUP SIEBSTO VOLUMES (**) VCAT SIEB;
- GRANT USE OF STOGROUP SIEBSTO TO PUBLIC;

2.3.3 DB2 objects for Siebel

Siebel 7 Financial Services, which we installed at the ITSO, has the following Database Objects:

- 22 Databases containing Siebel tables
- 181 Segmented table spaces containing Siebel tables
- 2,485 Tables
- 13,244 Indexes
Due to the large number of database objects, we created 450 track Volume Table of Contents (VTOC) on each pack of the storage group. We also created a large VSAM user catalog to accommodate all the index data sets for Siebel.

The VCAT entry of “SIEB” becomes the high-level qualifier for the VSAM files. You may want to make sure this is an alias entry; otherwise, all of the many index VSAM files will be cataloged in the master catalog.

### 2.3.4 Stored procedures

Siebel utilizes the stored procedure DSNUTILS to run RUNSTATS. You need to insure the following:

- The DSNUTILS stored procedure was successfully created in DB2 installation job DSNTIJSG.
- The Workload Manager (WLM) is enabled to support the stored procedure. See Appendix B of *DB2 Utility Guide and Reference* for an example of how to customize a WLM proc for DSNUTILS.

The information for installation and enablement of the Stored Procedure is covered in the *DB2 UDB for z/OS and OS/390 V7 Installation Guide*.

### 2.4 Mid-tier setup

Our base platform consisted of: AIX 5L™, the April 2002 refresh (v5.1.0.25); IBM Developer kit for Java platform v1.3 and IBM C++ Runtime v5.0.2.2 on an RS/6000 S7a with 2 GB of RAM; three 18.2 GB; two 9.1 GB SCSI disks; a 1 Gb Ethernet adapter.

#### 2.4.1 AIX V5L configuration

For the purpose of clarity in this redbook, the following environmental variable was added to all user ids as they were created:

```bash
export PS1="\${LOGNAME})\${PWD}->"
```

This allowed all examples to display the userid and current path.

We used the following directory structure for our installation:

- `/xdb7r` The home directory for our DB2 Connect instance “xdb7r”
- `/sieb704/gtwysrvr` The installation directory for the Siebel gateway server
- `/sieb704/siebsrvr` The installation directory for the Siebel server
2.4.2 SAMBA install and setup

Samba is a UNIX Network File Systems (NFS) server for NetBios (Windows) clients. Samba is needed in order to allow Windows Siebel clients access to the Siebel file system (/siebfile).

Samba is not required if using Siebel 7 Web Clients. Siebel 7 has a new component (FSM), which takes care of file attachments.

Samba 2.2.3 is available in installp format from

http://www.bullfreeware.com

We copied samba-2.2.3.0.exe to the directory /work, unzipped the files into /work/temp, created an installp table of contents (inutoc), and started smit as shown in Figure 2-2.

![Figure 2-2 Preparing to install Samba](image)

Note the blank space and dot (.) following inutoc; this forces inutoc to create the table of contents in the current directory.

We proceeded through the install process by entering /work/temp as the Input device/directory of software, and selecting 2.2.3.0 SAMBA, SMB client, as the software to install.

The SMIT completion screen then showed freeware.samba.rte 2.2.3.0 installed.

This version and release of Samba did not install correctly. To correct the problem, we had to create the following directories:

- /usr/local/samba/lib
- /usr/local/samba/var/locks
We then had to move the information in /usr/local/lib to /usr/local/samba/lib. We also copied /usr/local/samba/lib/samba-2.2.3a/examples/smb.conf.default to /usr/local/samba/lib/smb.conf, and modified the “publicly accessible directory” stanza as follows:

- [siebfile]
- comment = Siebel File System
- path = /siebfile
- public = yes
- writable = no
- printable = no
- write list = @xsiebeim

**Note:** This group will not exist until DB2 Connect is installed.

To start the samba daemons (after the “xsiebeim” group has been created), we entered the following commands:

```
/usr/local/bin/nmbd -D
/usr/local/bin/smbd -D
```

Figure 2-3 shows Samba successfully installed.

![Successful installation of Samba](image)

**Figure 2-3  Successful installation of Samba**

### 2.4.3 HTTP Web server install and setup

IBM HTTP server V2 is available for Siebel installations from:

We installed the HTTP server by copying IBMHTTPserver.aix.EN.tar to /work, untarring it into /work/temp, and then executing installIHSaix.bin from an X11 session; see Figure 2-4.

(At the ITSO, we used Hummingbird®’s Exceed to serve X11 sessions on our terminals).

This started the Java installation GUI, through which we followed the prompts and accepted the defaults, to finish installing the IBMIHS HTTP server.

During the install, a file called httpd.conf is created in the /usr/IBMIHS/conf directory. This file will be updated during the Siebel SWSE installation.

Siebel also recommends that the worker.c stanza in the httpd.conf file be updated with the following information:

- # worker MPM
- # StartServers: initial number of server processes to start
- # MaxClients: maximum number of simultaneous client connections
- # MinSpareThreads: minimum number of worker threads which are kept spare
- # MaxSpareThreads: maximum number of worker threads which are kept spare
- # ThreadsPerChild: constant number of worker threads in each server process
- # MaxRequestsPerChild: maximum number of requests a server process serves
- # IfModule worker.c

Figure 2-4   Preparing to install IBMHTTP server
Chapter 2. Installing the Siebel 7 system environment

To start the IBMIHS HTTP server, we logged on as root and entered:

```
(root) -> /usr/IBMIHS/bin/apachectl start <enter>
```

**Note:** After installing the Siebel SWSE, we started IBMIHS by using:
```
/usr/IBMIHS/bin/startapa <enter>. (This file, and “stopapa”, will be created by the
Siebel Web Server installation process.)
```

### 2.4.4 DB2 Connect EE V7.2 installation

We mounted the DB2 Connect EE CD and executed /cdrom/db2setup to begin
the installation.

**Note:** Throughout the DB2 installation screens, use either the tab or the arrow
keys to move between fields, and use the spacebar to select/deselect options.

![Image of DB2 installation screen]

We pressed the spacebar to select **DB2 Administration Client**, then tabbed to
highlight [Customize] and pressed <enter>.
We tabbed to Control Center, pressed the spacebar to select it, and then tabbed until we highlighted [OK] and pressed <enter> to return to the Install DB2 V7 panel.
We tabbed to DB2 Enterprise Edition, used the spacebar to de-select it, and then tabbed to DB2 Connect Enterprise Edition and selected that option with the spacebar. We then tabbed to highlight [OK] and pressed <enter>.

![Figure 2-8 Create an instance](image)

When we used the spacebar to select Create a DB2 Instance, DB2setup automatically jumped to the Customize panel.

![Figure 2-9 Create DB2 instance (continued)](image)

In this panel we entered the instance name, the instance primary group name and instance password, and then tabbed to highlight [Properties] and pressed <enter>. 

Chapter 2. Installing the Siebel 7 system environment
In the DB2 Instance Properties panel, we tabbed to TCP/IP [Properties] and pressed <enter>.

**Note:** The calling port name is db2cinstancename.
Here we changed the TCP/IP services calling port name to match the name of our instance, and then tabbed to [OK] and pressed <enter>.

We selected [OK], accepting the defaults on the successive panels, until we returned to the Create DB2 Services panel.

![Figure 2-12 Select Create the Administration Server]

On this panel, we used the spacebar to select Create the Administration Server. The db2setup jumped directly to the customize Administration Server panel.

We accepted the defaults on all of the panels that followed, to complete the installation of DB2.

### 2.4.5 DB2 Connect Fix Pack 5s installation

As of the time of writing, DB2 Connect Fix Pack 5s was the current certified Fix Pack for Siebel installations. The latest Siebel certified DB2 Connect Fix Pack can be found at:


Before installing the Fix Pack, we stopped all db2 processes, as follows.

```
(root)/-> su - xdb7r <enter>
(xdb7r)/xdb7r-> db2 force applications all <enter>
(xdb7r)/xdb7r-> db2 terminate <enter>
(xdb7r)/xdb7r-> db2stop <enter>
(xdb7r)/xdb7r-> db2licd end <enter>
(xdb7r)/xdb7r-> exit <enter>
(root)/-> su - db2as <enter>
$ db2admin stop <enter>
$ exit <enter>
(root)/-> /usr/sbin/slibclean <enter>
```
To install the Fix Pack, we untarred the Fix Pack file into /work, which created a directory /work/special_5066.fp32/.

We issued the following command:

```
(root)/work-> smitty update_all <enter>
```

![Figure 2-13](image.png)

Figure 2-13  Input the Fix Pack directory

We entered the full path for the directory of the Fix Pack and pressed <enter> to continue.

![Figure 2-14](image.png)

Figure 2-14  Install Fix Pack 5s

We changed Preview only? to no and pressed <enter> on this panel and on the following panel to install the Fix Pack.

The next step was to update the db2 instance and the administrative instance, as shown:

```
(root)/-> cd /usr/lpp/db2_07_01/instance
(root)/usr/lpp/db2_07_01/instance-> db2iupdt xdb7r
(root)/usr/lpp/db2_07_01/instance-> dasiupdt db2as
```
Our final step was to make xdb7r the owner of the /xdb7r directory (this was necessary because we created the directory manually before installing DB2 Connect):

(root)-> chown xdb7r:xsiebeim /xdb7r <enter>

2.4.6 DB2 Connect configuration

There are two DB2 instance-level environment variables that must be set when running the Siebel application server on AIX (or Solaris):

(siebadm)/home/siebadm-> db2set DB2CONNECT_IN_APP_PROCESS=NO
(siebadm)/home/siebadm-> db2set DB2ENVIList=EXTSHM

Note that export EXTSHM=0 must be set for the DB2 Instance owner and each DB2 Connect user. By default, AIX does not allow applications to attach more than 11 shared memory segments per process.

Setting this variable specifies that the connection from the Siebel server to DB2 Connect EE will take place in the db2agent process that is part of DB2 Connect and its address space, rather than in the db2agent process of Siebel server.

There are several database manager configuration parameters, as follows:

(siebadm)/home/siebadm-> db2 update dbm cfg using DIR_CACHE YES
(siebadm)/home/siebadm-> db2 update dbm cfg using RQRIOBLK 32767
(siebadm)/home/siebadm-> db2 update dbm cfg using MON_HEAP_SZ 1024
(siebadm)/home/siebadm-> db2 update dbm cfg using DRDA_HEAP_SZ 512
(siebadm)/home/siebadm-> db2 update dbm cfg using MAXAGENTS 1000
(siebadm)/home/siebadm-> db2 update dbm cfg using NUM_POOLAGENTS 1000
(siebadm)/home/siebadm-> db2 update dbm cfg using MAX_LOGICAGENTS 3000
(siebadm)/home/siebadm-> db2 update dbm cfg using MAX_COORDAGENTS 1000

Important: We used those settings for the purpose of our project. Siebel recommends Siebel Database Connection Pooling to be used instead of DB2 Connect Connection Concentrator for production deployments (refer to 1.6, “Siebel 7 database connection pooling” on page 18).

MAX_LOGICAGENTS should equal MAX_COORDAGENTS when Connection Concentrator is not being used:

(siebadm)/home/siebadm->db2 update dbm cfg using MAX_LOGICAGENTS 1000

Catalog the database:

db2 CATALOG TCPIP NODE D7R1 REMOTE wtsc04oe.itso.ibm.com SERVER 33750 OSTYPE MVS
db2 CATALOG DATABASE DB7R AT NODE D7R1 AUTHENTICATION DCS

db2 CATALOG DCS DATABASE DB7R AS DB7R parms "',',,sysplex"

**Note:** wtsc04oe.itso.ibm.com was one of the members of our data sharing group listening on TCP/IP port 33750. DB7R was the location name for our DB2 subsystem.

We tested the connection:

(siebadm)/home/siebadm-> db2 connect to db7r user siebadm using siebadm

This produced the following (when the node and database are correctly cataloged):

Database Connection Information

- *Database server* = DB2 OS/390 7.1.1
- *SQL authorization ID* = SIEBADM
- *Local database alias* = DB7R

The installation and configuration of DB2 Connect EE with Fix Pack 5s was complete.

### 2.5 Installation of Siebel 7 servers

Siebel 7 servers include the following:

- Gateway server
- Application server
- Database server

#### 2.5.1 Installation of Siebel 7 Gateway server

We initiated the installation of the Siebel gateway server (from AIX Server Program CD #1 of 2) by creating user xsiebadm, in group xsiebeim, and assigning a temporary password to xsiebadm. Next we changed ownership of /siebfile to xsiebadm:

(root)/-> chown xsiebadm:xsiebeim /siebfile <enter>

We added the following line to the xsiebadm's.profile:

. /xdb7r/sqllib/db2profile

Next, we logged in as xsiebadm, changed xsiebadm's password to siebel, changed our umask to 027 for the install, and proceeded as follows:
We accepted the default language, then specified /sieb704/gtwysrvr as the installation directory, to continue with the installation of the gateway server.

![Image of the Siebel gateway server installation process]

**Figure 2-15  Beginning the Siebel gateway server installation**

We accepted the default gateway server port [2303] to continue.
Figure 2-16 Installing the gateway server

We accepted the next two defaults to continue; see Figure 2-17.

Figure 2-17 Installing the gateway server (continued)

We entered N at the next prompt, to bypass configuring autostart and complete the gateway server installation.
At the end of the gateway server installation, the gateway name server started, which is required for the installation of the Siebel server. We used the following commands to manually start the gateway server, when that was needed:

```
(xsiebadm)/-> cd /sieb704/gtwysrvr <enter>
(xsiebadm)/sieb704/gtwysrvr-> . ./siebenv.sh <enter>
(xsiebadm)/sieb704/gtwysrvr-> cd bin <enter>
(xsiebadm)/sieb704/gtwysrvr/bin-> ./start_ns <enter>
```

### 2.5.2 Installation of Siebel 7 Application server

To install the Siebel application server, as xsiebadm (with umask = 027), we first unset the environment variable SIEBEL_ROOT to avoid the possibility of installing the Siebel server in the same directory as the gateway server:

```
(xsiebadm)/sieb704-> unset SIEBEL_ROOT <enter>
```

Then we changed directory to the siebsrvr/install directory on the CD-ROM:

```
(xsiebadm)/sieb704-> cd /cdrom/unix_server_aix_1_enu/siebsrvr/install <enter>
```

We started the installation:

```
(xsiebadm)/cdrom/unix_server_aix_1_enu/siebsrvr/install->install_server<enter>
```

We accepted the default language and entered our intended siebsrvr directory.
**Note:** If you are not prompted to specify the directory into which Siebel should be installed, the environment variable SIEBEL_ROOT is defined and will determine the installation directory for the Siebel server. If you wish to change the directory at this point, you can use <ctrl><c> to get out of the install script, then change or unset SIEBEL_ROOT, and restart install_server.

![Beginning the Siebel application server installation](image)

We accepted the default address (erps7A) of the gateway server (which is not shown here). We also accepted the default gateway server port and N for Resonate to continue with the installation.
Figure 2-20  Siebel server installation (continued)

We accepted the default language (not shown) and entered N to the question “Would you like to accept these settings”. We then individually selected or rejected components, to enable the following at the Siebel server startup; see Figure 2-21 on page 46.
We entered a non-default name for our development siebel server (siebdev) and at the next prompt, we entered the path of our Siebel file system.
We accepted the default to “apply settings” and for the description of the enterprise, then entered 3 to select IBM DB2 UDB for z/OS and OS/390. At the following prompts, we entered the name of our database, then the table owner (siebtbo), SQLID (group name - siebgrp), and our administrator name (we used siebadm instead of sadmin).
Next, we entered siebadm's password (twice), accepted the default Synchronization Manager port, the default for the FirstLogic Directory, and then pressed <enter> at the Chart Server prompt.

Then we entered siebadm's password, accepted the default Synchronization Manager port, the default for the FirstLogic Directory, and pressed <enter> at the Chart Server prompt.
Chapter 2. Installing the Siebel 7 system environment

At the next prompt we entered 2 to verify that we were not using a Chart Server host, then accepted the default Chart Image format and Encryption type.

We pressed <enter> to accept the parameters; see Figure 2-26 on page 50.
We accepted the default for our new Siebel server (erps7a) and entered a description; at the next prompt, we entered the home directory of our DB2 instance.

We entered 3 for eBriefings, and N to select not to configure SynchMgr; see Figure 2-28 on page 51.
We entered $N$ to select not to configure autostart.

We accepted $Y$ to apply these settings, then accepted $Y$ again to start the server and complete the installation; see Figure 2-30 on page 52.
Next, we updated /sieb704/siebsrvr/siebenv.sh by adding the following lines:

```bash
export LDR_CNTRL=LOADPUBLIC@MAXDATA=0x50000000
export MW_GMA_VADDR=0xb0000000
export AIXTHREAD_SCOPE=S
export MALLOCMULTIHEAP=considersize,heaps:4
# rule: heaps:n, where n is the number of cpus
export SIEBEL_ASSERT_MODE=0
export NUMBER_OF_USERS=1000
export SIEBEL_OSD_NLATCH=$(7 * $NUMBER_OF_USERS + 1000) | bc'
# rule: SIEBEL_OSD_NLATCH = 7*(Max Tasks) +1000
export SIEBEL_OSD_LATCH=$(1.2 * $NUMBER_OF_USERS)/1 | bc'
# rule: SIEBEL_OSDLATCH = 1.2*(Max Tasks)
```

We added the following line to /sieb704/siebsrvr/db2cli.ini:

```
DISABLEUNICODE=1
```

**Note:** The preceding parameter must be added to the db2cli.ini file of any dedicated clients.

The final step was to update /sieb704/siebsrvr/bin/siebmtshw by adding the following lines:

```bash
export AIXTHREAD_SCOPE=S
export AIXTHREAD_MUTEX_DEBUG=OFF
export AIXTHREAD_RWLOCK_DEBUG=OFF
export AIXTHREAD_COND_DEBUG=OFF
export MW_GMA_VADDR=0xb0000000
```
export MALLOCMULTIHEAP=considersize,heaps:4
# rule: heaps:n , where n is the number of cpus

The steps to start the Siebel server are:
(xsiebadm)/-> cd /sieb704/siebsrvr <enter>
(xsiebadm)/sieb704/siebsrvr-> .:/siebenv.sh <enter>
(xsiebadm)/sieb704/siebsrvr-> cd bin <enter>
(xsiebadm)/sieb704/siebsrvr/bin-> ./start_server <enter>

This concluded our Siebel server installation.

2.5.3 Installation of Siebel 7 database scripts

To install the Siebel database server scripts, as xsiebadm (with umask = 027), we first unset the environment variable SIEBEL_ROOT to avoid the possibility of installing the Siebel database server scripts in the same directory as the gateway server or Siebel server:
(xsiebadm)/sieb704-> unset SIEBEL_ROOT <enter>

Then we changed directory to the dbsrvr/install directory on the UNIX CD-ROM #2:
(xsiebadm)/sieb704-> cd /cdrom/unix_server_aix_2_enu/dbsrvr/install <enter>

We started the installation:
(xsiebadm)/cdrom/unix_server_aix_2_enu/dbsrvr/install->./install_dbsrvr <enter>

We accepted the default language; refer to Figure 2-31.

Figure 2-31  Beginning the Siebel database server installation
At the next prompt, we entered the installation directory (/sieb704/dbsrvr).

We entered N to change the default settings, and responded N to DB2 for Unix and Windows. We responded Y to DB2 for OS/390, and N to Oracle, and Y to Sample File Attachments. We then responded Y, to accept these settings.
We entered Y again, to accept the default language, and then accepted the value for SIEBEL_ROOT and ENU.

![Siebel Environment Variables](image)

**Figure 2-34 Siebel database server installation (continued)**

### 2.5.4 Installation of Siebel 7 maintenance - 7.0.4.200

The installation process for Siebel maintenance is the same for each component.

- We mounted the Siebel AIX Server Programs (CD 1 of 5), navigated to the appropriate installation directory, and executed the following install scripts:
  ```
  export SIEB_OVERRIDE_COMPATIBILITY_CHECK=Y <enter>
  cd /cdrom/unixserver_aix_1 <enter>
  ./unix_server_aix_1_enu/gtwysrvr/install/install_gateway -F
  ./unix_server_aix_1_enu/siebsrvr/install/install_server -F
  ./unix_server_aix_2_enu/dbsrvr/install/install_dbsrvr -F
  ```

  We followed the scripts, using the same parameters as the initial installation of the components.

### 2.5.5 Installation of Siebel 7 database schema

To begin the installation of the database schema, we navigated to the siebsrvr directory (/sieb704/siebsrvr) and sourced siebenv.sh to set our environment:

```
(xsiebadm)/sieb704/siebsrvr-> ./.siebenv.sh <enter>
```

We changed the directory to the bin directory, and executed dbsrvr_config.ksh to start the Siebel Schema installation.

```
(xsiebadm)/sieb704/siebsrvr-> cd bin<enter>
(xsiebadm)/sieb704/siebsrvr/bin->dbsrvr_config.ksh<enter>
```

We entered Y to accept SIEBEL_ROOT and Language; see Figure 2-35 on page 56.
We pressed Enter at the next screen to accept SIEBEL_ROOT.

Next, we entered the Siebel database root directory; see Figure 2-37.

On the next twenty-one prompts (which are not shown here), we selected/entered:

- 2 - IBM DB2 UDB for z/OS and OS/390
- 1 - Install Database
- 1 - Install Siebel Database
1 - English
1 - “siebsrvr_siebdev” for the ODBC Data Source Name
1 - “SIEBADM” for the Database User Name
1 - “SIEBADM” for the Database Password
1 - “SIEBTBO” for the Siebel Schema Qualifier
1 - “SIEBGRP” for the Group Authorization ID

1 - Standard Install
1 - Generate DDL into a File
<enter> to accept the default DDL Output Directory
1 - Multiple Tables per Table Space schema layout
1 - “SIEBSTO” to specify our table space storage group name
1 - “SIEBTBO” again, to use the same storage group for indexes
<enter> to accept BP1 for the 4 k buffer pool
<enter> to accept BP16k for the 16 k buffer pool
<enter> to accept BP32k1 for the 32 k buffer pool
<enter> to accept BP2 for the Index buffer pool
1 - “SIEB” to specify the Database Name Prefix
1 to specify ASCII

Finally, we entered Y to start the Siebel Upgrade Wizard (srvrupgwiz), as shown in Figure 2-38; this is used in AIX to start the Siebel Upgrade Wizard. The Upgrade Wizard generates all DDL changes in the Siebel application, and is used for the Siebel install, upgrade, and database utilities.

Next, we stopped to build the database schema on the zSeries (as shown in Figure 2-39 on page 58).
Figure 2-39  Ready to create the schema

We transferred the file sieb704/dbsrvr/db2390/dboutput/install/schema.sql using FTP with the following commands:

quote site cylinders primary=10 secondary=5
quote site recfm=fb lrecl=80 blksz=0
put schema.sql

We used this file in a DB2 batch job and created the Siebel schema.

Refer to Appendix B, “Sample Siebel 7 generate DDL output” on page 157, to review the sample DDL that we used to create the schema on our system.

After creating the schema, we restarted the Siebel Upgrade Wizard (srvrupgwiz):
(xsiebadm)/sieb704/siebsrvr/bin-> srvrupgwiz /m master_install_mf.ucf <enter>
The restart created the Siebel file
/sieb704/dbsrvr/db2390/dbouput/install/ddlview.sql. This was FTPed to the
OS/390 and included in a DB2 batch job.

The Siebel Upgrade Wizard (srvrupgwiz) was restarted a second time, which
then inserted the Siebel seed data into the new OS/390 DB2 Siebel schema.

After the seed data was installed, we followed these steps:
1. We executed the following:
   
   (xsiebadm)/sieb704/siebsrvr/bin->./dbsrvr_config.ksh<enter>

2. We accepted the SIEBEL_ROOT and Language values on the first screen.
3. We accepted the Siebel server root directory on the second screen.
4. On the third screen, we entered /sieb704/dbsrvr for the database server root
directory, and entered 2 for IBM DB2 UDB for z/OS and OS/390.
5. On the next screen, we entered 3 to select Import/Export Repository.
On the next screen, we entered 1 to select Import Repository; refer to Figure 2-42.

On the next screen, we entered 1 to select Import Standard Siebel Repository; refer to Figure 2-43.
On the next eight prompts (which are not shown here), we selected/entered:

- 1 - English
- “siebsrvr_siebdev” for the ODBC Data Source Name
- “SIEBADM” for the Database User Name
- “SIEBADM” for the Database Password
- “SIEBTBO” for the Siebel Schema Qualifier
- “SIEBGRP” for the Group Authorization ID
- Pressed Enter for default Repository Name
- Pressed Enter to accept the default repository file name

We then entered Y to accept the values for the Import Repository and run srvrupgwiz; see Figure 2-44.

The srvrupgwiz wizard then installed the Siebel Repository.

### 2.5.6 Post-installation tasks

The following tasks were performed at the end of the Repository Import.

- Grant Table privileges
  - Edit and run the grantstat.sql script.
    
    We edited the file /sieb704/dbsrvr/db2390/grantstat.sql by replacing &1 with the Siebel Admin Group ID “SIEBEIM”.

---

**Figure 2-44  Siebel database schema installation (continued)**

---
We then did a DB2 Connect to the OS/390 database as the Siebel table owner “SIEBTBO” (which has SYSADM authority), and ran the job:

(xsiebadm)/sieb704/dbsrvr/db2390-> db2 connect to db7r user siebtbo using siebadm <enter>
(xsiebadm)/sieb704/dbsrvr/db2390-> db2 -tvf grantstat.sql <enter>

- Load Statistics.

We edited the file /sieb704/dbsrvr/db2390/loadstat.sql by replacing “SIEBELTABLECREATOR” with “SIEBTBO”. We then did a DB2 connect to the OS/390 database as the Siebel Table Owner “SIEBTBO” (which has SYSADM authority), and ran the job.

(xsiebadm)/sieb704/dbsrvr/db2390->db2 connect to db7r user siebtbo using siebtbo<enter>
(xsiebadm)/sieb704/dbsrvr/db2390->db2 -tvf grantstat.sql <enter>

- Validate the Siebel Database.

We executed dbchck to verify the physical schema with the Repository.

(xsiebadm)/sieb704/siebsrvr/bin-> ./dbchck /u SIEBADM /p SIEBADM /t SIEBTBO /r "Siebel Repository" /l dbchck.log /dict /all /s siebsrvr_siebdev <enter>

We checked the dbchck.log file for errors at the end.

- Populate the Siebel File System.

We copied the all the files except KB.kb from /sieb704/dbsrvr/files to /siebfile/att.

We then copied /sieb704/dbsrvr/files/KB.kb to /siebfile/cms.

2.6 Installation of Siebel 7 Web server and extensions

We made xsiebadm the owner of the IBMIHS instance.

(root)/usr-> chown -R xsiebadm:xsiebeim IBMIHS <enter>

We changed directory to the eappweb/install directory on the first server CD-ROM:

(xsiebadm)/sieb704-> cd /cdrom/unix_server_aix_1_enu/eappweb/install <enter>

We then started the installation:

(xsiebadm)/sieb704-> unset SIEBEL_ROOT <enter>
(xsiebadm)/cdrom/unix_server_aix_1_enu/eappweb/install->install_eappweb<enter>
At the following prompts, we accepted the default language, the default Web server directory, “N” for Resonate, “eps7a” for our Gateway Server, the default Gateway Server port of 2320, and “siebdev” for our enterprise name.

We accepted erps7a for our Siebel server name, accepted 1 for compression type, accepted 1 for encryption type, accepted 1 for logging level, accepted 80 for the Web Server HTTP Port, and accepted 443 for the Web Server HTTPS Port; see Figure 2-46 on page 64.
We then entered the Web Update Protection Key, the First Internet Address of the Web Server, the Alternate Internet Address of the Web Server, the anonymous login userid, the password (twice), and the anonymous login contact name and password (twice); see Figure 2-47.

We then entered Y to accept ALL the settings, and Y for ENU.
We chose not to restart the IBMiHS Web Server, and the Siebel Web Server and Extension install completed; see Figure 2-48.

Figure 2-48  Siebel Web server and extension installation (continued)
2.6.1 Customizing the Siebel 7 Web server

We completed a few final customization steps for the Siebel Web server and IBMIHS, as follows:

- We gave all users rights to the eappweb directory:
  
  (xsiebadm)/~> chmod -R 777 /sieb704/eappweb <enter>

- We added the following environment variables to /usr/IBMIHS/bin/startapa:

  ```
  export RT_GQ=ON
  export MW_GMA_VADDR=0xb0000000
  export YIELDLOOPTIME=1
  export SPINLOOPTIME=1000
  export SIEBEL_ASSERT_MODE=0
  export MALLOCTYPE=buckets
  export MALLOCMULTIHEAP=considersize,heaps:4
  # rule: heaps:n, where n is the number of cpus
  ```

2.7 Installation of Siebel 7 Web server maintenance - 7.0.4.200

The installation process for Siebel maintenance is the same for the Web component.

- We mounted the Siebel AIX Server Programs (CD 1 of 5), navigated to the appropriate installation directory, and executed the following install script:

  ```
  export SIEB_OVERRIDE_COMPATIBILITY_CHECK=Y <enter>
  cd /cdrom/unixserver_aix_1 <enter>
  ./unix_server_aix_1_enu/eappweb/install/install_eappweb -F
  ```

  We followed the script, using the same parameters as the initial installation of the Web component.
Chapter 3. Modifying the Siebel schema for a production environment

In this chapter, we describe the process we used to create a partitioned table space in the DB2 Siebel production database. Our production database was created on DB2 for z/OS using the Siebel-provided, one table per table space, non-partitioned, storage control file. The production Siebel enterprise was installed on a Windows 2000 Server (TOT85.itso.ibm.com). We used table “S_ACT_EMP” to illustrate the steps involved in partitioning a table.

This chapter includes the following sections:

- Extracting a new storage control file from our existing database and modifying the storage control file to add partitioning to a table
- Creating a backup table and table space
- Using the Siebel Upgrade Wizard to produce the new DDL
- Creating the partitioned table space
- Creating a new Siebel storage control file reflecting the new Siebel schema
3.1 Extracting a current storage control file

We extracted a storage control file (prod_storage.ctl) reflecting the current production database schema. We started the Configure DB Server:

Start > Programs > Siebel Enterprise Server 7.0.4 > Configure DB Server

On the next 16 screens:
1. We entered the Gateway server address T0T85 and our “production” Enterprise name: SIEBPROD.
2. We accepted the default for the Siebel Server directory.
3. We accepted the default for the DB Server directory.
4. We selected Run Database Utilities.
5. We selected Configure Database.
6. We selected Extract Storage Control File.
7. We accepted SiebSrvr_siebprod as our ODBC Data Source Name.
8. We entered: PRODADM as the database user name and password.
9. We entered: PRODTBO as the Siebel Schema Qualifier.
10. We entered: PRODGRP as our Group Authorization ID.
11. We selected Extract from Catalog.
12. We entered a name for the new storage control file:
   C:\sea704\dbsrvr\db2390\prod_storage.ctl.
13. We reviewed all of the variables and clicked Finish.
14. We clicked OK to “Apply the configuration now” and create a new ddl file.
15. We clicked OK to begin the upgrade.
16. We clicked OK when the Siebel Upgrade Wizard completed.
3.2 Modifying the storage control file - table space partitioning

The Database Storage Configuration utility is shipped in the form of a Microsoft Excel macro. It is used to prepare and validate a storage control file for use by the Siebel database configuration utilities to generate the appropriate DDL to build the Siebel schema.

The utility is dbconf.xls in the database server\DB2390 directory. Siebel provides a number of standard storage control files in this directory.

It is possible to update a storage control file at any time to indicate that a table that was previously non-partitioned is to become a partitioned table. If a current storage control file already exists, then this can be amended using the Database Storage Configuration Utility to reflect the new definitions. If no current storage control file exists, then one can be extracted from the existing database.

The Siebel utilities will not carry out the DB2 work necessary to change a table from non-partitioned to partitioned; this will need to be done by the DBA.

The steps that we took to change S_ACT_EMP from a non-partitioned table to a partitioned table are detailed in the following sections.

3.2.1 Running the Siebel Database Configuration utility

We started the Siebel Database Storage Configurator Excel spreadsheet (dbconf.xls), enabling macros (see Figure 3-1 on page 70) and importing our control file prod_storage.ctl.
In the Table tab, we executed a find on $S_{\text{ACT}}_\text{EMP}$ and noted that it was in table space $H0393000$. For security reasons we decided to put the modified table into a new table space, which we called $S6001000$. By executing a find on $S6001000$, we determined that there were no other tables in table space $S6001000$.

With $S_{\text{ACT}}_\text{EMP}$ highlighted, we clicked **Partition Table**, entered 10 in the partitions pop-up window, then clicked **OK** to the pop-up message: Creating 10 partitions for tablespace ‘$S6001000$’, please review partitioning index...

We entered $S_{\text{ACT}}_\text{EMP}_\text{P99}$ in the next pop-up box for the name of the partitioning index, clicked **OK** to the pop-up message: Creating 10 index partitions for ‘$S_{\text{ACT}}_\text{EMP}_\text{P99}$, enter the limitKey values in this page., then clicked **OK** to the next pop-up message.

Next, we entered the following columns for the partitioning index, using the standard Siebel partitioning key specifications from the sample partitioning control file (storage_p_group.ctl):

```
Name       = $S_{\text{ACT}}_\text{EMP}_\text{P99}$
Table      = $S_{\text{ACT}}_\text{EMP}$
Partitions = 10
SpecialCol = PARTITION_COLUMN WCHAR(2) NOTNULL DEFAULT '  ' Function   = WHEN (N.EMP_ID IS NOT NULL) BEGIN ATOMIC SET N.PARTITION_COLUMN = RIGHT(N.EMP_ID, 2); END Column 1   = PARTITION_COLUMN ASC
Column 2   = EMP_ID ASC
```

Figure 3-1  The Database Storage Configurator

If you would like to configure a database storage control file, please use “Import” button to import it into this spreadsheet.

If you would like to save your configuration from this spreadsheet as a storage control file, please use “Export” button.

If you wish to overwrite the data in this spreadsheet with information from the storage control file, first click “Close” button to remove the current data, and then import your new storage control file.
We clicked **Show Index Parts** (Figure 3-2) and clicked **OK** to the message: Creating 9 partitions for 'S_EVT_ACT_P99', enter the LimitKey values in this page. We entered the Limit Keys (Figure 3-3).

**Note:** As shown in Figure 3-3, there are two single quotes preceding the highlighted LimitKey. These quotes **must** be used in order to set this value.
(however, they do not appear until you actually select the field). When working in this field, do not change or delete these two single quotes.

We then clicked the Tablespace tab and altered table space S6001000 by changing PriQty to 48, SecQty to 1440 and clearing the SegSize cell. We finished by clicking the Show TSPARTs tab and clicking OK to the pop-up message: Creating 9 partitions for tablespace ‘S6001000’, please review partitioning index...:

To create the new control file, we returned to the Home tab, clicked the Export tab, entered a name, and clicked Save. The Excel macro validated and exported our new control file.

3.2.2 Creating a backup table

We ran a batch job to create a backup table space and table, populated the backup table, and dropped the original table space and table. The job we ran is listed in Appendix C.1, “Batch job used to create a backup table space” on page 168.

The INSERT method we used here is suitable for small tables only. With the INSERT method, each inserted row is logged and this can fill up the log data set very quickly. Locking of million of rows can also be very expensive. Furthermore, in case of failure, rollback could be a major issue for long-running INSERTs with no commits to break down the unit of work. This is why the INSERT method is not recommended for large tables.

For large tables, we recommend the method described in Chapter 6, “Partitioning method using DB2 utilities” on page 97.

3.2.3 Synchronize schema definition - produce a new schema file

We followed this path to get started:

Start -> Programs -> Siebel Enterprise Server 7.0.4 -> Configure DB Server

We entered our Win2000 gateway server name T0T85 and our enterprise server name SIEBPR0D.
On the next 14 screens:

1. We accepted the default for the Siebel Server directory.
2. We accepted the default for the Siebel Database Server directory.
3. We selected **BM DB2 UDB for z/OS and OS/390**.
4. We selected **Run Database Utilities**.
5. We selected **Synchronize Schema Definition**.
6. We accepted SiebSrvr_siebprod as our ODBC Data Source Name.
7. We entered: PRODADM as the database user name and password.
8. We entered: PRODTBO as the Siebel Schema Qualifier.
9. We entered: PRODGRP as our Group Authorization ID.
10. We selected **Generate DDL into Files**.
11. We accepted the default output directory and entered our modified storage control file: \sea704\dbsrvr\db2390\prod_p_storage.ctl.
12. We accepted the name of our current repository Siebel Repository.
13. We reviewed all variables and clicked **Finish**.
14. We clicked **OK** to “Apply the configuration now” and create a new ddl file.
3.2.4 Applying the schema changes

We extracted, from the new schema.sql file, the ddl necessary to create the new partitioned table space, table, indexes, and trigger (sometimes required to populate the partitioning column).

We FTPed this DDL file to the OS/390 and included it in a batch job which created the new objects, granted privileges on the new table, populated the table, and dropped the temporary objects. This batch job is in Appendix C.2, “Batch job used to create a partitioned table space” on page 171.

**Note:** The batch job was run using the forward slash (/) as the delimiter. This was done because the create trigger statement contains a semi-colon (;) in the syntax.

3.2.5 Saving the storage control file

Our last step is to save the storage control file created for production, so it would be easy to track later. The Excel spreadsheet can be saved with a different name (for example, the same name as the storage control file prod_p_stolage.xls). This spreadsheet can be used later for reference or modification.
Chapter 4. Migration from development to the production system

In this chapter, we describe the process used to migrate the work of the development team to the new production system.

To illustrate development work, we used our Siebel 7 Tools client to add a column X_TEST to table S_EVT_ACT in our development system.

Note: We built our original production Siebel server on the Windows server TOT85. For logistical reasons, we switched to a new Win2000 Siebel server T21a with a Siebel installation path of E:\sea704\ for this migration chapter.

This chapter includes the following sections:

- Running dev2prod to migrate a copy of the development repository to our production system and create a new schema.sql file with schema updates for production
- Reviewing and editing the new schema.sql file and applying the changes
- Activating the new repository
4.1 Running dev2prod

Migrating the development respiratory to the production system is facilitated by the Siebel supplied utility dev2prod. This utility can be run from the Configure DB Server GUI:

Start -> Programs -> Siebel Enterprise Server 7.0.4 -> Configure DB Server

We entered our Gateway Server (note the new server name T21a) and Enterprise Server names; see Figure 4-1.

![Siebel Enterprise Parameters: Gateway Server Address](image)

Figure 4-1 Starting Configure DB Server

We accepted the default for the Siebel Server directory; see Figure 4-2 on page 77.
Chapter 4. Migration from development to the production system

We accepted the default for the Siebel Database Server directory; see Figure 4-3.

We selected IBM DB2 UDB for z/OS and OS/390, as shown in Figure 4-4 on page 78.
We selected Migrate Repository, as shown in Figure 4-5.

We next entered the database name of our development system (cataloged as db7r when we built this new Siebel server, T21a); see Figure 4-5 on page 78.
Figure 4-6  Database name of our development system

We entered the Siebel administrator name and password from the development system; see Figure 4-7.

Figure 4-7  Siebel administrator on db7r

We entered the Siebel Schema Qualifier of the development system; see Figure 4-8 on page 80.
We entered a name to be used in the production system for the imported copy of the development repository (Dev Siebel Repository); see Figure 4-9.

Figure 4-9  The source and target repositories

We indicated the target database platform; see Figure 4-10 on page 81.
Chapter 4. Migration from development to the production system

Figure 4-10 Target DB platform

We entered the target database ODBC name; see Figure 4-11.

Figure 4-11 Production database ODBC name

Then we entered the Siebel administrator name and password on the production system; see Figure 4-12 on page 82.
We entered the Production system schema qualifier and group authorization ID; see Figure 4-13.

We selected the option to generate a file with the DDL, which we will apply in a batch job; see Figure 4-14 on page 83.
In Chapter 3, “Modifying the Siebel schema for a production environment” on page 67, in which we describe how to partition a table in our new production database, we extracted a current storage control file (TOT85\c:\sea704\dbsrvr\db2390\new_storage.ctl).

We now copied that file to our new Siebel server (T21a) and specified it as the storage control file to be used in the dev2prod process; see Figure 4-15.

Figure 4-14  Option to generate a file with the DDL

Figure 4-15  Specify the DDL output directory and the storage control file
We reviewed the parameters and clicked **Finish**; see Figure 4-16.

![Figure 4-16](image1)

**Review parameters**

We clicked **OK** to confirm; see Figure 4-17.

![Figure 4-17](image2)

**Apply the configuration**

84  Siebel 7 with DB2 for z/OS: Database Implementation and Administration Guide
4.2 Reviewing the schema file - applying the DDL to the production system

The new schema.sql file (E:\sea704\dbsrvr\db2390\dboutput\dev2prod\schema.sql) contains extraneous DDL statements (we installed Siebel 7.0.4.200). We edited our schema.sql file to contain only the required ALTER statements.

At this point, the DBA has the choice (as in previous chapters) of executing the DDL from the Siebel server or transferring the DDL (via ftp) to the zSeries for execution by the DBA in a batch job. In our case, there was only one statement:

```sql
ALTER TABLE PRODTBO.S_EVT_ACT ADD X_TEST VARCHAR(26)
```

4.3 Activating the repository imported from development

To apply the development system modifications to the production system, we used Siebel Tools, logging on to production, in order to switch to the repository we imported from development.

**Start -> Programs -> Siebel Tools 7.0.4 -> Siebel Tools**

By default, Repository is not listed in the Object Explorer list. To add Repository to your Object Explorer, follow this path:

- View -> Options -> Object explorer -> Repository
- Click the right arrow to add, then click OK.

To rename the current repository to a backup name, we selected Repository in the left pane, as shown in Figure 4-18 on page 86.
We clicked the name of the current repository and changed the name to Original Siebel Repository; see Figure 4-19.
Next, we selected **Dev Siebel Repository** and renamed it to **Siebel Repository**; see Figure 4-20.

![Figure 4-20 Renaming Dev Siebel Repository](image)

We exited Siebel Tools. Our development repository had been migrated to our production system.
Chapter 5. Identifying and removing unused, non-unique indexes

In this chapter, we describe a procedure for dealing with DB2 indexes in Siebel. The elimination of unused, non-unique indexes benefits performance for inserts, deletes, and updates, particularly in a data sharing environment.

The discussion involves two topics:
- Identifying indexes within Siebel applications that can be eliminated
- Removing indexes physically from the Siebel schema, and logically from the Siebel repository

**Important:** It is mandatory that you seek approval for dropping indexes from Siebel prior to proceeding. Removing some indexes from the Siebel schema will impact product performance.

The Siebel support policy for inactivating and dropping unused indexes is rigidly followed. This support policy can be found in a Siebel technical services paper called *Dropping Indexes Process*. You should ask your Siebel Technical Account Manager for guidance.
5.1 Why remove indexes

Siebel uses many indexes to insure good performance regardless of the configuration that the customer deploys for their Siebel environment. For example, insurance companies have different processing patterns than banks, and individual banks will have unique processing patterns. The Siebel application comes prepackaged with indexes for all potential, generic configurations.

Are all of these indexes used at any one customer site? Probably not. So, once the Siebel implementation is configured, it is very likely that some of the indexes can be eliminated.

5.2 Which indexes to remove

All unique indexes, such as P1 and U1 indexes, are not candidates to be inactivated or dropped. Dropping any unique index can result in data integrity conflicts that cannot be detected during processing.

Indexes for the Siebel Repository and Siebel logging tables, such as S_DOCK_TXN_LOG, are not valid candidates to be dropped either.

Only unused, non-unique secondary indexes can be dropped. This group includes indexes on unused Foreign Keys that only store the Null value entry. These indexes are further identified by a cardinality of 1, which indicates these indexes are not being effectively used in the specific deployment.

5.3 Siebel functional areas of interest

There are two parts to any Siebel application. The first is the load of Siebel base tables using a process called EIM, and the second is Siebel online. In general, it is assumed that reviewing EIM indexes would take place as an installation activity; Siebel online would take place during a post-implementation phase.

The procedure for each is very similar, but the relevant indexes are very different. It is very likely that the indexes we remove to optimize EIM initial load performance are critical to the Siebel online application. Each part of Siebel needs to be treated as a distinct application in order to gain maximum performance benefits from dropping unused, non-unique indexes.

- The initial EIM load is normally a one-time activity. A substantial quantity of data is loaded from legacy systems. It is during initial load that we find the most potential benefit from dropping indexes.
Testing has shown that incremental loads done as part of the daily, weekly, or monthly cycle can normally handle indexes that are needed for online processing.

- The online Siebel application is obviously the purpose of having the software. Online performance is critical.

If you follow the processes outlined in the following sections, the indexes that remain are those that enforce table uniqueness and provide optimal access path for EIM incremental loads and Siebel online processing.

5.4 Identifying unused, non-unique indexes

The procedure for identifying indexes for removal is the same regardless of the application (EIM or Siebel online).

At IBM, we use the DB2 Performance Monitor for OS/390 product (DB2 PM) to help perform this task. Potentially, non-IBM monitors can be used as well, although we have not verified their capabilities. The procedure discussed uses DB2 PM.

Information regarding the setup and configuration of DB2 PM can be found in 12.5, “Using the performance database of DB2 PM” in the IBM Redbook *Squeezing the most out of dynamic SQL with DB2 for z/OS and OS/390*, SG24-6418.

5.4.1 In the EIM load process

The EIM process uses data extracted from legacy data stores to load into a set of staging tables, called EIM tables. The EIM process extracts the data from the EIM tables, validates the data according to internal Siebel rules, and loads it into the Siebel base tables.

Once the sample data is loaded into the EIM tables, follow the procedure described in 5.4.3, “Index identification procedure” on page 92. The load for this exercise is a proportional subset of the typical EIM load that represents all the tables to be loaded. We recommend that 5% provides a reasonable sample. Once the exercise is complete, insure you have the DDL for the indexes you have identified, and then drop them.

Complete the initial load from the EIM tables to the Siebel base tables, then recreate the dropped indexes. The indexes must be recreated, because the EIM portion of the Siebel application has different characteristics and access requirements than the online portion of the application. Siebel online will be adversely impacted if these indexes are not present.
5.4.2 In Siebel online

Once the application is configured and running for a while, online requests will be fairly standardized. Create a list of transactions that your configuration uses frequently and which should be fine-tuned.

Perform the procedure described in 5.4.3, “Index identification procedure” on page 92. The load for this exercise is your list of transactions. Once the exercise is complete, insure you have the DDL for the indexes you have identified and then drop them.

Rerun your transaction list to insure that there is no performance degradation. If there is, double-check that you have only dropped indexes from the list.

When new functionality is deployed and performance degradation occurs, use normal SQL analysis techniques in combination with the list of dropped indexes to determine if replacing an index will help the transaction.

5.4.3 Index identification procedure

1. Stop the DB2 subsystem and then restart it; this flushes the dynamic cache. (Note that you can also run RUNSTATS to flush the dynamic cache).
2. Insure that the GTF data set is available and sufficiently large to contain the results.
3. Turn on the performance trace:

   START TRA(P) CLASS(30) IFCID(22,63) DEST(GTF)

4. Apply load to the DB2 database (as described individually for EIM and Siebel online in the preceding section).
5. Stop the performance trace:

   STOP TRA(P) CLASS(30)

Once the trace has been completed, use DB2 PM to obtain the results.

Example 5-1 on page 93 shows a sample DB2 PM record trace job to format DB2 data and store it in sequential data sets suitable for use by the DB2 load utility.

The record is loaded into DB2 tables, and you produce reports using a reporting facility such as Query Management Facility (QMF™).

---

1 We use GTF as the target for our trace data, but SMF can also be used. It means using SMF in the DEST parameter of the TRACE command rather than GTF. It also means using the GTF data set as the input file for the Performance Monitor report.
Chapter 5. Identifying and removing unused, non-unique indexes

Example 5-1  Sample record trace job for IFCID 22 and 63

```
//PMREPORT JOB (),CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X,
//      REGION=OM
//REPORT EXEC PGM=DB2PM
//STEPLIB DD DSN=DSN710.DGO.SDGOLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//JOBSUMDD DD DUMMY
//RTFILDD1 DD DSN=SIEBEL1.DB2PM.IFCD2263.DG.FILE,
//           DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(1,10),RLSE),UNIT=SYSALLDA
//INPUTDD DD DISP=SHR,DSN=SIEBEL1.DB2PM.IFCD2263.DG
//SYSIN DD *
RECTRACE
    FILE
       INCLUDE(IFCID(22,63))
    EXEC
```

Several sample table create statements and load control statements come with DB2 PM in the SDGOSAMP library. Use the create table statement members DGONCFMB and DGONCFSQ initially to set up the environment and create tables as a one-time activity. The load control statement members DGONLFMB and DGONLFSQ use a LOAD REPLACE utility to load the DB2 tables.

Once the tables are loaded, run the query shown Example 5-2. The query output contains a DROP INDEX statement line for each non-unique index that is not being used in the application being tested.

Example 5-2  Query to obtain a DROP INDEX list for the unused non-unique indexes

```
SET CURRENT SQLID='SIEBTBO' ;
SELECT ' DROP INDEX SIEBTBO.'||A.NAME||' ;'
    FROM SYSIBM.SYSINDEXES A, SYSIBM.SYSINDEXPART B
    WHERE NOT EXISTS (  
        SELECT ACCESSNAME
        FROM DB2PM.DB2PMFRTRC_MINIPLN
        WHERE ACCESSNAME=A.NAME
        AND ACCESSCREATOR=A.TBCREATOR)
    AND UNIQUERULE='D'
    AND A.TBCREATOR='SIEBTBO'
    AND B.PARTITION=0
    AND A.NAME=B.IXNAME
    AND A.CREATOR=B.IXCREATOR ;
```

However, there is probably no need to drop and recreate later all indexes on Siebel tables that are not used during EIM and used, for example, during online
Siebel. This is a waste of time and resource. So, when you know all the tables used during a specific application, you can append the lines in Example 5-3 to the query in Example 5-2, along with the proper table list.

Example 5-3  Lines to append to query in Example 5-2 to reduce list of indexes to drop

```sql
AND A.TBNAME IN (
    'EIM_ACCOUNT' ,
    'EIM_ACTIVITY' ,
    ...................,  
    ...................,  
    'S_PARTY' ,
    'S_USER' ) ;
```

Important: Ensure you have the create statements for all these indexes in an easily accessed file.

- For EIM, you must recreate all the indexes following the load.
- For Siebel online, the indexes are likely to be required later when new transactions are adopted.

When you are satisfied that everything is as required, drop the indexes in your list.

### 5.5 Inactivation of unused indexes

Before doing any development work for zSeries, such as inactivating indexes, you must check the “Developing for deployment on DB2 for zSeries” checkbox found by selecting View -> Options -> Database in Siebel Tools (Figure 5-1 on page 95).
Once you have identified and dropped indexes, you need to inactivate them in the Siebel Repository. Inactivating an index in the Siebel Repository prevents that index from being recreated when changes are migrated from one environment to another.

Here we show you how to inactivate the EIM_AGREEMENT_T05 index. (We picked this index randomly simply to describe the steps.)

With this box selected, we carried out the following procedure to inactivate index EIM_AGREEMENT_T05. We selected the table EIM_AGREEMENT and locked the project (Tools -> Lock Project) so that we could inactivate the index. Next, we checked the Inactive column in the EIM_AGREEMENT_T05 row.

Figure 5-2 on page 96 shows the inactivated index in red just after we moved the cursor on the next row.
Now the index is dropped from the DB2 schema and inactivated from the Siebel repository. The Siebel repository and the DB2 database layout are in sync.
Partitioning method using DB2 utilities

In Siebel 7, customers can choose to install the Siebel-DB2 database with one, or many, tables per table space, and with partitioned, or non-partitioned, table spaces. When tables grow large enough, the need to partition table spaces arises.

In this chapter, we present a table space partitioning method using DB2 utilities with REXX programming to automate the process. This method is suitable for partitioning large-volume tables. The REXX procedures build the appropriate JCL streams and control cards to automate the execution of the process.

The REXX procedures, jobs, and DB2 data definition language (DDL) required to prepare and run the partitioning process are described in this chapter, and the code samples are provided in Appendix D. You can also download them from the ITSO Web site; see Appendix F, “Additional material” on page 211 for downloading instructions.
6.1 Planning steps for partitioning Siebel tables

Partitioning a DB2 table is not a complex process, but it does require some thought and planning. The following is a checklist of the planning steps:

1. Identify the tables to be partitioned and determine the partitioning columns.
2. Plan for REXX procedures to automate the table partitioning process.
3. Unload the tables you want to partition:
   a. Create and populate a reference table which contains information on the tables to partition. This reference table is needed for the REXX procedure.
   b. Use a REXX procedure to generate the JCL for unloading the tables.
   c. Execute the generated JCL to unload the tables.
4. Create the new tables with partitions:
   – Drop the old tables.
   – Create the new partitioned table spaces.
   – Create the new tables in their partitioned table spaces.
   – Create indexes and triggers (if needed).
   – Grant authorizations.
5. Load the new tables:
   a. Use a REXX procedure to generate the JCL for reloading the new tables.
   b. Execute the generated JCL to re-load the new partitioned tables.
6. Reorg the table spaces (if needed).

6.2 Identifying candidate tables and partition columns

You need to identify the candidate tables for partitioning and determine the partitioning column you want to use to partition these tables. Partitioning strategy is explained in Chapter 3 of *Implementing Siebel eBusiness Applications on DB2 UDB for OS/390 and z/OS, Version 7.0.4*.

You want a partition column that contains a good spread of values, so that data is distributed evenly. The partitioning solution can use an additional column and a trigger on any table where there is no index that can be used as a candidate for a partition index. In general, this is accomplished by using a combination of user keys, a partial user key, and Siebel row IDs.

However, because the row IDs are generated automatically and, most of the time, in sequential order, these columns are not good candidates for partitioning. The partition column and a trigger can be used to generate a more random
number by extracting the last 2 bytes of a row ID column and storing this in the partition column.

This number should provide a reasonable distribution. Since partitioning and clustering go together, in general, the partition index structure should be similar to the original clustering index (that is, if the original clustering index contains more than one column, these additional columns should also be added to the partitioning index in order to ensure clustering order).

**Important:** Some tables may not have default partition columns. There may be situations when the DBA knows from data analysis that better partitioning columns exist than those recommended by Siebel. This situation would need to be reviewed by the Siebel Technical Account Manager.

### 6.3 Planning for REXX procedures

Plan for REXX procedures if you want to automate the process of table partitioning and simplify your task by partitioning many tables in one pass. The REXX procedures offer the possibilities to unload data from the old non-partitioned tables, generate the partition column value during the unload process, and reload data, including the partition column value, into the new partitioned tables. The REXX procedures are particularly helpful for doing initial partitioning of tables in which the partitioning column will be subsequently populated using a DB2 trigger function.

Prepare your environment as follows:

- Verify that DSNREXX is installed and bound in your environment; otherwise, run DSNTIJRX from the SDSNSAMP library (this will install DSNREXX).
- The core REXX procedure itself is generic and does not require customization. However, the members which call the procedure and the members called by the procedure are environment-specific and need to be customized. You need to customize DB2 and ISPF libraries to suit your installation and standards. The following members have to be customized:

  ```
  ISPBAT  ISPF Proc
  TBLUNLD  DSNTIAUL unload proc
  LOADJOBC  JCL job card
  LOADSKEL  Load skeleton member
  UNLDJ  Unload job generator
  LOADGENJ  Load job generator
  ```
6.4 Unloading the tables

You can use a REXX procedure to unload data from the non-partitioned tables. The unload REXX procedure requires a reference table to be built with the list of all tables that will be partitioned.

6.4.1 Reference table

The REXX procedure to unload the tables is driven by a reference table with the following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBNAME</td>
<td>Name of tables to be partitioned.</td>
</tr>
<tr>
<td>PART_COL</td>
<td>Partitioning column. Some tables may require an additional partitioning column; otherwise, leave blank.</td>
</tr>
<tr>
<td>PART_SRC</td>
<td>The source column to populate the partitioning column.</td>
</tr>
<tr>
<td>COL_FUNC</td>
<td>The column function which will be used to generate data for the PART_COL partitioning column.</td>
</tr>
<tr>
<td>CARD</td>
<td>Number of rows to be unloaded; should be set to 0.</td>
</tr>
<tr>
<td>IDX_CNT</td>
<td>Number of indexes for this table; should be set to 0.</td>
</tr>
</tbody>
</table>

An example of how to create this table is shown in Example: D-1, “REFTBDDL member” on page 177.

Populate the reference table by loading a list of tables to be converted to partition by using SYSPUNCH card member TBL00001, as illustrated in Example 6-1.

**Example 6-1 Reference table - list of tables to partition**

<table>
<thead>
<tr>
<th>TBNAME</th>
<th>PART_COL</th>
<th>PART_SRC</th>
<th>COL_FUNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_EVT_ACT</td>
<td>PARTITION_COLUMN OPTY_ID</td>
<td>CHAR(RIGHT(OPTY_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_ACT_EMP</td>
<td>PARTITION_COLUMN EMP_ID</td>
<td>CHAR(RIGHT(EMP_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_ADDR_ORG</td>
<td>PARTITION_COLUMN OU_ID</td>
<td>CHAR(RIGHT(OU_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_OPTY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_ADDR_PER</td>
<td>PARTITION_COLUMN PER_ID</td>
<td>CHAR(RIGHT(PER_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_APPLD_CVRG</td>
<td>PARTITION_COLUMN ASSET_ID</td>
<td>CHAR(RIGHT(ASSET_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_ASSET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_ASSET_CON</td>
<td>PARTITION_COLUMN ASSET_ID</td>
<td>CHAR(RIGHT(ASSET_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_ASSET_POSTN</td>
<td>PARTITION_COLUMN ASSET_ID</td>
<td>CHAR(RIGHT(ASSET_ID,2),2)</td>
<td></td>
</tr>
<tr>
<td>S_CONTACT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_COMMUNICATION</td>
<td>PARTITION_COLUMN ROW_ID</td>
<td>CHAR(RIGHT(ROW_ID,2),2)</td>
<td></td>
</tr>
</tbody>
</table>

Tables that do not require an additional partitioning column value to be provided have only the TBNAME column populated in the reference table (such as the S_OPTY and S_CONTACT tables shown in Example 6-1). These tables are downloaded as a whole.
In this case, a regular DSNTIAUL (parm DB2) unload of the table is generated by the REXX procedure (refer to Example: D-8, “DSNTIAUX member” on page 193).

The tables which require an additional partitioning column value to be provided have the PART Кол, PART_SRC, and COL_FUNC columns populated in the reference table. Those tables are unloaded by specifying each column name.

The REXX procedure, in this case, generates an SQL SELECT statement which builds the value to populate the partitioning column during the unload process. The unload REXX procedure builds the SELECT statement using the SQL function provided in the COL_FUNC column of the reference table. (This SQL function is the same as the one which will be used by the DB2 Trigger to populate the partitioning column later in the everyday usage of the table.) In this case, a DSNTIAUL (parm SQL) unload of the table is generated by the REXX procedure (refer to Example: D-7, “DSNTIAUI member” on page 192).

The CHAR function in the COL_FUNC column ensures that the output is a character field with a fixed length of 2 bytes (the same attribute we need to populate the PARTITION_COLUMN). Without the CHAR function, the RIGHT function returns a varying length field, that is, VARCHAR.

### 6.4.2 Unloading the data

The unload process uses the DB2 DSNTIAUL utility to unload the tables.

We have an unload job UNLDJ (refer to Example: D-3, “UNLDJ member” on page 178) which drives the unload process. When UNLDJ executes, it calls ISPBA T, which calls the REXX procedure UNLDGEN.

UNLDGEN generates unload jobs based on the list of tables supplied in the reference table. The process is as follows:

- UNLDJ executes ISPBA T (see Example: D-4, “ISPBA T member” on page 179),
- ISPBA T generates a batch ISPF environment. It executes the UNLDGEN REXX procedure (see Example: D-6, “UNLDGEN member” on page 181), which runs in ISPF batch mode.
- The UNLDGEN REXX procedure performs the following:
  - It reads a SYSIN parm card (refer to the last section in Example: D-3, “UNLDJ member” on page 178) to get the following input parameters required to generate the unload jobs:

```
SSID        DB2 subsystem ID
TABLEOWNER  Table owner
```

Chapter 6. Partitioning method using DB2 utilities 101
BUILD number used as the output PDS second level qualifier
OUT_HLQ Output PDS high level qualifier
TABLEINP Fully qualified DB2 reference table name
BASEPDS An existing model PDS to allocate output
DELOADCRDS Delete existing output PDS, if any

- It allocates the following SYSPUNCH PDS and SYSREC output files which will contain the unload jobs:
  HLQ.build#.UNLDDATA.LOADCRDS A common PDS file
  HLQ.build#.UNLDDATA.TBL# One sequential file for each table

- It reads the reference table to get the list of the tables to unload.
- It generates the unload jobs which will use DSNTIAUL to unload the tables (refer to Example: D-7, “DSNTIAUI member” on page 192, Example: D-8, “DSNTIAUX member” on page 193, and Example: D-9, “TBLUNLD member” on page 193).
- It generates an index cross-reference ($README) in the output PDS. The index cross-reference contains a reference number and a table name, which is useful information to use for debugging purposes.
- It submits, automatically, the unload jobs.

The UNLDGEN REXX procedure submits these jobs via the internal reader. Results of the unload jobs go into the SYSPUNCH PDS and SYSREC output files.

### 6.5 Creating new tables with partitions

To create new tables with partitions, you need to do the following:

- Drop the old tables
- Create the new partitioned table spaces
- Create the new tables in their partitioned table spaces
- Create indexes and triggers (if needed)
- Grant authorizations

**Creating the new tables**

Use the Siebel DDLIMP tool to generate the DDL which will create the partitioned table spaces, tables with partitioning columns, indexes and triggers. Review and update the generated DDL with correct space allocation, and add appropriate Grants.

Drop all tables to be converted to partition. If the original tables were installed using one table per table space, drop their table space as well.
Execute the generated and customized DDL (from ddlimp) to create the partitioned objects.

### 6.6 Loading the tables

The load process uses the DB2 LOAD utility to load the tables.

We have a LOADGENJ job (refer to Example: D-11, “LOADGENJ member” on page 194) which drives the table load process. When LOADGENJ executes, it calls the REXX procedure LOADGEN, which generates a load job for each table. The process is as follows:

- LOADGENJ executes the LOADGEN REXX procedure (see Example: D-12, “LOADGEN member” on page 195).
- LOADGEN performs the following:
  - It reads a SYSIN parm card (refer to the last section in Example D-11 on page 194) to get the following input parameters required to generate the load jobs:
    - **SSID**: Source subsystem ID
    - **TABLEOWNER**: Table owner
    - **BUILD**: BUILD number
    - **LOADPDS**: Template PDS generated from UNLDJ
    - **DELOUTPDS**: Delete loadcard PDS if exist flag
    - **MACRO_NAME**: Name of edit macro to edit load card
    - **TABLEINP**: DB2 reference table name that contains a list of tables already unloaded
  - It allocates a new PDS file which will contain the load jobs
    - HLQ.build#.UNLDDATA.ssid
  - It copies all load card members from the previous unload PDS file (LOADCRDS)
  - It modifies load card members to add appropriate SORTKEYS values, LOG NO, RUNSTATS and NOCOPYPEND options (see Example: D-14, “CARDEDIT member” on page 204).
  - It generates one load job for each table. All generated load jobs are put in member JCL2LOAD.

Review the JCL2LOAD member carefully, and submit it when you are satisfied with the results.
6.7 Reorganizing the tables

After all jobs have completed successfully, review the clustering index statistic for each table to verify that the cluster ratio is low. If not, reorg tables accordingly.
Chapter 7. Monitoring dynamic SQL

In this chapter, we discuss how to use the DB2 Performance Monitor for OS/390 (DB2 PM) Workstation Online Monitor and Siebel SQL traces to identify and monitor poorly performing dynamic SQL statements in Siebel.

A comprehensive description of dynamic SQL analysis and usage of the DB2 Performance Monitor for OS/390 (DB2 PM) Workstation Online Monitor tool can be found in the following IBM Redbooks:

- *Squeezing the Most out of Dynamic SQL with DB2 for z/OS and OS/390*, SG24-6418
- *DB2 for z/OS and OS/390 Tools for Performance Management*, SG24-6508
7.1 Monitoring dynamic SQL in the DSC

The Siebel application uses dynamic SQL for all data retrieval. The dynamic statement cache (DSC), where the SQL are cached, is the best place to monitor ongoing SQL activity. Siebel requires you have the DSC turned on, and this is done by setting the CACHEDYN ZPARM parameter to YES.

DB2 Instrumentation Facility Component Identifiers (IFCID) 316, 317, 318 have been specifically implemented to monitor the DSC. To obtain DSC information, an IFI program has to request IFCIDs 316, 317 and 318. These IFCIDs are available when DB2 monitor trace class 1 is active. We discuss these in more detail in “IFCID 316, IFCID 317, and IFCID 318” on page 108.

7.1.1 DB2 PM Workstation Online Monitor tool

The DB2 PM Workstation Online Monitor tool provides access to the Dynamic Statement Cache to analyze the performance of cached dynamic SQL statements. It reads the previously mentioned IFCIDs with the help of a host server task (the data collector), which collects DB2 performance data through the instrumentation facility interface (IFI).

For data-sharing installation, the data collector has to be started for each DB2 member to monitor, because each member has its own DSC. The DB2 PM Workstation Online Monitor client GUI presents this data to you in a user-friendly format.

Figure 7-1 on page 107 illustrates how the DB2 PM Workstation Online Monitor provides a detailed look into the dynamic statement cache.
When you first open this window, it may be blank because a filter is activated to limit the number of rows displayed. You can disable filtering, or modify the filter to an appropriate value, to get information about statements in the cache. Click the check box to enable data to be transferred from the DSC.

**Note:** The information is only downloaded from the host (data collector) when you request it, so ensure that the check box is checked. To update the information, use the Refresh button on the upper toolbar.
Detailed information about the statement cache is only available to DB2 PM Workstation Online Monitor functions. The information cannot be externalized to SMF or GTF and processed by a DB2 PM batch job.

**IFCID 316, IFCID 317, and IFCID 318**
As mentioned, in order to obtain cache information, the IFI program has to request IFCIDs 316, 317, and 318, and these IFCIDs are available when DB2 monitor trace class 1 is active.

**IFCID 316**
This contains a large number of fields. It contains the first 60 bytes of SQL and by default only gathers basic status information, such as when the statement was inserted in the cache and the number of time it was reused. Other statistics are filled when triggered by IFCID 318. If accounting class 3 is active, six suspension timers are also collected.

**IFCID 317**
This can be used in addition to IFCID 316 to obtain the full SQL statement text.

**IFCID 318**
This acts as a switch for IFCID 316 to collect all available information. It does not itself contain any data; it simply indicates to fill in all available fields in IFCID 316 including the statistic fields. However, if you want to collect the six suspension timers (wait times), you must also have DB2 accounting class 3 active.

You can start IFCID 318 any time you wish. From that moment, DB2 starts collecting DSC statistics for each statement that is already in the cache, or for each new statement inserted into the cache.

Each time you press the refresh button, you get the statement statistics since the last time IFCID 318 was started. As soon as you stop IFCID 318, the statistic counters are not collected anymore, and their values are set to zero.

Statements that are not executed for a while can be pushed out of the cache, at which time their statistics are lost. Statements that are executed frequently can stay in the cache for weeks.

This means that if IFCID 318 is always turned on, you do not know the exact starting point for statement statistics—and without it, it is difficult to compare the impact of different statements. Therefore, it is helpful to stop and restart IFCID 318 periodically when doing DSC analysis (for example, each morning), and then view the statistics during the day, to examine the SQL run during that day.

If you do not start IFCID 318 at all, you will still be able to retrieve the content of the DSC, but as the counters will be zeros, you will not be able to sort the SQL
statements in order to identify the poor performers. In this case, you will not be able to identify the inefficient SQLs described in the next section.

**Note:** If you are CPU-conservative, you can choose to start IFCID 318 only when you encounter a problem, then perform your analysis, and then stop it.

You can also sort the content of the DSC on any column you want by simply clicking it. Once you’ve found a statement of a particular interest, then double-click on its row to zoom in on it (for more detail, refer to 7.1.3, “Identifying inefficient SQL” on page 112).

Also keep in mind that most of the numbers in the SQL statement window are updated at the *end* of the SQL execution—except for cursor counters, which are incremented at the end of open cursor and fetch cursor operations. (In this case, you do not have to wait until the cursor closes before the numbers get incremented.)

But in general, the numbers here are not as accurate as the numbers you might get, for example, in DB2 accounting, where the counter is updated as the event takes place. Therefore, if a query scans a lot of rows before a qualifying row is returned between fetch operations, the counters do not get updated for a long time, so using the statement cache information to monitor long-running queries should be used with this information in mind.

### 7.1.2 Explaining SQL statement from the cache

Figure 7-2 on page 110 shows a specific SQL statement we selected, as well as all the information stored in the DSC that relates to this statement.
You get the complete SQL statement text, and if you have Visual Explain installed on your workstation, you simply have to click the Explain button to get the explanation of this statement. A Visual Explain window will give you the access path graph, as shown on Figure 7-3 on page 111.
Be aware that Visual Explain re-prepares the statement using the actual catalog information to present this information. The explained access path is not necessarily the same as the cache statement used. Visual Explain uses catalog table information, like sysibm.sysindexes available at explain time; they will be different when indexes have been created or dropped.

The real access path that is used resides in the cache and can be different from the Explain as discussed. You must have a trace active to capture IFCID 22 (the mini-plan) at the time the statement is inserted into the cache to verify the access path used by DB2 when the SQL statement was added.

IFCID 63 is also used for this purpose, but the large SQL statements built by Siebel are frequently too large and get truncated above 5 K. (This 5 K limitation may be dropped in DB2 for z/OS V8.)
7.1.3 Identifying inefficient SQL

The key indicators of inefficient SQL are:

- High getpages and low rows processed
- High rows examined and low rows processed
- Long statement average elapsed time
- User complaints

When searching the DSC for inefficient SQL, it is helpful to sort the statement entries by total getpages, total rows processed, or total elapsed time, and then use these indicators to find individual problem statements. The sort presents the high impact statements on the top of the list.

High getpages and low rows processed (per execution)

A getpage occurs when DB2 loads a table or index page to check the contents of the page. Examining numerous pages uses additional CPU, and contributes to pressure on the buffer pools. High “getpages per row processed” indicators are seen when:

- Predicates contain columns that are not indexed, or not in the index used to access the table.
- SQL contains range predicates - these will sometimes cause index columns to the right of the range predicate to not be indexable.
- Index screening, where there are gaps in matching index columns from the predicate columns.

High rows examined and low rows processed (per execution)

A row is examined when DB2 checks it physically to determine if it satisfies the predicate criteria. When a row can be disqualified based on index access, it does not count as a “row examined”. When DB2 reads rows rather than indexes, then “rows examined per row processed” is high. This commonly happens when:

- Predicates contain columns which are not indexed, or not in the index used to access the table
- SQL contains range predicates - these will sometimes cause index columns to the right of the range predicate to not be indexable

After finding a statement with high “rows examined per row processed”, you should also check the average number of rows examined and rows processed, to confirm that the statement is inefficient.

Long “average elapsed time” per execution

There are a number of reasons why one execution of a statement may take a long time. In the case where the statement fetches, inserts, or changes hundreds
or thousands of rows, it is normal. Check “Avg rows processed” to see the number of rows returned per execution. In the situation where only few rows are processed on each execution, or the time per row processed is long, it could point to one of several things:

- I/O constraint on the disk where the table or index resides
- Logical row lock contention (because of INSERT,UPDATE,DELETE, and so on)
- Inefficient SQL, as described above
- DB2 contention on page latches
- CPU-constrained or storage-constrained machine

### 7.1.4 Considerations on parameter markers

Siebel uses dynamic SQL. Statements are first prepared with parameter markers, which are placeholders for the execution-time parameters. Since parameter markers are used for prepare, DB2 cannot make use of all the optimizer capabilities. For example, the column distribution statistics gathered with RUNSTATS FREQVAL (the TYPE=F statistics in SYSIBM.SYSCOLDIST), which track the values that most often occur in a column, are not used when optimizing dynamic SQL prepared with parameter markers.

When the TYPE=F statistics are not used, it can cause situations where DB2 does not choose an index that would be the most efficient way to access the data. Generally, these are situations where the SQL has a predicate referencing an index with a low cardinality column (for example, a column that is a flag for processing status, which might have only a few possible values).

When the statement is optimized with parameter markers, DB2 does not know whether the statement is searching for the small percentage of unprocessed rows, or the large percentage of processed rows, and it chooses another way to access the data, rather than the low cardinality index. If the statement is re-optimized at execution, then DB2 will see the value of the predicate variable, and be able to compare it against the column distribution data—and can then choose the best access path by determining whether the value occurs frequently or rarely in the column.

In most cases the values of host variables are not known at prepare time. Consequently, the optimizer cannot take advantage of columns values distribution. This can result in severe access path problems.
7.2 Siebel SQL tracing

In this section, we describe the options available to the Siebel and/or database administrator to identify the SQL statements issued by Siebel to assist with any performance investigation:

- Ways of identifying the SQL issued by the Siebel user interface
- EIM performance hints and tips

7.2.1 Object Manager

To set tracing at the Object Manager level, navigate to View -> Site Map -> Server Administration -> Components -> Component Event Configuration and select the required Object Manager. Since we were using the Siebel Call Center, we selected the Call Center Object Manager, shown in Figure 7-4 on page 115.
Figure 7-4  Set Object Manager SQL Log event = 4

This same level of tracing can be set by starting the server manager from the command line. This is done by issuing the following command from the siebserver/bin directory:

```
srvrmgr /g erps7a /e siebdev /s erps7a /u SIEBADM /p SIEBADM
```

where:

- `/g` - Gateway server
- `/e` - Enterprise server
- `/s` - Siebel server
- `/u` - Userid
- `/p` - Password
We ran the command `list component` to determine the name of the Call Center Object Manager and found that this was SCCObjMgr. We then ran the following command to switch SQL tracing on for this Object Manager:

```
change evtloglvl ObjMgrSqlLog=4 for comp SCCObjMgr
```

![Figure 7-5](image)

Setting this level of tracing will cause details of the SQL statements issued by the Siebel Object Manager to be written to a log file called SCCObjMgr_nnnnn.log. In our environment, these log files were written to a directory called `/sieb704/siebsrvr/enterprises/siebdev/erps7a/log`.

These files can be reviewed, to identify particular SQL statements to investigate further, by using tools such as the DB2 explain plan feature. Tracing can be turned off by resetting the value back to 1 (which is the default).

When setting the SQL tracing at the Object Manager level, individual log files will be produced for all users who connect to this Object Manager, as shown in Figure 7-6.

![Figure 7-6](image)

The SQL detailed in the Object Manager trace logs shown in Figure 7-7 can be used as input to a DB2 explain plan process, but not directly; they have to be "processed" first.

In the following sections, we describe two methods for doing explain, the first one directly on the host, and the second one using the DB2 command center on AIX.
Doing explain on the host

The Object Manager logs may have lines that are greater than the 72-character limit of SPUFI or DSNTAP2. In addition, information other than SQL statements are present, that will generate explain errors if used directly.

We used a REXX program to filter out unneeded information and to cope with the 72-character limit. The REXX also adds “DELETE FROM PLAN_TABLE WHERE QUERYNO = xxx”, “DELETE FROM DSN_STATEMNT_TABLE WHERE QUERYNO = xxx” and an “EXPLAIN ALL SET QUERYNO = xxx FOR” for each SQL to explain. Therefore, the REXX output file can be directly used as input for explain.

Figure 7-7 Extract from an Object Manager trace log
Use the following process to explain the SQL statements:

1. Upload the Object Manager trace log to the host. We used FTP to transfer it to a variable blocked (VB) sequential file, with a record length of 1024. We used ASCII parameter for the transfer. We called it SIEBEL.EIM.LOG.

2. Edit the SIEBEL.EIM.LOG host file, and use:\n   ```
   c x'05' ' ' all
   ```
   to replace incorrect 05 hexadecimal characters with space characters.

3. Run the REXX program. We used a job that calls the REXX program providing the SIEBEL.EIM.LOG file as input. The REXX output file name was SIEBEL.EIM.OUT.

4. Review the SIEBEL.EIM.OUT file. ODBC Functions like { ts '2002-10-15 12:21:06' } need to be replaced with TIMESTAMP('2002-10-15-12.21.06') in order for explain to execute correctly.

5. Run explain using the SIEBEL.EIM.OUT file through SPUFI or DSNTEP2.

From that point, use regular plan table analysis to identify poor access paths.

Note: The REXX program and a sample job to call it can be found in Appendix E, “REXX program for EXPLAIN” on page 205.

Doing explain with the DB2 command center on AIX

In order to run an explain plan, we first needed to create the plan tables on the DB7R instance.

We changed the default DB2 Command Center settings to set the semi-colon (;) as the termination character by selecting Tools -> Tools Settings and checking the box Use statement termination character.

We then copied one the SQL statements from the Object Manager trace log and pasted it into the DB2 Command Center script window. We added the text explain plan set queryno = 666 for before the SQL statement, as shown Figure 7-8 on page 119, and then clicked execute to run the explain plan.
We retrieved the detailed explain plan information from the DB2 plan table using the following SQL statement:

```sql
select * from siebadm.plan_table where queryno = 666;
```

### 7.2.2 Dedicated Web client

It is also possible to install a dedicated Web client and request that the SQL be written to a log file. To do this, you need to set the `/s <filename>` parameter on the properties of the dedicated Web client. The SQL log file produced will be found in the `client\bin` directory, if the full path details are not provided with the filename.

### 7.2.3 EIM performance considerations

This section highlights how to identify any SQL statements from an EIM task that may need to be reviewed as part of a performance review.
We issued the following commands from the Server Manager command line interface, to ensure that EIM produced a log file containing the details of the SQL processed:

```
change evtloglvl SQL=4 for comp EIM
change evtloglvl SQLSummary=4 for comp EIM
change evtloglvl Trace=3 for comp EIM
```

We then ran EIM to import data into S_ASSET using the SQLPROFILE parameter on the IFB file, as shown in Example 7-1.

**Example 7-1 Sample IFB file using the SQLPROFILE parameter**

```
[Siebel Interface Manager]
PROCESS = Import
    UPDATE STATISTICS = FALSE
SQLPROFILE = sql.log

[Import]
    TYPE = IMPORT
    BATCH = 1
    TABLE = EIM_ASSET
```

Once EIM had completed, we were able to review the file topsql.log, produced by EIM, which showed us the longest-running SQL statements; refer to Figure 7-9.

**Figure 7-9 Extract from log file produced by the SQLPROFILE parameter**

The SQL statements identified can also be obtained from the log file, if needed.
Chapter 8. Monitoring the Siebel 7 database

In this chapter, we describe the database monitoring and administration tasks. We discuss the following:

- Installation considerations
- Database objects
- Monitoring using DB2 Statistics
- Database reorganization
- Backup and recovery
# 8.1 DB2 installation considerations

We recommend that you create a separate DB2 subsystem for Siebel. The reason for this is because point-in-time recovery scenarios, discussed later in this chapter, are difficult to manage and implement in a DB2 subsystem shared with other applications.

Siebel installations with DB2 for z/OS and OS/390 require some PTFs to be installed. A list of recommended PTFs for OS/390 Siebel installations may be found in the following IBM informational APARs: APAR ii13012 for Siebel 7.0.4 and APAR ii13487 for Siebel 7.5.x. Make sure you have those PTFs installed.

When you install Siebel 7 in a DB2 for z/OS and OS/390 V7 data sharing environment, you may want to consider the observations and recommendations on the following DB2 issues:
- Database objects
- Free space
- EDM pool
- Buffer pools

## 8.1.1 Database objects

The Siebel 7 installation process (for Multiple Tables per Table Space schema layout) produces several Siebel-DB2 objects; the number varies, depending on the Siebel implementation. The following numbers are approximate.

- **22 DB2 databases** are defined. All databases have the same creator - SIEBTBO. The databases are named as follows:
  - SIEBDB00 through SIEBDB21 (22 databases)

  The database SIEBDB15 alone contains 54 table spaces. The other databases contain 1 to 18 table spaces each.

- **181 table spaces** are defined. All table spaces have the same creator - SIEBTBO. 160 of those table spaces have tables defined in them. 22 table spaces have no tables defined in them.

- **There are 2485 tables and 3 views** defined. All of the tables and views have the same creator - SIEBTBO.

  The table spaces are defined with page sizes of 4 K, 16 K, or 32 K.

  83% of the tables have 4 K pages. The remaining tables have 16 K or 32 K pages. The larger page size is due to the fact that approximately 60% of the columns defined in the tables are Varchar or Long Varchar.
There are 5 table spaces with 103 to 282 tables defined in them. There are 33 table spaces with 20 to 70 tables defined in them. This can have major implications on the use of the REORG utility discussed later in this book.

Given the number of tables and the challenge of managing the data sets containing those tables (a DB2 table space maps to a VSAM data set), the Siebel 7 installation option of multiple tables per table space represents, in general, a practical strategy.

* There are 13,244 indexes defined on the 2,485 tables. All of the indexes have the same creator - SIEBTBO.

There is an average of 5 indexes per table in Siebel 7. 25 tables have between 30 and 70 indexes. The high number of indexes can stress performance during data load processes, and during REORG.

However, Siebel 7 applications contain a large amount of text, which suggests a preponderance of single-row processing (multiple-row processing tends to involve data summarization, which does not apply in general to text applications). Consequently, the large number of indexes appears to support single-row SQL processing, which is a characteristic of text applications.

There is another characteristic of Siebel 7 indexes that requires planning. The indexes frequently contain Varchar data columns. DB2 pads Varchar index columns to the maximum size of the column. This means that the DASD space requirements for the indexes could be significant. Varchar field length reduction is described in Chapter 7 in *Implementing Siebel eBusiness Applications on DB2 UDB for OS/390 and z/OS, Version 7.0.4*.

Adhering to the Siebel 7 install process will cause indexes to be generated with a primary allocation of 48 K and a secondary allocation of 1440 K. We believe that these allocations may be inadequate for some indexes. However, our testing did not enable us to identify specific indexes that would require a greater DASD allocation. We recommend that you leave the primary allocation to 48 K (keep it small), but raise secondary allocation for the indexes to 9600 K.

In this way, small indexes will take minimal space (perhaps as little as the primary allocation of 48 K), and larger indexes will not exceed the maximum number of extents.

Some Siebel databases contain a relatively large number of objects. Be aware of the impact of such databases:

* The DB2 Database Descriptor (DBD) will be large.

The DBD must be loaded contiguously into the DB2 Environmental Descriptor Manager (EDM) pool. DB2 will not load a DBD into an EDM pool if it is larger than 25% of the pool size. In the *DB2 V7 Installation Guide*, GC26-9936, it is recommended that the EDM pool generally should be 10 times the size of the
The largest database descriptor (DBD) or plan, whichever is greater. The Installation Guide gives techniques for determining DBD sizes.

- Log activity may be significant.
  DB2 logs the necessary undo/redo information for the unit of recovery in which the DBD was changed. Updating large DBDs can cause significant logging.

- Contention should be considered.
  Create, Alter, or Drop of a DB2 object locks the DBD. This may cause contention. The contention on active databases may cause DDL (Data Definition Language - SQL statements) to time out, or users may be locked out while DDL is being executed.

For these reasons, you may want to consider splitting databases that contain a large number of DB2 objects.

### 8.1.2 Free space

Most of the indexes and table spaces are defined with free space of 30%. Given the large amount of Varchar and Long Varchar data (approximately 60% of the columns), the 30% free space appears reasonable. However, this free space may add significantly to your DASD requirements.

### 8.1.3 EDM pool size

The EDM pool size for Siebel implementations may range from 20 MB to 150 MB. The major variants are the number of objects included in the database and the amount of storage devoted to the dynamic statement cache (or global cache). Siebel only uses dynamic SQL, so for performance reasons, the EDM pool size should allow a good reuse for the cached statements, to avoid the need to frequently reprepare the same SQL statements.

You can calculate the Global Cache Hit Ratio as follows:

\[
\text{Global Cache Hit Ratio (\%)} = \frac{\text{short prepares}}{\text{full prepares} + \text{short prepares}}
\]

The value should be above 90 to 95%; if it is not, then increase the EDM pool size. If you are close to the 2 GB addressability limit for the DBM1 address space, then you can use a data space for the part of the EDM pool used for global caching. In such a case, make sure that the data space is backed by real storage, to avoid increasing the system’s paging activity significantly.
8.1.4 Buffer pools

The following recommendations are for initially configuring the DB2 buffer pools in support of Siebel 7. These recommendations assume that the DB2 subsystem is dedicated to Siebel.

- **BP0**: restrict BP0 to the DB2 catalog and directory. This is a general DB2 data sharing recommendation which improves the performance of the dynamic SQL that Siebel executes.
- **BP1**: Siebel data
- **BP2**: Siebel indexes
- **BP7**: supports DB2 temporary storage (DSNDB07). The SQL Order By or Group By, for example, have the potential to require a DB2 sort, which DB2 supports through the use of the DSNDB07 work database. Siebel does a lot of sort.
- **BP9**: large Siebel table spaces (split out of existing table space)
- **BP10**: indexes for large Siebel table spaces
- **BP16K1**: Siebel data for 16 K table spaces
- **BP32K1**: Siebel data for 32 K table spaces

Based on this initial setup, you should monitor your environment and adapt accordingly.

The following is the buffer pools setup for a Siebel 7 environment with 5000 online users - a test performed in the DB2 Silicon Valley Lab.

- **BP0**: catalog and directory, 5000 buffers
- **BP1**: Siebel data, 20000 buffers
- **BP2**: Siebel indexes, 20000 buffers
- **BP7**: sort, 60000 buffers
- **BP9**: Not used
- **BP10**: Not used
- **BP16K1**: Siebel data for 16 K table spaces, 12000 buffers
- **BP32K1**: Siebel data for 32 K table spaces, 1000 buffers

There are two major differences between those two setups:

- Notice the sort buffer pool size (60000 buffers) compared to BP1 and BP2.
- BP9 and BP10 were not used; the performance achieved with BP1 and BP2 met the test targets, so it was not necessary to isolate large Siebel table spaces and associated indexes in their own buffer pools.
8.2 DB2 statistics

In this section we describe which DB2 statistics to collect for Siebel 7, and how to use them to monitor the database.

8.2.1 Why collect statistics

Statistics about the data contained within DB2 objects are required for two main reasons:

- To provide information about the space occupied by the data and the state of the underlying data sets. These figures can be used to:
  - Determine the optimum sizing for the underlying data sets
  - Determine the correct settings for PCTFREE and FREEPAGE
  - Determine when to reorganize the data
  - Monitor growth for capacity planning
  - Monitor the effectiveness of the compression dictionary
- To provide data that the DB2 optimizer can use to determine the correct access path to the data, for queries, at either BIND or PREPARE time.

Both sets of statistics can also be used by DBAs to reevaluate physical database design.

8.2.2 What statistics should you gather for Siebel

Gathering statistics can be performed from the Siebel server, but can also be run natively from DB2.

For tables with data, we recommend that you gather statistics on all columns. For indexes with data, we recommend that you gather statistics with KEYCARD and FREQVAL parameters (use defaults for FREQVAL).

Note: Columns in SYSCOLDIST contain statistics about distributions and correlated key values. Specifying the KEYCARD option of RUNSTATS allows you to collect key cardinality statistics between FIRSTKEYCARDF and FULLKEYCARDF (which are collected by default). Specifying the FREQVAL option of RUNSTATS allows you to specify how many key columns to concatenate, and how many frequently occurring values to collect. By default, the 10 most frequently occurring values on the first column of each index are collected.

8.2.3 Monitoring VSAM data set extents

DB2 statistics can help in monitoring VSAM data set extents.
The maximum number of extents has increased from 119 to 251 with DFP 1.4. Although this was a concern in older devices, it is less a performance issue with new devices like RVA and ESS, where data is physically stored in small pages all over the I/O devices. Nevertheless, it is still a performance issue during open/close and the SWITCH phase of online REORG.

There are upper bounds to the number of extents. The size of each extent may influence the limit of maximum volumes per data set (59 volumes per data set and 133 volumes per STOGROUP).

Continuous monitoring of the extents in SYSTABLEPART and SYSINDEXPART is necessary to track objects that exceed the number of extents past the threshold value of 50 (this can be done through the EXTENTS field of those two tables). When this value is exceeded, REORG should be scheduled for the object (TS or IX).

You will need to review the values of PRIQTY and SECQTY. If one or both of these values have changed, then avoid the REUSE parameter, so that the data set can be deleted and recreated with the new extents. PCTFREE and FREEPAGE may also be reduced, to squeeze more data into less space.

With DB2 V7 enhancements to parallel data set open, FASTSWITCH for online REORG, and the disk capabilities provided by PAV, the requirement to reduce the number of extents is certainly not so critical in terms of performance. However, it is good practice to keep the environment under control in terms of DSMAX and VTOC size.

### 8.3 DB2 history statistics

DB2 Version 7 introduces the ability to keep historical statistics relating to objects. The use of history statistics provides a way to study trends, to determine when utilities (such as REORG) should be run for maintenance, and to aid in space management. Additionally, the historical data could be used for reinstating old catalog statistics in case of access-path degradation.

By using the HISTORY option either during RUNSTATS or inline REORG, DB2 will update the following new history catalog tables:

- SYSIBM.SYSCOLDIST_HIST
- SYSIBM.SYSCOLUMNS_HIST
- SYSIBM.SYSINDEXES_HIST
- SYSIBM.SYSINDEXPART_HIST
- SYSIBM.SYSINDEXSTATS_HIST
- SYSIBM.SYSLOBSTATS_HIST
- SYSIBM.SYSTABLEPART_HIST
Those new catalog tables provide historical statistics for other catalog tables. For instance, SYSIBM.SYSTABLESPACE_HIST provides statistics for activity in SYSIBM.SYSTABLESPACE, SYSIBM.SYTABLEPART_HIST provides statistics for activity in SYSIBM.SYTABLEPART, and so on.

When DB2 adds or changes rows in a catalog table, DB2 might also write information about the rows to the corresponding catalog history table. Although the catalog history tables are not identical to their counterpart tables, they do contain the same columns for access path information and space utilization information.

In the following sections, we show two examples of how to take advantage of those new historical statistics tables for a Siebel implementation.

### 8.3.1 Monitoring space growth

Table SYSTABLEPART_HIST has two fields, CARDF and SPACEF, which contain the values for total number of rows and total amount of DASD space allocated to the table space or partition space, respectively. Similar fields also exist for index space in SYSINDEXPART_HIST.

CARDF and SPACEF can be tracked regularly for space growth of an object. Assuming constant growth over time, these numbers can be used to derive growth trends to plan for future needs. Consider the sample SQL shown in Example 8-1.

**Example 8-1 Query to get table space growth information**

```sql
SELECT
  MAX(CARDF), MIN(CARDF), ((MAX(CARDF) - MIN(CARDF))*100)/MIN(CARDF),
  (DAYS(MAX(STATSTIME))-DAYS(MIN(STATSTIME)))
FROM SYSIBM.SYSTABLEPART_HIST
WHERE DBNAME='DB' AND TSNAME='TS';
```

Assuming that the number of rows is constantly increasing, so that the highest number is the latest, the query shows the percentage of rows added over a specific time period. This could be extrapolated to scale on a monthly or yearly basis. A similar query can be run against SYSINDEXPART_HIST to get index spaces growth information.
COPY utility
The space growth and SYSCOPY entries of a DB2 object can be used in determining the requirement for an image copy. If the space growth is greater than a pre-determined percentage growth value since the last full image copy, then the COPY utility can be triggered to take a full image copy of the object.

8.3.2 Monitoring the rebuild of the compression dictionary
When COMPRESS YES is set for a table space, the compression dictionary is initially built either via LOAD or REORG utility. The PAGESAVE field in SYSTABLEPART and SYSTABLEPART_HIST gives the percentage of pages saved due to the compression. The performance of the compression and decompression dictionary depends on the update activity on the table space. Over a time period, the PAGESAVE can be reduced below a threshold_limit value, and the DBA needs to rebuild the dictionary.

Regularly building the dictionary is expensive, and it should be avoided. The PAGESAVE value in SYSTABLEPART_HIST can be used to monitor the effectiveness of the dictionary. Assume the following situation occurs:

\[(\text{max(pagesave)} - \text{min(pagesave)}) \times 100 / \text{max(pagesave)} > \text{threshold_limit}\]

In this case, you should remove the keyword KEEPDICTIONARY from the REORG or LOAD utility. The utility will build a new dictionary, or else code the keyword in the utility command. We recommend a threshold_limit of 10%.

8.4 DB2 realtime statistics
The objective of this new function, made available through maintenance on DB2 V7, is to help in the database administration of DB2. DB2 now provides realtime statistics (RTS) to be used by database administrators or the automated task scheduler to determine which objects require REORG, RUNSTATS, or COPY.

The first planned users of this function are the DB2 Control Center for OS/390 and DB2 tools such as the DB2 Automation Tool.

Some of the statistics collected are:
- The number of rows, LOB values or index entries modified since last REORG, RUNSTATS, or COPY
- The physical space information such as the number of pre-formatted pages, allocated space, and the number of extents
The number of distinct pages updates and the time of first update since the last COPY

DB2 always generates realtime in-memory statistics for each table space and index space on your system. Statistics are generated for each partition for partitioned table spaces, and indexes. Optionally, these in-memory statistics can be externalized to two DB2 tables based on a user-specific interval (STATSINT in ZPARM), or when necessary.

When externalizing in-memory statistics, DB2 inserts a row for each partition or non-partitioned page set in SYSIBM.TABLESPACESTATS or SYSIBM.INDEXSPACESTATS. If a row exists, it will be updated. The absolute statistic values (for example, TotalRows) are replaced with the new values, and incremental values are summed with the in-memory statistics. Those two new tables are placed in the DSNRTSDB.DSNRTSTS table space.

You can decide when to run REORG, RUNSTATS and COPY utilities, or when to enlarge your data sets, by querying these tables. A new sample stored procedure, DSNACCOR, is provided to help you in this task.

The default scope for this stored procedure is to scan “all” data in the RTS tables and provide recommendations for “any” condition previously mentioned. To override either the scope of the data queried or the type of recommendation evaluated, the caller will pass along the appropriate parameter with the call to the stored procedure. Parameters can also be passed that override the default threshold settings. Calls to the stored procedure that do not provide parameters will employ default parameters (or thresholds), where appropriate. The stored procedure reports back the objects that require REORG, RUNSTATS, or COPY.

Note: The RTS tables are not used for access path determination by the DB2 optimizer. Executing RUNSTATS is still required to update the traditional catalog tables used by the DB2 optimizer. For data sharing installation, RTS must be started on each member in order to get total statistics.

You can find further information about RTS in DB2 UDB for OS/390 and z/OS Version 7 Administration Guide, SC26-9931, Appendix G and H. You can get the latest version of this book from the following link:


8.5 Database reorganization

The need for REORG is mainly driven by changes in the data. These changes can lead to disorganization of the data and cause, among other effects, longer I/O time to return data.
The need for REORG can also be driven by the need to activate compression, or to take into account the new space allocation induced by an ALTER TABLESPACE/INDEXSPACE PRIQTY and SECQTY.

There are many statistics in the DB2 catalog that can indicate when a REORG should be run; these include LEAFDIST, NEAROFFPOS, and FAROFFPOS, among others. Degradation of performance indicates that a REORG may be necessary, and the number of extents allocated to a DB2 object can also indicate a REORG is needed.

We recommend that the Siebel 704 table spaces be reorganized online with a frequency that supports application performance. If you’re looking for continuous availability, the SHRLVL CHANGE option is the one that gives almost 100% read/write access to the data during the REORG process. Due to the fact that there are many indexes on several of the tables, the FASTSWITCH - YES feature of the DB2 V7 online REORG utility will be highly desirable.

The REORG utility is invoked at the table space level. When there are multiple indexes on multiple tables within the table space, the Switch phase of REORG is exposed to a potentially long duration due to the renaming of underlying VSAM data sets. Without FASTSWITCH, DB2 must change data set names using AMS RENAME function, which can be time-consuming.

With FASTSWITCH, no VSAM rename is needed; the fifth data set qualifier changes from a fixed value (I0001) to a variable one, alternating between I0001 and J0001. Measurements have shown that DB2 with FASTSWITCH can effect the name change in approximately 10% of the time. However, even with FASTSWITCH, we recommend doing online REORG at quiet time.

The REORG utility can also be invoked at the index level. There are benefits to reorganizing indexes, such as reclaiming free space, establishing free space, consolidating extents, removing pseudo-deleted keys, and so on. Use online REORG against Siebel indexes to significantly reduce the impact of this process.

Since Version 6 of DB2, it is possible to gather the statistics during the execution of the REORG utility. Using this option reduces the elapsed time of a REORG followed by RUNSTATS. This is achieved by eliminating the need for RUNSTATS to scan the data to collect the statistics. Instead, this is achieved by reading the data during the REORG. The STATISTICS keyword allows this inline statistics gathering.

DB2 Version 7 added the collection of historical statistics. They are collected when the HISTORY keyword is added to the REORG utility syin.
8.6 Siebel-DB2 database recovery

In a Siebel-DB2 V7 environment, you should consider the following recovery requirements:

- Recovery to currency
- Disaster recovery
- Point-in-time recovery (PIT)

8.6.1 Recovery to currency

“Recovery to currency” focuses on how to handle a hardware failure, like the loss of a DASD device.

From the perspective of the Siebel environment, recovery to currency considerations are the same as for any DB2 environment. If you are an experienced DB2 user, your present recovery to currency strategy can be applied to your Siebel environment.

8.6.2 Disaster recovery

Disaster recovery addresses how you would handle the loss of your computing center.

From the perspective of the Siebel environment, disaster recovery considerations are the same as for any DB2 environment. If you have disaster recovery procedures that address your current DB2 environment, they will also likely address your Siebel disaster recovery requirements.

If you do not presently have DB2 disaster recovery procedures, you will want to evaluate the following potential starting points for disaster recovery support:

- Disaster recovery using Dump/Restore
  
  This is a frequent starting point for disaster recovery support. The scenario “PIT recovery using Dump/Restore utilities” on page 136 could also provide a starting point in addressing your disaster recovery requirements.

- Documented disaster recovery scenario

  In DB2 Administration Guide V7, SC26-9931, the index entry “disaster recovery” refers you to a section on preparing for disaster recovery, and to a disaster recovery scenario.

  If you ship copies of both your image copies and archive logs offsite, the documented scenario will enable you to recover to the last completed DB2 unit of recovery on the last archive you sent offsite. The scenario is popular; however, it does require practice.
Disaster recovery with additional hardware

This is a broad topic that varies widely in both user requirement and scenario implementation. The scenario “PIT recovery using suspension of DB2 updating” on page 139 could provide a starting point in addressing your disaster recovery requirements. This scenario will require high-speed hardware to get copies of your DB2 log, BSDS, and data offsite.

8.6.3 Point-in-time recovery

The usual reason for a point-in-time recovery is an application programming error or a flawed operational procedure. This exposure is always present, regardless of your hardware/software configuration. A point-in-time recovery, in a Siebel environment, has the potential to be the most disruptive outage you are likely to encounter.

Note: The discussion on recovery in this section relates to the DB2 database only. Keep in mind that if you perform a PIT recovery on the database you also need to recover the Siebel file system to the same point in time as the database.

You may need to recover all Siebel-DB2 objects to a prior point in time. Depending on how you have mapped your tables to table spaces, this could mean recovering 181 table spaces and 13,244 indexes to a prior point of consistency. Your usual point-in-time recovery techniques, which you probably regard as conventional at this time, may be insufficient in this environment.

A failure in the application caused by a programming defect, or in a flawed operational procedure, such as running a job twice, introduces the requirement for point-in-time recovery. As much as possible, you will want to avoid PIT recovery with preventive measures such as increased attention to:

- Change management
- Problem management
- Testing

Each of these disciplines is procedure-oriented and management-driven. As attention to these disciplines is increased, the need for point-in-time recovery is usually decreased. Unfortunately, the need for point-in-time recovery is never entirely eliminated. Consequently, you will want to make every effort to avoid having to do a point-in-time recovery, but you should be prepared to do one if required.
Unless you are facing an application error and you have guidance from Siebel on which exact subset of tables requires PIT recovery, you will be required to reset all Siebel tables to a prior point of consistency.

There are several techniques to effect a point-in-time recovery, including:
- PIT recovery using user-written application programs
- PIT recovery using DB2 utilities
  The DB2 Quiesce and Copy utilities are the primary tools.
- PIT recovery using a Dump/Restore scenario
  This scenario typically employs non-DB2 utilities.
- PIT recovery using a DB2 conditional restart
  Any conditional restart scenario is potentially risky. The benefit of this scenario is the “effectively free” definition of the candidate points of consistency. Candidate points of consistency are a set of points on the log. Consistency may be established at any point in the set that you choose.
- PIT recovery using suspension of DB2 updating
  DB2 update activity is suspended using the SET LOG command to SUSPEND/RESUME logging. This function was introduced into DB2 in V6.

In the following sections, we discuss these techniques in more detail.

**PIT recovery using user-written application programs**
This is a strategic direction and not a scenario. It acknowledges that data can be corrupted due to program error. If this happens, you may attempt to correct the contaminated data with application programs. If you fail to correct the data with application programming, a scenario such as one of those following could be used as a last resort. This approach is gaining favor among users striving for high availability.

In implementing an approach like this, it is important to determine which transactions will make the data more corrupt or propagate the errors. This information then can either be communicated to end users, or the DBA can use it to disable the erroneous processes.

**PIT recovery using DB2 utilities**
The scenario for a PIT recovery using DB2 utilities is as follows:
- Execute the QUIESCE utility on all of the tables that are candidates to be reset if a PIT recovery is required.
  This causes the DB2 buffers for the quiesced tables to be externalized to DASD.
Execute the COPY utility on all the table spaces that are candidates to be reset if a PIT recovery is required.

The obvious reason for this step is to back up the data. However, COPY will fail if it cannot externalize the DB2 buffers to DASD. That is the reason we invoked the QUIESCE utility first (to remove one reason why COPY may fail).

You may want to place your Image Copy output on DASD. With image copies on DASD, there is no error-prone and time-consuming tape handling. Both DASD and tape-resident image copies enable parallelism in the COPY and RECOVER utilities.

For the second time, execute the QUIESCE utility on all the tables that are candidates to be reset if a point-in-time recovery is required.

This invocation of QUIESCE will establish a log Relative Byte Address/Log Record Sequence Number (RBA/LRSN) that will be the point of consistency.

When it is necessary to recover to this point of consistency, RECOVER to the RBA/LRSN established by the second invocation of the QUIESCE utility.

With the combination of the COPY and the second QUIESCE, the RECOVER TORBA/TOLOGPOINT (RECOVER TOLOGPOINT is used to recover to a LRSN) will perform as efficiently as a RECOVER TOCOPY would perform, assuming no logging during the execution of this scenario.

When it is necessary to recover to this point of consistency, RECOVER all indexes on all the tables that have been reset to the prior point of consistency. The indexes must be synchronized with the data in the recovered table spaces.

DB2 V6 added the functional capability to recover the index from an image copy. The recovery of an index from an image copy, which was introduced in DB2 V6, is a significant performance benefit over prior versions of DB2 that rebuild the index (this includes reading the table to unload and construct all keys, sorting the keys, and then rebuilding the index). If you are going to use this scenario, it is recommended that you recover your indexes using image copies.

The benefits of this scenario are:

- There is minimal disruption to the user in preparing for the scenario.
  
  The user may see some slowness in response during the execution of the QUIESCE utility, but this can likely be scheduled during off-hours to minimize the disruption.

- There is no disruption to the user when COPY is run with Share-Level Change.

  COPY - Share-Level Change allows updates concurrent with COPY utility execution.
The recovery of the table spaces will be efficient.

RECOVER TORBA/TOLOGPOINT will perform as well as RECOVER TOCOPY, assuming no logging between the execution of the COPY utility and the second execution of QUIESCE.

There is also a disadvantage to this scenario: the requirement to recover many objects may take too long, thus making this scenario impractical. However, given the performance improvements introduced in DB2 V6 (which includes parallelism in COPY and RECOVER, Fast Log Apply, and RECOVER of indexes from image copies), this scenario may meet your performance requirements. If you are evaluating the use of this scenario, time it to determine if it meets your availability requirements.

**PIT recovery using Dump/Restore utilities**

This scenario uses a DB2 command and usually a non-DB2 Dump/Restore utility program.

- Use the DB2 STOP DATABASE command to stop all of the table spaces.
  
The STOP DATABASE command will cause the DB2 buffers to be externalized to DASD, and will cause the VSAM data sets that hold the DB2 data to be closed. While the table spaces are stopped, the data will be unavailable to users. You may evaluate bringing down DB2 as an alternative to the STOP DATABASE command.

- Dump the table space and index data sets, the DB2 log, and the BSDS using your installation's high-speed Dump utility. You may consider using FlashCopy® or SnapShot to dump the data sets. The dumped data sets represent your point of consistency.

- When it is necessary to recover to this point of consistency, do the following:
  
  - Stop DB2.
  - Restore the data sets that were dumped in the previous step.
  - Restart DB2.

Because the table spaces were stopped when backed up, there were no outstanding units of recovery, and the data was consistent. The restored data consists of the table spaces and associated indexes, the DB2 log, and the BSDS.

The recovery portion of this scenario is faster than the previous one, but preparing for it is more disruptive to the user. The data is unavailable to the user while the table spaces are stopped and while the data is being dumped. The length of time that data is unavailable can be lessened by using FlashCopy or SnapShot. See the IBM Redbook *Implementing ESS Copy Services on S/390®, SG24-5680*, for more information on FlashCopy.
PIT recovery using DB2 conditional restart

This scenario will likely appeal to the more experienced DB2 user. The scenario requires a DB2 conditional restart, which is a part of DB2 not frequently exercised by users. Its key advantage is an “almost free” establishment of a candidate point of consistency.

The first three steps of the scenario are new and will receive the major part of our attention here. Once those steps are complete, the remainder of this scenario is conceptually similar to the later steps of the scenario “PIT recovery using DB2 utilities” on page 134, with the difference that you will recover to currency and not to RBA/LRSN.

- Identify a set of candidate points of consistency.

Consider a list that contains many items (or rows). Each list item has two entries in columns: The first column is a timestamp, and the second column is the DB2 Log RBA/LRSN associated with that time. The list can be quite long (that is, showing many timestamps). This list of timestamps is your set of candidate points of consistency.

The list of candidate points of consistency might have an entry for each hour in the day or for each minute in the day. For each entry in the list, you have a timestamp and the corresponding DB2 Log RBA/LRSN. This allows you to map a specific time to a DB2 Log RBA/LRSN.

How do you build a list of timestamps and the associated Log RBA/LRSN? You start by defining a dummy database and table space. This will be a real DB2 database and table space, but there will be no activity against the dummy table space. Siebel will not know about this table space.

Once the dummy table space is defined, initiate a user-developed procedure that will periodically QUIESCE that dummy table space. Since you will allow no activity against the dummy table space, the QUIESCE will be fast. The QUIESCE will cause the Log RBA/LRSN and the timestamp to be entered into SYSIBM.SYSCOPY.

The entries within SYSIBM.SYSCOPY for the dummy table space make up your list of candidate points of consistency. If you do the QUIESCE each hour, there will be an entry for the dummy table space in SYSIBM.SYSCOPY each hour.

Note that the defined dummy table is only used as an aid in determining the RBA/LRSN in the log that corresponds to a given time. It is possible to calculate the RBA/LRSN without the dummy table but the calculation is complex; use of the table is simpler and less prone to error.

As an alternative, use DSNJU004 to print the bootstrap dataset and examine the list of recent checkpoints to get a time / LRSN table. The information may also be available in the logs in checkpoint records using dsn1logp.
Select the candidate point of consistency that best meets your requirements.

Suppose you determine that because of an application programming error, data in your Siebel system is inconsistent. This part cannot be automated. You must determine when the inconsistencies began to enter your system.

Suppose you determine that at 5:00 PM on January 14, 2002, erroneous data began to enter your system. You make the determination that you want to take your system back before that point in time.

You have one more task: query SYSIBM.SYSCOPY for the dummy table space entry before 5:00 PM on January 14, 2002. Once you determine that entry from the list, note the DB2 Log RBA/LRSN.

Where do you stand now? You have the Log RBA/LRSN of the nearest time before the data inconsistencies entered your system. You are now ready to make that Log RBA/LRSN, which relates to a candidate point of consistency, into a true point of consistency.

Make the candidate point of consistency into a true point of consistency.

This is the point at which you do the conditional restart. The conditional restart makes your candidate point of consistency the true point of consistency on the DB2 Log.

There is probably data inconsistency at the Log RBA/LRSN you identified. You are running an active Siebel system and it is likely that at the time you have identified, there was work in process (including in-flight units of recovery). However, you can make that Log RBA/LRSN a true point of consistency.

By doing a DB2 conditional restart, you can make the Log RBA/LRSN you identified into a point of consistency.

You will use the CHANGE LOG INVENTORY DB2 utility to create a conditional restart control record using the following statement, where XXXX is the true point of consistency you determined from your SYSIBM.SYSCOPY query:

```
CRESTART
    CREATE, FORWARD=YES, BACKOUT=YES, ENDRBA/ENDLRSN=XXXX
```

The conditional restart causes DB2 to truncate the log at your true point of consistency. Log entries beyond that point are disregarded. Additionally, DB2 removes from SYSIBM.SYSLGRNGX and SYSIBM.SYSCOPY any entries that occurred after the true point of consistency.

Recover all table spaces to the true point of consistency.

The conditional restart positions you to recover all your table spaces. Because of the conditional restart, use RECOVER to currency, not
RECOVER TORBA/TOLOGPOINT (recovery to an RBA or LRSN is common in most point-in-time recovery scenarios).

- Recover all indexes on the tables that have been reset to the prior point of consistency.
  
The indexes must be synchronized with the data in the recovered table spaces.

**PIT recovery using suspension of DB2 updating**

This scenario is functionally similar to the scenario “PIT recovery using Dump/Restore utilities” on page 136. The salient features of both scenarios are:

- Determine the set of objects requiring backup and recovery.
- Stop processing.
- Dump the set of objects.
- Restore the set of objects and restart DB2.

The unique characteristic of the “Point-in-time recovery using suspension of DB2 updating” scenario is the technique to stop DB2 update processing. Stopping update processing and then starting it again is accomplished by the SET LOG SUSPEND/RESUME command introduced in DB2 V6.

The specific characteristics of this “Point-in-time recovery using suspension of DB2 updating” scenario are:

- Determine the set of objects requiring backup and recovery. This is your entire Siebel system.
- Stop processing.
  
The unique feature of this scenario is the method for stopping the processing which is the ability to “suspend” and “resume” DB2 logging. This has the effect of “freezing” updates to your DB2 data while you make copies of the DB2 data, the DB2 log and the BSDS. Specifically, after logging is “suspended”, the DB2 log, BSDS, and the data are consistent.

When the SET LOG SUSPEND command is issued, the following occurs:

- A system checkpoint is taken.

  The scope of the command is single system-only. In a data sharing environment, the command will have to be entered for each member.

- Unwritten log buffers are externalized to DASD.
- The BSDS is updated with the high-written log RBA.
- A log-write latch is obtained to prevent further logging.
- Logging is suspended until the SET LOG RESUME command is issued.
- Dump the set of objects.
To take maximum advantage of this scenario, you must have a high-speed dump capability like FlashCopy or SnapShot. With this scenario, you include the DB2 log and BSDS to your list of objects to be dumped. The function of SUSPEND/RESUME logging makes the log consistent with the DB2 data.

- Restore the set of objects and restart DB2.

You restore the DB2 data, the BSDS, and the log. Since the log is consistent with the data, the point-in-time recovery scenario requires only a normal DB2 restart. In-flight units of recovery are backed out, just as with any normal DB2 restart. After the normal DB2 restart, the data will include committed units of recovery at the time that logging was initially suspended.

**PIT recovery considerations**

Your strategy should be to avoid a point-in-time recovery if possible, but also to provide a process that allows it to be performed when necessary. If a point-in-time recovery must be done, then choose the most effective recovery scenario consistent with your requirements. Following are the considerations for the five options previously discussed.

**PIT recovery using user-written application programs**

This is a preferred approach among users when possible. It has the following benefits:

- There is little or no disruption to the user.
- Data that has not been corrupted will be available to the user. However, the user may see the erroneous data prior to the recovery.
- There is no loss of data entered between the time when the data contamination was introduced into the system, and the time when the contaminated data was corrected.
- Fewer processor resources are likely to be required than using either of the other techniques.

Remember that this is an ideal and not a rigorous scenario that can be documented and tested. You must be prepared to reset your system by doing the point-in-time recovery when a “programming fix” is not possible or practical. Application programming may not be able to repair Siebel data; if Siebel defects exist, other point-in-time recovery techniques may be the only alternative.

**PIT recovery using DB2 utilities**

This scenario is preferred when few table spaces and indexes are to be recovered. However, the recovery of the number of table spaces and indexes in a Siebel system may imply an outage of many hours, making this scenario potentially impractical. If you believe that this scenario may be workable in your
environment, run a benchmark to make sure that your availability requirements can be met.

In running that benchmark, be sure to integrate the recovery performance enhancements introduced in DB2 V6, including parallel COPY/RECOVER, Fast Log Apply, and COPY support for indexes.

**PIT recovery using Dump/Restore utilities**
The issue regarding this scenario is the number of table spaces and indexes that must be recovered to an established point of consistency. As the number of tables and indexes to be recovered grows, dump-restore may become the practical alternative. Dumping packs and restoring data (using features like FlashCopy or SnapShot) is faster than a recovery-based scenario if the regularly scheduled dumping activities and database non-availability can be tolerated. Note that recovery is required only when errors occur, while backup occurs on a scheduled basis even if no errors ever occur.

The major disadvantage to this scenario is that the dumping of the data is disruptive to the user and occurs on a regular basis (usually daily). The Siebel system must be stopped in order to dump data that is consistent. Other applications must also be stopped if they are affected by Siebel being unavailable.

An additional disadvantage (affecting all PIT recovery scenarios to some degree, but particularly noticeable in this one) is that work is lost when data is reset to a prior point of consistency. Considering a once-per-week backup schedule, then on average, production data for half of a week is lost when a point-in-time recovery is executed using this scenario.

**PIT recovery using a DB2 conditional restart**
The main benefit to this scenario is that there is effectively no impact on the user to create the list of candidate points of consistency.

The time required to actually recover the table spaces and indexes will likely be somewhat longer than the time required to do the recovery described in the scenario “PIT recovery using DB2 utilities” on page 134, because this scenario will likely require more DB2 log processing during the recovery.

Since this scenario contains a conditional restart, anyone using it must first practice it. An improperly done conditional restart has the potential of severely damaging your DB2 subsystem.

In summary, users should consider this scenario when:
- Having no disruption in defining the candidate points of consistency is of the greatest significance.
- The long outage to actually perform recovery is acceptable.
- You are willing to practice conditional restart to develop and maintain the skills necessary for its success.

**PIT recovery using suspension of DB2 updating**

This scenario has the advantages of the scenario “PIT recovery using Dump/Restore utilities” on page 136, with the added benefit that suspended logging is less disruptive to the user than stopping/restarting the DB2 subsystem. Data consistency is assured with this scenario through copying the log and BSDS in addition to the data, and then by executing a normal DB2 restart.

In addition, the SET LOG SUSPEND/RESUME feature may serve as a basis, providing a starting point for your offsite disaster recovery support.

**Choosing a PIT recovery scenario**

In choosing a point-in-time recovery scenario, you must evaluate three things:

- First you must evaluate the cost of preparing for the point-in-time recovery. The factors you will consider are the time it takes to back up your data, and the disruption the backup will cause your users.
- After evaluating the cost of preparing for the PIT recovery, you must then evaluate the cost of actually performing your PIT recovery scenario. You will evaluate the duration of the outage.
- And finally, you must evaluate the impact of lost data inherent in PIT recovery.

In the following sections, we examine these issues in more detail.

**Evaluate the user impact when preparing for the PIT recovery**

Although preparation or taking backups is much less time-consuming than the actual point-in-time recovery, it is potentially more disruptive because you prepare or take backups weekly, daily, or even hourly based upon your requirements. Since you may never actually do a point-in-time recovery, the preparation costs must be carefully weighed.

Example 8-1 on page 143 rates the previously discussed PIT recovery scenarios in terms of time required and disruption to the user. The rating scale is 1 through 4, with 1 being the best.
Table 8-1 PIT recovery scenarios - rating time required and disruption to user

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Restart</td>
<td>1</td>
</tr>
<tr>
<td>Suspend Logging</td>
<td>2/3</td>
</tr>
<tr>
<td>DB2 Utilities</td>
<td>2/3</td>
</tr>
<tr>
<td>Dump/Restore</td>
<td>4</td>
</tr>
</tbody>
</table>

The Conditional Restart scenario is effectively non-disruptive. It is enabled by the periodic QUIESCE of a dummy table that later enables mapping a point on the log to a date and time.

The Suspend Logging scenario has the potential to be non-disruptive if managed properly. At issue is the duration of suspended logging. Suspended logging can cause update transactions to time out. You can limit this exposure by using a fast dump facility like FlashCopy. Also you can suspend the logging at a time when there is little or no updating going on in your system.

The potential disruption of the DB2 Utilities scenario is influenced by two things. First, image copies are required for the later recovery. Since COPY can be run “Sharelevel - Change”, your users should see no disruption in backing up the data. However, a QUIESCE is also required, and it can disrupt your system availability. As with the Suspend logging scenario, this can be minimized by scheduling the process during a time of little or no update activity.

The Dump/Restore scenario is the most disruptive to end-user system availability. It requires either stopping all objects or bringing down DB2. Although you might consider this scenario in a non-data sharing environment, it is unlikely that you would use it in a data sharing environment. Bringing down a data-sharing system (the most straightforward way to implement this scenario) runs contrary to the high availability reason for choosing data sharing in the first place.

Evaluate the impact of executing of the PIT recovery scenario

There are two factors that must be considered when evaluating the execution of a PIT recovery scenario: the amount of time it is likely to require to do the recovery, and the amount of potential data loss. Unfortunately, the faster-running scenarios carry the potential of greater data loss. Consequently, two ratings will be presented: the first based upon the time to do the recovery, and the second based upon the potential for data loss.
Evaluate the time to do the PIT recovery

Table 8-2 rates the previously discussed PIT recovery scenarios in terms of time required to do the recovery. The rating scale is 1 through 4, with 1 being the best.

Table 8-2  PIT recovery scenarios - rating time required to do the recovery

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suspend Logging</td>
<td>1/2</td>
</tr>
<tr>
<td>Dump/Restore</td>
<td>1/2</td>
</tr>
<tr>
<td>DB2 Utilities</td>
<td>3</td>
</tr>
<tr>
<td>Conditional Restart</td>
<td>4</td>
</tr>
</tbody>
</table>

The Suspend Logging scenario and the Dump/Restore scenario are significantly faster than the other alternatives because they require only a restart of DB2 after restoring the data. No further recovery is required.

Both the DB2 Utility and the Conditional Restart scenarios require recovery of the entire subsystem data. The DB2 Utility scenario is likely to be faster because there is likely to be less log processing. Since the QUIESCE is done at the time the image copies are made, minimum log processing is likely.

However, with the DB2 Utility scenario, you may choose to QUIESCE more frequently than you COPY or back up the data in order to establish more points of consistency. If you do this, there will be more log processing if you select a QUIESCE point that is not the same as the point of the backup copies.

Note: You may consider more frequent invocations of the QUIESCE utility when using the DB2 Utility scenario, to more effectively manage the loss of data.

Evaluate impact of lost data using the PIT recovery scenarios

Table 8-3 on page 145 rates the previously discussed PIT recovery scenarios in terms of the potential loss of data. Since data is being reset to a prior point of consistency, data entered after that point of consistency is lost. The rating scale is 1 through 4, with 1 being the best.
Table 8-3  PIT recovery scenarios - rating the potential loss of data

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Restart</td>
<td>1</td>
</tr>
<tr>
<td>DB2 Utilities</td>
<td>2</td>
</tr>
<tr>
<td>Suspend Logging</td>
<td>3</td>
</tr>
<tr>
<td>Dump/Restore</td>
<td>4</td>
</tr>
</tbody>
</table>

The Conditional Restart scenario offers the capability to recover to the point in time nearest to the start of data corruption. This is enabled by the frequent QUIESCE of the dummy table to establish many candidate points of consistency. Consequently, the loss of data is more effectively minimized with this scenario.

There are some similarities in potential data-loss characteristics between the DB2 Utility scenario and the Suspend Logging scenario. Both are driven mainly by the frequency with which the point of consistency is established.

The point of consistency is established in the DB2 Utility scenario by execution of the QUIESCE utility. The point of consistency for the Suspend Logging scenario is established when the data is FlashCopied. It is easier to QUIESCE more frequently (perhaps several times a day) than it is to execute a FlashCopy process.

Additionally, management of the resulting backup points of consistency is easier with the DB2 Utility scenario. The QUIESCE point is maintained by DB2 in SYSIBM.SYSCOPY. Managing multiple FlashCopies is a greater challenge.

Consequently, since both scenarios enable more frequent backups, but since the establishment of the point of consistency and management of the backups is easier with the DB2 Utility scenario, the DB2 Utility scenario is rated 2 and the Suspend Logging scenario is rated 3.

Since the backup for the Dump/Restore scenario is the most disruptive, it is the scenario that is least likely to be executed more than daily. Consequently, this scenario has the greatest exposure to data loss.

In all cases, you also need to develop a strategy to recover the data that you have lost between the point in time to which you recovered and the current time. You will have to rerun the applications that occurred after the point in time to which you recovered.
Summary: Choosing a PIT recovery scenario

Choosing an PIT recovery scenario is ultimately a business decision. You have three key factors to evaluate. They are:

- The disruption caused by taking the backup
- The disruption caused by executing a PIT recovery
- The potential for lost data

When selecting your PIT recovery strategy, keep your business requirements in mind as you prioritize each of these factors.
In this appendix, we describe the DB2 parameters we used at the ITSO. These parameters include the required DSNZPARMS for Siebel, and the values used at the IBM Silicon Valley Lab during the Scalability Characterization tests.
A.1 DSNZPARM values used for the Siebel 7 installation

//D7R1JUN JOB (999,POK), 'D7R1 INSTALL', CLASS=A, MSGCLASS=T,
// NOTIFY=RC04, TIME=1440, REGION=OM
/* JOBPARM SYSAFF=SC04, L=9999
// JCLLIB ORDER=(DB7RU.PROCLIB)
="/*******************************************************************************/
/* JOB NAME = DSNTIJUZ */
/* DESCRIPTIVE NAME = INSTALLATION JOB STREAM */
/* LICENSED MATERIALS - PROPERTY OF IBM */
/* 5675-082 */
/* (C) COPYRIGHT 1982, 2000 IBM CORP. ALL RIGHTS RESERVED. */
/* STATUS = VERSION 7 */
/* FUNCTION = DSNZPARM AND DSNHDECP UPDATES */
/* PSEUDOCODE = */
/* DSNTIZA STEP ASSEMBLE DSN6... MACROS, CREATE DSNZPARM */
/* DSNTIZL STEP LINK EDIT DSNZPARM */
/* DSNTLOG STEP UPDATE PASSWORDS */
/* DSNTIZP STEP ASSEMBLE DSNHDECP DATA-ONLY LOAD MODULE */
/* DSNTIZQ STEP LINK EDIT DSNHDECP LOAD MODULE */
/* DSNTIMQ STEP SMP/E PROCESSING FOR DSNHDECP */
/* NOTES = STEP DSNTIMQ MUST BE CUSTOMIZED FOR SMP. SEE THE NOTES */
/* NOTES PRECEDING STEP DSNTIMQ BEFORE RUNNING THIS JOB. */
="/*******************************************************************************/
//DSNTIZA EXEC PGM=ASMA90, PARM='OBJECT,NODECK'
//SYSLIB DD DISP=SHR,
// DSN=DB7R7.SDSNMACS
// DD DISP=SHR,
// DSN=SYS1.MACLIB
//SYSLIN DD DSN=&LOADSET(DSNTILM1), DISP=(NEW, PASS),
// UNIT=VIO,
// SPACE=(800,(50,50,2)), DCB=(BLKSIZE=800)
//SYSPRINT DD SYSSOUT=* 
//SYSUDUMP DD SYSSOUT=* 
//SYSUT1 DD UNIT=VIO, SPACE=(800,(50,50),,ROUND)
//SYSUT2 DD UNIT=VIO, SPACE=(800,(50,50),,ROUND)
//SYSUT3 DD UNIT=VIO, SPACE=(800,(50,50),,ROUND)
//SYSIN DD *
// DSN6ENV MVS=XA
// DSN6SPRM RESTART,
ALL,
ABEXP=YES,
ABIND=YES,
AUTH=YES,
AUTHCACH=1024,
BINDNV=BINDADD,
BMPTOUT=4,
CACHEDYN=YES,
CACHEPAC=32768,
CACHERAC=32768,
CATALOG=DB7RU,
CDSSRDEF=1,
CHGDC=NO,
CONTSTOR=YES,  
DBACRVW=NO,
DECDIV3=NO,
DEFLTID=IBMUSER,
DESCSTAT=NO,
DLITOUT=6,
DMLMAX=18700,  
EDMBFIT=NO,
EDMDSMAX=100000,  
EDMPool=40000,  
EDMDP=NO,
EDMRD=NO,
EVALUNC=YES,  
HOPAUTH=BOTH,
IRLMAUT=YES,
IRLMPRC=D7R1IRLUM,
IRLMSID=I7R1,
IRLMRWT=300,  
IRLMSWT=600,  
LEMAX=20,
MAXBLK=8000,  
MAXKEEPD=0,  
MINDVSC=NONE,
MINSTOR=YES,  
MXTBJOIN=70,  
NPGTHRSH=10,  
NUMKTS=1000,
NUMKUS=100000,  
OPTHINTS=YES,
PARAMDEG=1,
PARTKEYU=YES,
RECALL=YES,
RECALLD=120,
RELCURRL=YES,
RETLWAIT=0,
RETVLCFk=NO,
RGFCOLID=DSNRGCOL,
RGFDBNAM=DSNRGDB,
RGFDEDPL=NO,
RGFDEFLT=ACCEPT,
RGFESCp=,
RGFFULLQ=YES,
RGFINSTL=NO,
RGFMORT=DSN_REGISTER_OBJT,
RGFMNRT=DSN_REGISTER_APPL,
RRULOCK=NO,
SEQCACH=SEQ,  
SEQPRES=YES,
SITETYP=LOCALSITE,
SJTABLES=10,
SRTPool=12000,  
STARJOIN=DISABLE,
STATHIST=NONE,
STATROLL=NO,
STATSINT=30,
SUPERRS=YES,
SYSADM=RCO4,
SYSADM2=HAIMO,
SYSOPR1=SYSPRP,
SYSOPR2=SYSPRP,
TRKRSITE=NO,
UTIMOUT=6,
XLKUPDLT=NO

DSN6ARVP
   ALCUNIT=BLK,
   ARCWRTC=(1,3,4),
   ARCWTOR=YES,
   ARCPFX1=DB7RU.D7R1.ARCLG1,
   ARCPFX2=DB7RU.D7R1.ARCLG2,
   ARCRETN=9999,
   BLKSIZE=28672,
   CATALOG=YES,
   COMPACT=NO,
   PRIQTY=1234,
   PROTECT=NO,
   QUIESCE=5,
   SECQTY=154,
   SVOLARC=NO,
   TSTAMP=NO,
   UNIT=3390,
   UNIT2=

DSN6LOGP
   DEALLCT=(0),
   MAXARCH=1000,
   MAXRTU=2,
   OUTBUFF=4000,
Appendix A. DB2 parameters

TWOACTV=YES,
TWOARCH=YES,
ARC2FRST=NO

AUDITST=NO,
BACKODUR=5,

CHKFREQ=15, <- WAS 50000
CONDBAT=10000, <- WAS 64
CTHREAD=70,

DBPROTCL=DRDA,
DLDFREQ=0, <- WAS 5

DSN6SYSP

EXTRAREQ=100,
EXTRASRV=100,

EXTSEC=YES, <- WAS NO
IDBACK=100, <- WAS 20
IDFORE=40,

IDXBPOLL=BP2, <- WAS BP0

LBACKOUT=AUTO,
LOBVALA=2048,
LOBVALUES=2048,
LOGAPSTG=0,

MAXDBAT=500, <- WAS 64
MON=NO,
MONSIZE=32768,
PCLOSEN=5,
PCLOSET=10,
RLF=NO,

RLFBL=01,
RLFERR=NOLIMIT,
RLFAUTH=SYSIBM,
ROUTCDE=(1),

SMFACCT=NO,
SMFSTAT=NO,
STATIME=1,
STORMXAB=0,
STORPROC=D7R1SPAS,
STORTIME=180,

SYNCVAL=NO,

TBSPPOOL=BP1, <- WAS BP0
TRACSTR=NO,
TRACTBL=16, <- WAS 4
URCHKTH=0,

WLMENV=

DSN6FAC

DDF=AUTO,

CMSTART=INACTIVE, <- WAS ACTIVE
IDHTOIN=600, <- WAS 0
RESYNC=2,
RLFERRD=NOLIMIT,
TCPALVER=NO,
MAXTYPE=0,
TCPKALV=ENABLE,
POOLINAC=120

DSN6GRP
DSHARE=YES,
GRPNAME=DB7RU,
MEMBNAME=D7R1,
COORDNTR=NO,
ASSIST=NO,
IMMEDWRI=NO

END

//*********************************************************************
//* LINK EDIT THE NEW DSNZPARM MEMBER. PUT LOAD MODULE IN
//* SDSNEXIT. *
//*********************************************************************
//DSNTIZL EXEC PGM=IEWL,PARM='LIST,XREF,LET,RENT',
//COND=(4,LT)
//ADSNLOAD DD DISP=SHR,
//DSN=DB7R7.SDSNLOAD
//DD DISP=SHR,
//DSN=DB7R7.ADSNLOAD
//SYSPUNCH DD DSN=&&LOADSET(DSNTILM1),COND=(4,LT)
//SYSLMOD DD DISP=SHR,
//DSN=DB7R7.SDSNEXIT
//SYSPRINT DD SYSDUMP DD SYSPUNCH(DSNTILM1)
//SYSLIN DD *
//INCLUDE SYSPUNCH(DSNTILM1)
//INCLUDE ADSNLOAD(DSNZPARM)
//ORDER DSNAA
//INCLUDE ADSNLOAD(DSNAA)
//INCLUDE ADSNLOAD(DSNF5SYS)
//INCLUDE ADSNLOAD(DSNJARVP)
//INCLUDE ADSNLOAD(DSNJLOGP)
//INCLUDE ADSNLOAD(DSNTPAR)
//INCLUDE ADSNLOAD(DSNVDIR1)
//INCLUDE ADSNLOAD(DSN3DIR1)
//INCLUDE ADSNLOAD(DSN7GRP)
ENTRY DSN2MSTR
NAME DSNZPAR1(R)

//*
//* CHANGE LOG INVENTORY:
//* UPDATE BSDS
//*
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=DB7R7.SDSNLOAD
//SYSLIN DD DISP=OLD,DSN=DB7RU.D7R1.BSDS01
Appendix A. DB2 parameters

//SYSUT2 DD DISP=OLD,DSN=DB7RU.D7R1.BSOS02
//SYSPRINT DD SYSOUT=
//SYSSDUMP DD SYSOUT=
//SYSIN DD *

DDF LOCATION=DB7R,LUNAME=SCPD7R1,
      NOPASSWD,REPORT=33751,PORT=33750

DATASHR ENABLE

* WARNING! DO NOT CHANGE ANY PARAMETERS IN THE GROUP
* STATEMENT BELOW!
GROUP GROUPNAM=DB7RU,GROUPMEM=D7R1,MEMBERID=1
//

//*********************************************************************
//*   ASSEMBLE AND LINK EDIT DATA-ONLY LOAD MODULE DSNHDECP.
//*   THE FOLLOWING STEPS ARE NEEDED ONLY IF THE
//*   VALUES ARE CHANGED FROM THOSE WHICH ARE SHIPPED.
//*********************************************************************
//DNSTIZP EXEC PGM=ASMA90,PARM='OBJECT,NODECK',COND=(4,LT)
//SYSLIB DD DISP=SHR,
//         DSN=DB7R7.SDSNMACS
//SYSLIN DD DSN=&&LOADSET(DSNHDECA),DISP=(NEW,PASS),
//          UNIT=VIO,SPACE=(80,(50,50,2)),DCB=(BLKSIZE=80)
//SYSPRINT DD SYSOUT=
//SYSUDUMP DD SYSOUT=
//SYSUT1 DD UNIT=VIO,SPACE=(800,(50,50),,,ROUND)
//SYSUT2 DD UNIT=VIO,SPACE=(800,(50,50),,,ROUND)
//SYSUT3 DD UNIT=VIO,SPACE=(800,(50,50),,,ROUND)
//SYSIN DD *

DSNHDECM CHARSET=ALPHANUM,
     ASCCSID=5348,<-- WAS 0
     AMCCSID=65534,
     AGCCSID=65534,
     SCCSID=1140,<-- WAS 37
     MCCSID=65534,
     GCCSID=65534,
     USCCSID=367,
     UMCCSID=1208,
     UGCCSID=1200,
     ENSCHEME=EBLCDIC,
     APPENSCH=EBLCDIC,
     DATE=ISO,
     DATELEN=0,
     DECARTH=DEC15,
     DECIMAL=PERIOD,
     DEFLANG=IBMCOB,
     DELIM=APOST,<-- WAS DEFAULT
     MIXED=NO,
     SQLDELI=DEFAULT,
     DSQDELI=APOST,
SSID=D7RG,
STDSQL=NO,
TIME=ISO,
TIMELEN=0,
DYNRULS=YES,
LC_CTYPE=,
COMPAT=OFF
END

//*********************************************************************
//* DO SMP/E PROCESSING TO TRACK DSNHDECP CHANGES.*
//* THIS STEP IS ONLY USED WHEN THE DEFAULT DSNHDECP IS NOT
//* SUITABLE.*
//* NOTE: THIS STEP MUST BE CUSTOMIZED AS FOLLOWS FOR SMP:
//* 1. LOCATE AND CHANGE THE FOLLOWING STRINGS TO THE VALUES
//*    YOU SPECIFIED FOR THEM IN JOB DSNTIJAE:
//*    A.'?SMPPRE?' TO THE PREFIX OF YOUR SMP LIBRARY NAME.
//*********************************************************************
/* 2. UPDATE SYSOUT CLASSES AS DESIRED (DEFAULT IS '*')
***************************************************************
//DSNTIMQ EXEC PGM=GIMSMP,PARM='CSI=\?SMPPRE\.?SMPMLQ\.?CSI',
    //              REGION=4096K,COND=(2,LT)
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SMPCNTL DD *
    //          SET BDY('DSNTARG').
    //JCLIN.
//SMPJCLIN DD DISP=SHR,
    //          DSN=DB7RU.D7R1.NEW.SDSNSAMP('DSNTIJUZ')
/*
Sample Siebel 7 generate DDL output

This appendix contains a portion of the DDL generated by the Siebel 7 install process.
B.1 Sample generate DLL output

This sample DDL illustrates the type of database(22), table space(181), table(2,485), grant(1 per table), and index(13,244) control statements generated by the Siebel install process.

This DDL was generated using the standard control file supplied by Siebel to create multiple tables per table space with no partitioning.

```
//SIEBVAL JOB (POK,999),DORAN,MSGLEVEL=(1,1),MSGCLASS=X,
// CLASS=A,NOTIFY=&SYSUID
//*JOBPARM SYSAFF=SC04,L=9999
//*
//******************************************************************************
//DSNTIST EXEC PGM=IKJEFT01
//STEPLIB DD DSN=DB7RU.RUNLIB.LOAD,DISP=SHR
//         DD DSN=DB7R7.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSSDUMP DD SYSOUT=*
//SYSSIN DD *
 DSN SYSTEM(D7R1)
   RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71)
END
//SYSSIN DD *
SET CURRENT SQLID = 'SIEBTBO';
CREATE DATABASE SIEBDB01 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB02 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB03 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB04 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB05 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB06 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB07 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB08 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB09 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB10 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB11 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB12 CCSID ASCII BUFFERPOOL BP1;
```
commit;
CREATE DATABASE SIEBDB13 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB14 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB15 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB16 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB17 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB18 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB19 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB20 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBDB21 CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE DATABASE SIEBTMP CCSID ASCII BUFFERPOOL BP1;
commit;
CREATE TABLESPACE F0001032 IN SIEBDB00 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP32K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
CREATE TABLESPACE F0002016 IN SIEBDB00 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP16K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
CREATE TABLESPACE H0001016 IN SIEBDB00 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP16K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
CREATE TABLESPACE H0001032 IN SIEBDB00 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP32K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
CREATE TABLESPACE H0002016 IN SIEBDB00 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP16K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
CREATE TABLESPACE H0EIM016 IN SIEBDB17 USING STOGROUP SIEBSTO
  PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32
  BUFFERPOOL BP16K1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO;
commit;
*****
***** THIS WOULD CONTINUE ON TO CREATE 181 TOTAL TABLESPACES
*****
*****

Appendix B. Sample Siebel 7 generate DDL output 159
CREATE TABLE SIEBTBO.EIM_ACCNT_DTL (  
MS_IDENT NUMERIC(15,0) GENERATED ALWAYS AS IDENTITY, 
ROW_ID VARCHAR(15) NOT NULL, 
CREATED TIMESTAMP DEFAULT , 
CREATED_BY VARCHAR(15), 
LAST_UPD TIMESTAMP DEFAULT , 
LAST_UPD_BY VARCHAR(15), 
MODIFICATION_NUM NUMERIC(10,0) DEFAULT 0, 
CONFLICT_ID VARCHAR(15) DEFAULT '0', 
IF_ROW_BATCH_NUM NUMERIC(15,0) DEFAULT 0 NOT NULL, 
IF_ROW_STAT VARCHAR(30) NOT NULL, 
PARTY_TYPE_CD VARCHAR(30) NOT NULL, 
PARTY_UID VARCHAR(100) NOT NULL, 
IF_ROW_STAT_NUM NUMERIC(10,0) DEFAULT 0, 
ACC_PR_LOGO CHAR(1), 
EXT_ATTRIB_08 CHAR(1), 
EXT_ATTRIB_09 CHAR(1), 
EXT_ATTRIB_10 CHAR(1), 
EXT_ATTRIB_11 CHAR(1), 
EXT_ATTRIB_12 TIMESTAMP, 
EXT_ATTRIB_13 TIMESTAMP, 
EXT_ATTRIB_14 NUMERIC(22,7), 
EXT_ATTRIB_15 NUMERIC(22,7), 
EXT_ATTRIB_16 NUMERIC(22,7), 
EXT_ATTRIB_17 NUMERIC(22,7), 
EXT_ATTRIB_18 NUMERIC(22,7), 
EXT_ATTRIB_19 NUMERIC(22,7), 
EXT_ATTRIB_20 NUMERIC(22,7), 
EXT_ATTRIB_21 NUMERIC(22,7), 
EXT_ATTRIB_22 NUMERIC(22,7), 
EXT_ATTRIB_23 NUMERIC(22,7), 
EXT_ATTRIB_24 NUMERIC(22,7), 
EXT_ATTRIB_25 NUMERIC(22,7), 
EXT_ATTRIB_26 TIMESTAMP, 
EXT_ATTRIB_27 TIMESTAMP, 
EXT_ATTRIB_28 TIMESTAMP, 
EXT_ATTRIB_29 TIMESTAMP, 
EXT_ATTRIB_30 TIMESTAMP, 
EXT_ATTRIB_31 TIMESTAMP, 
EXT_ATTRIB_32 TIMESTAMP, 
EXT_ATTRIB_54 TIMESTAMP, 
EXT_ATTRIB_55 TIMESTAMP, 
EXT_ATTRIB_56 CHAR(1), 
EXT_ATTRIB_57 NUMERIC(22,7), 
EXT_ATTRIB_58 NUMERIC(22,7), 
EXT_ATTRIB_59 NUMERIC(22,7), 
)
EXT_ATTRIB_60                  NUMERIC(22,7),
EXT_ATTRIB_61                  NUMERIC(22,7),
EXT_ATTRIB_62                  NUMERIC(22,7),
EXT_ATTRIB_63                  NUMERIC(22,7),
EXT_ATTRIB_64                  NUMERIC(22,7),
EXT_ATTRIB_65                  NUMERIC(22,7),
IT_LO_NUM1                     NUMERIC(10,0),
IT_LO_NUM2                     NUMERIC(10,0),
IT_LO_NUM3                     NUMERIC(10,0),
IT_LO_NUM4                     NUMERIC(10,0),
NOTE_PRIV_FLG                  CHAR(1),
SKLI_SCORE_VAL                 NUMERIC(10,0),
T_ACCNT_ATT__EXS               CHAR(1),
T_ACCNT_ATT__STA               NUMERIC(10,0),
T_ACCNT_ATT__UNQ               CHAR(1),
T_NOTEACNT__EXS                CHAR(1),
T_NOTEACNT__STA                NUMERIC(10,0),
T_NOTEACNT__UNQ                CHAR(1),
T_ORGEXT_XM__EXS               CHAR(1),
T_ORGEXT_XM__STA               NUMERIC(10,0),
T_ORGEXT_XM__UNQ               CHAR(1),
T_ORGSKILL__EXS                CHAR(1),
T_ORGSKILL__STA                NUMERIC(10,0),
T_ORGSKILL__UNQ                CHAR(1),
T_ORG_EXT_T__EXS               CHAR(1),
T_ORG_EXT_T__EXS1              CHAR(1),
T_ORG_EXT_T__STA               NUMERIC(10,0),
T_ORG_EXT_T__STA1              NUMERIC(10,0),
T_ORG_EXT_T__UNQ               CHAR(1),
T_ORG_EXT_T__UNQ1              CHAR(1),
T_ORG_EXT_X__EXS               CHAR(1),
T_ORG_EXT_X__STA               NUMERIC(10,0),
T_ORG_EXT_X__UNQ               CHAR(1),
T_ORG_EXT__EXS                 CHAR(1),
T_ORG_EXT__STA                 NUMERIC(10,0),
T_ORG_EXT__UNQ                 CHAR(1),
T_PARTY__EXS                   CHAR(1),
T_PARTY__STA                   NUMERIC(10,0),
T_PARTY__UNQ                   CHAR(1),
XM_ATTRIB_53                   NUMERIC(22,7),
XM_ATTRIB_54                   NUMERIC(22,7),
XM_ATTRIB_55                   NUMERIC(22,7),
ACCNNT_BI                      VARCHAR(15),
ACCNNT_BU                      VARCHAR(100),
ASGN_IT_TYPE_NAME          VARCHAR(75),
ATT_COMMENTS                   VARCHAR(250),
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATT_FILE_EXT</td>
<td>VARCHAR(10),</td>
</tr>
<tr>
<td>ATT_FILE_NAME</td>
<td>VARCHAR(200),</td>
</tr>
<tr>
<td>ATT_FILE_SRC_TYPE</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>VARCHAR(250),</td>
</tr>
<tr>
<td>EXT_ATTRIB_01</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>EXT_ATTRIB_02</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>EXT_ATTRIB_03</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>EXT_ATTRIB_04</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>EXT_ATTRIB_05</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>EXT_ATTRIB_06</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>EXT_ATTRIB_07</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>EXT_ATTRIB_34</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_35</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_36</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_37</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_38</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_39</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_40</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_41</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_42</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_43</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_44</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>EXT_ATTRIB_45</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>EXT_ATTRIB_46</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>EXT_ATTRIB_47</td>
<td>VARCHAR(255),</td>
</tr>
<tr>
<td>EXT_ATTRIB_48</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_49</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_50</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_51</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>EXT_ATTRIB_52</td>
<td>VARCHAR(250),</td>
</tr>
<tr>
<td>EXT_ATTRIB_53</td>
<td>VARCHAR(250),</td>
</tr>
<tr>
<td>IF_ROW_MERGE_ID</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>INCL_EXCL_CD</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>IT_COMMENTS</td>
<td>VARCHAR(250),</td>
</tr>
<tr>
<td>IT_EXPERTISE_CD</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>IT_LO_CHAR1</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>IT_LO_CHAR2</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>IT_LO_CHAR3</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>IT_LO_CHAR4</td>
<td>VARCHAR(250),</td>
</tr>
<tr>
<td>LOC</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(100),</td>
</tr>
<tr>
<td>NOTE_NOTE_TYPE</td>
<td>VARCHAR(30),</td>
</tr>
<tr>
<td>ORG_SKL_NAME</td>
<td>VARCHAR(50),</td>
</tr>
<tr>
<td>T_ACCNT_ATT_PARROW</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>T_ACCNT_ATT_RID</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>T_DELETED_ROW_ID</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>T_EXPORTED_ROW_ID</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>T_MERGED_ROW_ID</td>
<td>VARCHAR(15),</td>
</tr>
<tr>
<td>T_NOTEACCNT_SRCrow</td>
<td>VARCHAR(15),</td>
</tr>
</tbody>
</table>
T_NOTEACCNT__RID               VARCHAR(15),
T_ORGEXT_XM_PARROW             VARCHAR(15),
T_ORGEXT_XM__RID               VARCHAR(15),
T_ORGSKILLI_ORGSKI             VARCHAR(15),
T_ORGSKILLI__RID               VARCHAR(15),
T_ORG_EXT_BU_ID                VARCHAR(15),
T_ORG_EXT_PARROW               VARCHAR(15),
T_ORG_EXT_T_PARROW             VARCHAR(15),
T_ORG_EXT_T__RID               VARCHAR(15),
T_ORG_EXT_T__RID1              VARCHAR(15),
T_ORG_EXT_X_PARROW             VARCHAR(15),
T_ORG_EXT_X__RID               VARCHAR(15),
T_ORG_EXT__RID                 VARCHAR(15),
T_ORG_SKILL_ORG_ID             VARCHAR(15),
T_ORG_SKILL__RID               VARCHAR(15),
T_PARTY__RID                   VARCHAR(15),
XM_ATTRIB_48                   VARCHAR(30),
XM_ATTRIB_49                   VARCHAR(30),
XM_ATTRIB_50                   VARCHAR(30),
XM_ATTRIB_51                   VARCHAR(100),
XM_ATTRIB_52                   VARCHAR(100),
XM_NAME                        VARCHAR(100),
XM_TYPE                        VARCHAR(30),
NOTE_NOTE                      LONG VARCHAR) IN SIEBDB17.HOEIM016;
commit;
GRANT SELECT,INSERT,UPDATE,DELETE ON SIEBTBO.EIM_ACCNT_DTL TO
   SIEBGRP;
commit;
CREATE UNIQUE INDEX SIEBTBO.EIM_ACCNT_DTL_U1 ON
   SIEBTBO.EIM_ACCNT_DTL
   (IF_ROW_BATCH_NUM, ROW_ID)
   USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
   CLUSTER BUFFERPOOL BP2;
commit;
CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T01 ON SIEBTBO.EIM_ACCNT_DTL
   (IF_ROW_BATCH_NUM, T_DELETED_ROW_ID)
   USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
   BUFFERPOOL BP2;
commit;
CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T02 ON SIEBTBO.EIM_ACCNT_DTL
   (IF_ROW_BATCH_NUM, T_PARTY__RID)
   USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
   BUFFERPOOL BP2;
commit;
CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T03 ON SIEBTBO.EIM_ACCNT_DTL
   (IF_ROW_BATCH_NUM, T_PARTY__RID)
   USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
   BUFFERPOOL BP2;
commit;
CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T04 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ACCNT_ATT__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T05 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ACCNT_ATT_PARROW, ATT_FILE_NAME, ATT_FILE_EXT)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T06 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_NOTEACCNT__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T07 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T08 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT_T__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T09 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, NAME, LOC, T_ORG_EXT_BU_ID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T10 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT_X__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T11 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT_X_PARROW)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T12 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORGEXT_XM__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;

CREATE INDEX SIEBTBO.EIM_ACCNT_DTL_T13 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORGEXT_XM_PARROW, XM_TYPE, XM_NAME)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T14 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_SKILL__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T15 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_SKILL_ORG_ID, ORG_SKL_NAME)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T16 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_SKILL_ORG_ID, ASGN_IT_TYPE_NAME,
    INCL_EXCL_CD)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T17 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORGSKILLI__RID)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T18 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT_T__RID1)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
CREATE  INDEX SIEBTBO.EIM_ACCNT_DTL_T19 ON SIEBTBO.EIM_ACCNT_DTL
  (IF_ROW_BATCH_NUM, T_ORG_EXT_T_PARROW)
  USING STOGROUP SIEBSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
  BUFFERPOOL BP2;
commit;
Batch jobs for table space partitioning

This appendix contains the following batch jobs:

- The batch job used to create a backup table space and table prior to creating the partition table space.
- The batch job used to create a partitioned table space. The INSERT technique we used here is suitable for small tables only.
C.1 Batch job used to create a backup table space

```sql
/*JOBPARM SYSAFF=SC04,L=9999 */
/* */
/* **************************************************************************/  
/DSN1ST EXEC PGM=IKJEFT01  
/STEPLIB DD DSN=DB7UU.RUNLIB.LOAD,DISP=SHR  
/DD DSN=DB7U7.SDSNLOAD,DISP=SHR  
/SYSTSPRT DD SYSOUT=*  
/SYSPRINT DD SYSOUT=*  
/SYSUDUMP DD SYSOUT=*  
/SYSTFIN DD *  
DSN SYSTEM(D7U1)  
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71)  
END  
/* */
SYSSIN DD *  
SET CURRENT SQLID = 'PRODTBO'  
;
CREATE TABLESPACE H0393999 IN PROD01 USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 FREEPAGE 4 PCTFREE 30 DEFINE NO SEGSIZE 32 BUFFERPOOL BP1 LOCKSIZE PAGE LOCKMAX 0 COMPRESS NO  
;
commit  
;
CREATE TABLE PRODTBO.XS_ACT_EMP (  
  ROW_ID                         VARCHAR(15)          NOT NULL,  
  CREATED                        TIMESTAMP            DEFAULT  NOT NULL,  
  CREATED_BY                     VARCHAR(15)          NOT NULL,  
  LAST_UPD                       TIMESTAMP            DEFAULT  NOT NULL,  
  LAST_UPD_BY                    VARCHAR(15)          NOT NULL,  
  MODIFICATION_NUM               NUMERIC(10,0)        DEFAULT 0 NOT NULL,  
  CONFLICT_ID                    VARCHAR(15)          DEFAULT '0' NOT NULL,  
  ACTIVITY_ID                    VARCHAR(15)          NOT NULL,  
  ACT_ALARM_FLG                  CHAR(1)              DEFAULT 'X' NOT NULL,  
  ACT_APPT_REPT_FLG              CHAR(1)              DEFAULT 'X' NOT NULL,  
  ACT_CAL_DISP_FLG               CHAR(1)              DEFAULT 'X' NOT NULL,  
  ACT_TEMPLATE_FLG               CHAR(1)              DEFAULT 'X' NOT NULL,  
  ASGN_MANL_FLG                  CHAR(1)              DEFAULT 'Y' NOT NULL,
); 
```
null,
EMP_ID            VARCHAR(15)     NOT NULL,
ROW_STATUS        VARCHAR(10)     DEFAULT 'Y' NOT NULL,
NULL,
ASGN_DNRM_FLG     CHAR(1)        DEFAULT 'N',
ASGN_SYS_FLG      CHAR(1)        DEFAULT 'N',
ACT_APPT_RPTEND_DT TIMESTAMP,
ACT_APPT_START_DT TIMESTAMP,
ACT_TODO_PLNEND_DT TIMESTAMP,
ACT_TODO_PLNSTRDT TIMESTAMP,
ACT_EVT_STAT_CD   VARCHAR(30),
EMP_STATUS_CD     VARCHAR(30)) IN PROD01.H0393999
;
commit
;
GRANT SELECT,INSERT,UPDATE,DELETE ON PRODTBO.XS_ACT_EMP TO PRODGRP
;
commit
;
CREATE  INDEX PRODTBO.XS_ACT_EMP_M1 ON PRODTBO.XS_ACT_EMP
(EMP_ID, ACT_TODO_PLNSTRDT DESC, ACT_TODO_PLNEND_DT DESC,
ACT_ALARM_FLG, ACT_CAL_DISP_FLG, ACT_TEMPLATE_FLG,
ACT_APPT_REPT_FLG, ACT_APPT_RPTEND_DT DESC, ACT_APPT_START_DT DESC,
ACT_EVT_STAT_CD)
USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
CLUSTER BUFFERPOOL BP2
;
commit
;
CREATE  INDEX PRODTBO.XS_ACT_EMP_F1 ON PRODTBO.XS_ACT_EMP
( ACTIVITY_ID)
USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
;
commit
;
CREATE  INDEX PRODTBO.XS_ACT_EMP_M3 ON PRODTBO.XS_ACT_EMP
( ROW_STATUS, EMP_ID)
USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
;
commit
;
CREATE UNIQUE INDEX PRODTBO.XS_ACT_EMP_P1 ON PRODTBO.XS_ACT_EMP
( ROW_ID)
USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
;
commit

Appendix C. Batch jobs for table space partitioning  169
CREATE UNIQUE INDEX PRODTBO.XS_ACT_EMP_U1 ON PRODTBO.XS_ACT_EMP
( ACTIVITY_ID, EMP_ID, CONFLICT_ID )
USING STOGROUP PRODSTO PRIQTY 48 SECQTY 1440 PCTFREE 30 DEFINE NO BUFFERPOOL BP2
;
commit
;
INSERT INTO PRODTBO.XS_ACT_EMP ( SELECT * FROM PRODTBO.S_ACT_EMP )
;
COMMIT
;
DROP TABLE PRODTBO.S_ACT_EMP
;
COMMIT
;
DROP TABLESPACE PROD01.H0393000
;
COMMIT
;
//--
PARTITION_COLUMN               CHAR(2)              DEFAULT '  ' NOT NULL,
ROW_ID                         VARCHAR(15)          NOT NULL,
CREATED                        TIMESTAMP            DEFAULT  NOT NULL,
CREATED_BY                     VARCHAR(15)          NOT NULL,
LAST_UPD                       TIMESTAMP            DEFAULT  NOT NULL,
LAST_UPD_BY                    VARCHAR(15)          NOT NULL,
MODIFICATION_NUM               NUMERIC(10,0)        DEFAULT 0 NOT NULL,
CONFLICT_ID                    VARCHAR(15)          DEFAULT '0' NOT NULL,
ACTIVITY_ID                    VARCHAR(15)          NOT NULL,
ACT_ALARM_FLG                  CHAR(1)              DEFAULT 'x' NOT NULL,
ACT_APPT_REPT_FLG              CHAR(1)              DEFAULT 'x' NOT NULL,
ACT_CAL_DISP_FLG               CHAR(1)              DEFAULT 'x' NOT NULL,
ACT_TEMPLATE_FLG               CHAR(1)              DEFAULT 'x' NOT NULL,
ASGN_MANL_FLG                  CHAR(1)              DEFAULT 'Y' NOT NULL,
EMP_ID                         VARCHAR(15)          NOT NULL,
ROW_STATUS                     VARCHAR(10)          DEFAULT 'Y' NOT NULL,
ASGN_DNRM_FLG                  CHAR(1)              DEFAULT 'N',
ASGN_SYS_FLG                   CHAR(1)              DEFAULT 'N',
ACT_APPT_RPTEND_DT             TIMESTAMP,
ACT_APPT_START_DT              TIMESTAMP,
ACT_TODO_PLNEND_DT             TIMESTAMP,
ACT_TODO_PLNSTRDT              TIMESTAMP,
ACT_EVT_STAT_CD                VARCHAR(30),
EMP_STATUS_CD                  VARCHAR(30)) IN PROD01.H0393000
/
CREATE  INDEX PRODTBO.S_ACT_EMP_P99 ON PRODTBO.S_ACT_EMP
(PARTITION_COLUMN, EMP_ID)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO CLUSTER
(PART 1 VALUES ('B') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4 ,
PART 2 VALUES ('C') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4 ,
PART 3 VALUES ('E') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4 ,
PART 4 VALUES ('G') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4 ,
PART 5 VALUES ('K') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4 ,
PCTFREE 30 FREEPAGE 4,
PART 6 VALUES ('N') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4,
PART 7 VALUES ('P') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4,
PART 8 VALUES ('R') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4,
PART 9 VALUES ('T') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4,
PART 10 VALUES ('Z') USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360
PCTFREE 30 FREEPAGE 4
) BUFFERPOOL BP2
/
CREATE TRIGGER PRODTBO.PTH0393 NO CASCADE BEFORE INSERT ON
PRODTBO.S_ACT_EMP REFERENCING NEW AS N FOR EACH ROW MODE DB2SQL
WHEN (N.EMP_ID IS NOT NULL) BEGIN ATOMIC SET N.PARTITION_COLUMN =
RIGHT(N.EMP_ID, 2); END
/
commit
/
GRANT SELECT, INSERT, UPDATE, DELETE ON PRODTBO.S_ACT_EMP TO PRODGRP
/
commit
/
CREATE INDEX PRODTBO.S_ACT_EMP_F1 ON PRODTBO.S_ACT_EMP
(ACTIVITY_ID)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
/
commit
/
CREATE INDEX PRODTBO.S_ACT_EMP_M1 ON PRODTBO.S_ACT_EMP
(EMP_ID, ACT_TODO_PNSTRDT DESC, ACT_TODO_PNEND_DT DESC,
ACT_ALARM_FLG, ACT_CAL_DISP_FLG, ACT_TEMPLATE_FLG,
ACT_APPT_REPT_FLG, ACT_APPT_RPTEND_DT DESC, ACT_APPT_START_DT DESC,
ACT_EVT_STAT_CD)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
/
commit
/
CREATE INDEX PRODTBO.S_ACT_EMP_M3 ON PRODTBO.S_ACT_EMP
(ROW_STATUS, EMP_ID)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
/
commit
/
CREATE UNIQUE INDEX PRODTBO.S_ACT_EMP_P1 ON PRODTBO.S_ACT_EMP
(ROW_ID)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
/
commit
/
CREATE UNIQUE INDEX PRODTBO.S_ACT_EMP_U1 ON PRODTBO.S_ACT_EMP
(ACTIVITY_ID, EMP_ID, CONFLICT_ID)
USING STOGROUP PRODSTO PRIQTY 12 SECQTY 360 PCTFREE 30 DEFINE NO
BUFFERPOOL BP2
/
commit
/
INSERT INTO PRODTBO.S_ACT_EMP (ROW_ID, CREATED, CREATED_BY, LAST_UPD, LAST_UPD_BY, MODIFICATION_NUM, CONFLICT_ID, ACTIVITY_ID, ACT_ALARM_FLG, ACT_APPT_REPT_FLG, ACT_CAL_DISP_FLG, ACT_TEMPLATE_FLG, ASGN_MANL_FLG, EMP_ID, ROW_STATUS, ASGN_DNRM_FLG, ASGN_SYS_FLG, ACT_APPT_RPTEND_DT, ACT_TODO_PNSTRDT, ACT_EVT_STAT_CD, EMP_STATUS_CD)
(SELECT ROW_ID, CREATED, CREATED_BY, LAST_UPD, LAST_UPD_BY, MODIFICATION_NUM, CONFLICT_ID, ACTIVITY_ID, ACT_ALARM_FLG, ACT_APPT_REPT_FLG, ACT_CAL_DISP_FLG, ACT_TEMPLATE_FLG, ASGN_MANL_FLG,
EMP_ID,
ROW_STATUS,
ASGN_DNRM_FLG,
ASGN_SYS_FLG,
ACT_APPT_RPTEND_DT,
ACT_APPT_START_DT,
ACT_TODO_PLNEND_DT,
ACT_TODO_PLNSTRDT,
ACT_EVT_STAT_CD,
EMP_STATUS_CD
FROM PRODTBO.XS_ACT_EMP)
/
COMMIT
/
DROP TABLE PRODTBO.XS_ACT_EMP
/
COMMIT
/
DROP TABLESPACE PROD01.H0393999
/
COMMIT
/
//*
Appendix D. REXX procedures for table space partitioning

This appendix contains the REXX procedures, jobs, and DDL listings of all members required to prepare and execute table space partitioning. During our tests, all those members were installed in a PDS called 'SIEBEL1.XUAN.SIEBEL.PDS'.

D.1 Reference table

The following members relate to the reference table.

Example: D-1  REFTBDDL member

```
CREATE TABLE XATRAN.TAB_PART_LST
(TBNAME CHAR(18) NOT NULL WITH DEFAULT,
 PART_COL CHAR(18) NOT NULL WITH DEFAULT,
 PART_SRC CHAR(18) NOT NULL WITH DEFAULT,
 COL_FUNC CHAR(70) NOT NULL WITH DEFAULT,
 CARD INTEGER NOT NULL WITH DEFAULT,
 IDX_CNT SMALLINT NOT NULL WITH DEFAULT)
IN DATABASE DSNDB04
AUDIT NONE
CCSID EBCDIC
```

© Copyright IBM Corp. 2003. All rights reserved.
Example: D-2  TBL00001 member

```
LOAD DATA INDDN SYSREC   LOG NO RESUME YES
  EBCDIC  CCSID(00500,00000,00000)
INTO TABLE "XATRAN  "."TAB_PART_LST      
WHEN(00001:00002 = X'00CF')
  ( "TBNAME            " POSITION(  00003:00020) CHAR(018)
    , "PART_COL          " POSITION(  00021:00038) CHAR(018)
    , "PART_SRC          " POSITION(  00039:00056) CHAR(018)
    , "COL_FUNC          " POSITION(  00057:00126) CHAR(070)
    , "CARD              " POSITION(  00127:00130) INTEGER
    , "IDX_CNT           " POSITION(  00131:00132) SMALLINT
  )
```

D.2 Unload jobs and procedures

We used the following unload jobs and procedures.

Example: D-3  UNLDJ member

```
//UNLDJ JOB  (999,POK),'IM',REGION=24M,MSGLEVEL=(1,1),CLASS=A,
  //   COND=(4,LT),LINES=(500,WARNING),NOTIFY=&SYSUID
/*JOBPARM S=SC04
  // JCLLIB ORDER=(SIEBEL1.XUAN.SIEBEL.PDS)
  //*------------------------- UPLOAD DATA FROM A LIST OF TABLES --------
  //REFCREXX EXEC ISPBAT
  //ISPFBACH.SYSTSIN DD *
  //ISPSTART CMD(UNLDGEN)
  */
  //SYSSID=D7R1
  //TABLEOWNER=SIEBTBO
  //OUT_HLO=SIEBEL1
  //TABLEINP=XATRAN.TAB_PART_LST
  //BASEPDS=SIEBEL1.JCL
  //DELOADCRDS=Y
/*
Example: D-4 ISPBAT member

```rexx
//ISPBAT PROC
//**ISPBAT PROC DB2VER=DB7U7,MYCLIST=SIEBEL1.XUAN.SIEBEL.PDS,
//** MYMLIB=SYS3.ISPF.ISPMLIB,MYSLIB=SYS3.ISPF.ISPSLIB,
//** MYTLIB=SYS3.ISPF.ISPTLIB
//*******************************************************************************/
//** EXECUTE ISPF IN BATCH - USE YOUR OWN LIBS AS NEEDED. */
//** CREATED 11/21/01 RRR */
//*******************************************************************************/
//** CREATE UNIQUE ISPSPROF. STOPS ENQUEUE FAILURE ON ISPSPROF. */
//*******************************************************************************/
//SPROF EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//INDD DD DISP=SHR,DSN=SIEBEL1.SC04.ISPF42.ISPPROF
//OUTDD DD DSN=&&ISPPROF,
//            DISP=(NEW,PASS),
//            UNIT=SYSDA,
//            SPACE=(TRK,(5,1,5)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=9040)
//SYSIN DD DISP=SHR,DSN=SIEBEL1.XUAN.SIEBEL.PDS(IEBCOPYA)
//*******************************************************************************/
//**- RUN ISPF IN BATCH */
//*******************************************************************************/
//ISPFBACH EXEC PGM=IKJEFT1B,DYNAMNBR=25,REGION=0M
//STEPLIB DD DSN=DB7R7.SDSNLOAD,DISP=SHR  <== DB2 LOADLIB
//         DD DSN=DB7R7.SDSNEXIT,DISP=SHR  <== DB2 EXITLIB
//         DD DSN=DB7RU.RUNLIB.LOAD,DISP=SHR <== DB2 RUNLIB
//*******************************************************************************/
//** ALLOCATE PROFILE,ETC DATASETS
//*******************************************************************************/
//ISPPROF DD DSN=&&ISPPROF,
//            DISP=(OLD,DELETE)
//SYSEXEC DD DISP=SHR,DSN=ISP.SISPEXEC
//ISPEXEC DD DISP=SHR,DSN=ISP.SISPEXEC
//SYSPROC DD DISP=SHR,DSN=SIEBEL1.XUAN.SIEBEL.PDS
//         DD DISP=SHR,DSN=ISP.ISPCLIB
//         DD DISP=SHR,DSN=SYS1.DGTCLIB
//         DD DISP=SHR,DSN=ICQ.ICQCLIB
//         DD DISP=SHR,DSN=DB7R7.SDNSCLST
//         DD DISP=SHR,DSN=ADB.V3R1M0.SADBCLST
//ISPPLIB DD DISP=SHR,DSN=DB7R7.SDSNSPPF
//         DD DISP=SHR,DSN=DB7R7.SDSNPFPE
//         DD DISP=SHR,DSN=ISP.ISPPENU
//         DD DISP=SHR,DSN=ISP.ISFCLIB
//ISPLIB DD DISP=SHR,DSN=SIEBEL1.XUAN.SIEBEL.PDS
//** DD DISP=SHR,DSN=SYS3.ISPF.ISPMLIB
//** DD DISP=SHR,DSN=ISP.ISPMENU
//** DD DISP=SHR,DSN=ICQ.ICQCLIB
```

Appendix D. REXX procedures for table space partitioning 179
// DD DISP=SHR,DSN=DB7R7.SDSNSPFM
//ISPTLIB DD DSN=&TLIB,UNIT=SYSDA,SPACE=(TRK,(1,1,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920),
//          DISP=(NEW,DELETE)
// DD DISP=SHR,DSN=SIEBEL1.XUAN.SIEBEL.PDS
// DD DISP=SHR,DSN=ISP.SISPTENU
// DD DISP=SHR,DSN=ICQ.ICQTABLS
// DD DISP=SHR,DSN=DB7R7.SDSNSPFT
//DSNETBLS DD DISP=SHR,DSN=DB7R7.SDSNSPFT
//ISPSLIB DD DISP=SHR,DSN=SIEBEL1.XUAN.SIEBEL.PDS
// DD DISP=SHR,DSN=ISP.SISPSLIB
// DD DISP=SHR,DSN=DB7R7.SDSNSPFS
//ISPFILE DD DISP=(NEW,DELETE,DELETE),
//          UNIT=SYSALLDA,SPACE=(CYL,(10,10)),
//          DCB=(LRECL=80,BLKSIZE=27920,RECFM=FB),
//          DSN=&ISPFILE
//ISPC10 DD DISP=(NEW,PASS),DSN=&ISPC10,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPC11 DD DISP=(NEW,PASS),DSN=&ISPC11,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPC11 DD DISP=(NEW,PASS),DSN=&ISPC11,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPC11 DD DISP=(NEW,PASS),DSN=&ISPC11,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPC11 DD DISP=(NEW,PASS),DSN=&ISPC11,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPC11 DD DISP=(NEW,PASS),DSN=&ISPC11,
//          SPACE=(CYL,(1,1)),
//          DATAclas=DATAF
//ISPLOG DD DUMMY
//ISPLIST DD DUMMY
//SYSHELP DD DUMMY
//*
//SYSUADS DD DISP=SHR,DSN=SYS1.UADS
//SYSLBC DD DISP=SHR,DSN=SYS1.BRODCAST
//*
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//*
//SYSTSPRT DD SYSOUT=* 
//*******************************************************************************
// OVERRIDE THIS IN JCL FOR INPUT DATASET (REXX EXECs, ETC)
//**SYSTSPRT DD DISP=SHR,DSN=MYLIB.EXEC(PPPKGS)
//*******************************************************************************
//SYSTSPRT DD DUMMY
Appendix D. REXX procedures for table space partitioning

Example: D-5  IEBCOPYA member

COPY INDD=INDD,OUTDD=OUTDD

Example: D-6  UNLDGEN member

/************************ rexx *******************************************/
/* Language : REXX - UNLOAD via DSNTIAUL */
/* ----------------------------------------------------------- */
/* This REXX program reads a SYSIN parm card and interrogates */
/* a DB2 table to unload tables that need to be converted to */
/* partition. Generated jobs will automatically be submitted */
/* */
/* */
/* SYSIN parameters: */
/* */
/* SSID: Source Sub system ID */
/* */
/* TABLEOWNER: Table Owner */
/* */
/* BUILD: BUILD number */
/* */
/* BASEPDS: Template partition dataset */
/* */
/* DELLOADCRDS: Delete existing output loadcard ds flag */
/* */
/* OUT_HLQ: Output high level qualifier, out PDS will be: */
/* */
/* HLQ.build#.UNLDDATA.LOADCRDS */
/* */
/* TABLEINP: DB2 input table that contains list of tables */
/* */
/* to be unloaded plus function to extract data */
/* */
/* for an additional partition column */
/* */
/***************************************************************/
Address ISPEXEC
'CONTROL ERRORS RETURN'

BATCHIN='Y'
SUBB = 'N'
TESTING = 'N'
GOTPARMS = 'N'
DELCRDS = 'N'
CLEANUP = 'N'
IIMPORT = 'X'
IHOST = 'X'
LPAR = 'SC04'
exitrc = 0
address ispexec
/***************************************************************/
/* Get input parameters */
/***************************************************************/
CALL Get_Util_Sysin_Parameters
IF GOTPARMS = 'Y' THEN
DO
SYSD= LOCSSID
SSID=SYSD
TARLOC = 'PM01' /* MAINFRAME CTL TABLES SSID */
CRETR=TABCRTR
PLAN='DSNTIB71'
CSRCT=0
STPCT=0
TBLCT=0
LOPCT=8
OKTODO = 'Y'
NOSTATS=0
ATTRSTRING = ''
ATTIND = -1
S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')

ADDRESS DSNREXX /* ADD HOST CMD ENVIRONMENT */
/* BY DEFAULT SEND COMMANDS OTHER */
/* THAN REXX INSTRUCTIONS TO */
/* DSNREXX */
/* */

SQLSTMT0="SELECT T1.NAME, T1.CARD, T1.DBNAME, T1.TSNAME, "||
  "T2.PART_COL, T2.PART_SRC , T2.COL_FUNC "||
  "FROM "||TABLE_INP||" T2 "||
  " , SYSIBM.SYSTABLES T1 "||
  "WHERE T1.NAME = T2.TBNAME "||
  "AND T1.CREATOR = ''||TABCRTR||'''"

ADDRESS DSNREXX
"CONNECT" SYSID
If SQLCODE < 0 then
call SQLERR
DO
If EXITRC = 0 then
  DO
    DTTM = DATE('W')||', '||DATE()||' 'TIME()
    ADDRESS ISPEEXEC
    CALL Allocate_output_pds
    DBNME = 'NONE'
    TSNME = 'NONE'
    DONEC1 = 'Y'
    DT = DATE('S') /* MAKE DB2 TIMESTAMP */
    PARSE VAR DT I YR 5 MO 7 DA
    JOBTS = YR'-'MO'-'DA'-'||
    TRANSLATE(TIME('L'),':',':')

182  Siebel 7 with DB2 for z/OS: Database Implementation and Administration Guide
TJOBTS = JOBTS
JOBCT=1
JOBNM='J'||substr(IBILD,3,5)||right(JOBCT,2,'0')
TJOBNM = JOBNM
DTTM = DATE('W')||', '|DATE()||' TIME()

address dsnrexx
"EXEC SQL DECLARE C1 CURSOR FOR S1"
"EXEC SQL PREPARE S1 ",
"ATTRIBUTES :ATTRSTRING INDICATOR ",
" :ATTIND FROM :SQLSTMT0"
If SQLCODE < 0 then
  call SQLERR

DTTM = DATE('W')||', '|DATE()||' TIME()
SAY 'OPENING CURSOR C1 TO OBTAIN BUILD LOAD JOBS.' DTTM
address dsnrexx
"EXEC SQL OPEN C1"
OPENRC1=SQLCODE

If OPENRC1 = 0 then
  DO
    DONEC1 = 'N'
    ADDRESS ISPEXEC
      'FTOPEN TEMP'
    DTTM = DATE('W')||', '|DATE()||' TIME()
    SAY 'BUILDING JOB 'JOBNM' FOR BUILD 'T2BILD DTTM
      'FTINCL LOADJOB C'
  END
  DO UNTIL(DONEC1 = 'Y')
    ADDRESS DSNREXX
      "EXEC SQL FETCH C1 INTO ",
      " :TBLNM, :CARD, :DBNAME, :TSNAME, " ||
      " :PARTCOL, :PARTSRC, :COLFUNC"
    If SQLCODE < 0 then
      DO
        DONEC1 = 'Y'
        SAY 'SQLCODE= ' SQLCODE
        if batchin = 'Y' THEN
          exitrc = 12
        ELSE
          EXIT 12
      END
    If CARD = 0 | CARD = -1 then
      DO
        SAY TBLNM ' may be empty or does not has accurate '||
        ' runstat row count = ' CARD
If SQLCODE = 0 then
   DO
      TBLENME=STRIP(TBLNM)
      DBNME  =STRIP(DBNAME)
      TSNME  =STRIP(TSNAME)
      /* Update reference table with current stat */
      Do
         ADDRESS DSNREXX
         "EXEC SQL UPDATE " TABLE_INP ,
         " SET CARD = " CARD ,
         " WHERE TBNAME = 'TBLENME'"
         IF SQLCODE < 0 then
            call SQLERR
         End
         CSRCT=CSRCT+1
         STPCT=STPCT+1
         TBLCT=TBLCT+1
         If STPCT > LOPCT then
            Do
               DT = DATE('S') /* MAKE DB2 TIMESTAMP */
               PARSE VAR DT 1 YR 5 MO 7 DA
               JOBTS = YR'-'MO'-'DA'-'|
               TRANSLATE(TIME('L'),'.',':')
               TJOBTS = JOBTS
               JOBCT=JOBCT+1
               JOBNM='J'||substr(IBILD,3,5)||
               right(JOBCT,2,'0')
               TJOBNM = JOBNM
               SAY 'BUILDING JOB 'JOBNM' FOR BUILD 'T2BILD
               'FTINCL LOADJOBC'
               STPCT=1
            End
            MBRNM='TBL'||right(TBLCT,5,'0')
            STPNO=MBRNM
            DSND=OHLQ||'.'||IBILD||
            '.UNLDDATA.LOADCRDS('MBRNM')'
            LEGEND=OHLQ||'.'||IBILD||
            '.UNLDDATA.LOADCRDS($README')
            DSNC=OHLQ||'.IBILD'.UNLDDATA.'MBRNM'
            If PARTCOL > ' ' then
               CALL Process_Part_Col
            Else
               Do
Appendix D. REXX procedures for table space partitioning

ADDR ISP EXEC
"FTINCL DSNTIAUX"
END
XREF.TBLCT = LEFT(MBRNM,9,' ')||,
   LEFT(TBLENME,71,' ')
END  /* if sqlcode = 0 then */
ELSE
   IF OPENRC1 = 0 then
      DO
         IF CLEANUP = 'N' then
            DO
               ADDRESS ISP EXEC
               'TBADD JOBDD ORDER'
            END
         DONEC1 = 'Y'
         DTTM = DATE('W')||', '||DATE()||' 'TIME()
         SAY 'CLOSING CURSOR C1.' DTTM
         ADDRESS DSNREXX
         "EXECSQL CLOSE C1"
      END
   END /* do until donec1 = y */
ADDRESS ISP EXEC
 'FTCLOSE'
ADDRESS TSO
 "ALLOC FI(OUTDD) DA('"LEGEND"') OLD "
IF RC = 0 then
   DO
      "EXECIO * DISKW OUTDD (STEM XREF. OPEN FINIS"
      "free fi(outdd)"
   END
ADDRESS ISP EXEC
 'VGET (ZTEMPF) SHARED'
say 'ztempf file is: ' ztempf
IF BATCHIN = 'Y' then
   DO
      ADDRESS TSO
      "SUBMIT '"ZTEMPF"'"
      ADDRESS ISP EXEC
   END
ELSE
   "EDIT DATASET( '"ZTEMPF"')"
END
ADDRESS ISP EXEC
ZISPFRC = exitrc          /* set step completion code */
 'VPUT (ZISPFRC) SHARED'

Appendix D. REXX procedures for table space partitioning  185
EXIT EXITRC
/*-----------------------------------------------------------------*/
/*-----------------------------------------------------------------*/
/* Full columns select needs to be done on partition table with */
/* the additional columns.                                    */
/* Get this information from SYSIBM.SYSCOLUMNS                */
/*-----------------------------------------------------------------*/
Process_Part_Col:
/*                                             */
SQLSTMT1="SELECT NAME, COLTYPE, LENGTH "||,
"FROM SYSIBM.SYSCOLUMNS "||,
"WHERE TBCREATOR = 'TABCREATOR' "||,
"AND TBNAME = 'TBNAME' "||,
"ORDER BY COLNO "
address ispexec
"TBCREATE tabcols NOWRITE REPLACE ",
"KEYS (colname) names(coltype, collen, alias)"
colname = strip(COLFUNC)
alias   = PARTCOL
coltype = 'PART'
'TBADD TABCOLS'
address dsnrexx
"EXEC SQL DECLARE C2 CURSOR FOR S2"
"EXEC SQL PREPARE S2 ",
"ATTRIBUTES :ATTRSTRING INDICATOR ",
" :ATTIND FROM :SQLSTMT1"
IF SQLCODE < 0 then
call SQLERR

DTTM = DATE('W') || ', ' || DATE() || ' ' TIME()
colcnt = 0
"EXEC SQL OPEN C2"
OPENRC2=SQLCODE

IF OPENRC2 = 0 THEN
  DONEC2 = 'N'
DO UNTIL(DONEC2 = 'Y')
  ADDRESS DSNREXX
  "EXEC SQL FETCH C2 INTO ",
  ":NAME, :COLTYPE, :LENGTH "
  IF SQLCODE < 0 THEN
    DO
      DONEC2 = 'Y'
      say 'Fetch C2 error'
      call SQLERR
      SAY 'SQLCODE=' ' SQLCODE
      EXIT EXITRC
if batchin = 'Y' THEN
  exitrc = 12
ELSE
  EXIT 12
END
IF SQLCODE = 0 THEN
  DO
    colcnt = colcnt + 1
    colname = STRIP(NAME)
    alias = colname
    collen = STRIP(LENGTH, L, 0)
    coltype = STRIP(COLTYPE)
    Select
      When coltype = 'ROWID',
        colname = 'PARTITION_COLUMN' then
          coltype = 'SKIP'
      Otherwise
        coltype = 'OTHER'
End
  If coltype <> 'SKIP' then
    Do
      ADDRESS ISPEXEC
      'TBADD TABCOLS'
    End
  END
ELSE
  IF OPENRC2 = 0 THEN
    Do
      ADDRESS ISPEXEC
      "FTINCL DSNTIAUI"
      DONEC2 = 'Y'
      DTTM = DATE('W') || ', ' || DATE() || ' ' || TIME()
      ADDRESS DSNREXX
      "EXEC SQL CLOSE C2"
    END
  END /* do until (doneC2 = 'y') */
Return
/*-----------------------------------------------------------------*/
/* Allocate output PDS based on a supply pds model                 */
/*-----------------------------------------------------------------*/
Allocate_output_pds:
/*                                             */
Likedsn = PDSMODEL
newdsn = OHQ."IBILD".UNLDDATA.LOADCRDS"
Address tso
  IF SYSDSN('""|newdsn|""') = "OK" THEN
    DO
IF DECLRDS = 'N' THEN
  DO
    SAY NEWDSN' EXISTS ALREADY. ',
       'TERMINATING UNTIL CORRECTED. '
    EXITRC = 9
  END
ELSE
  DO
    "DELETE '"NEWDSN"'"
  END
END
IF SYSDSN('"'||newdsn||'"') = "DATASET NOT FOUND" THEN
  DO
    "alloc da('"newdsn") like('"likedsn") "
    "free dataset('"newdsn") *
  END
Return
/*-----------------------------------------------------------------*/
Get_Util_Sysin_Parameters:
  /*                                             */
  Address tso
    'EXECIO * DISKR SYSIN (STEM UTIL_SYSIN.'
  If RC > 0 Then
    Do
      Err_Msg = 'Unable to Read Input Parameters'
    End
  Else
    Do
      Sysin_Parms      = ''
      Do Sysin_Rec_Cnt = 1 To Util_Sysin.0
         Sysin_Parms  = Sysin_Parms||,'Strip(Util_Sysin.Sysin_Rec_Cnt,'B')'||'$'
      End
    End
  Call Check_and_Reformat_Parms
  If GOTBILD = 'Y' & GOTPOS = 'Y' &,
     GOTHOST = 'Y' & GOTHGLQ = 'Y' & GOTTABIN = 'Y' then
    GOTPARMS = 'Y'
Else
  GOTPARMS = 'N'
Return
  /*----------------------------------------------------------------*/
Check_and_Reformat_Parms:

Parse Var Sysin_Parms Key_Word '=' Key_Word_Value $$ Sysin_Parms_Rem

SYSIN_CNT = 0
StatsChk = 'OK'
GOTPARMS = 'N'
GOTBILD = 'N'
GOTHOST = 'N'
GOTHQ = 'N'
GOTTABIN = 'N'
BILD = '?'
DELDSN = 'N'
CLEANUP = 'N'
LOC = '?'
ILOC = '?'
IHOST = 'X'
IPORT = 'X'

Do Until Strip(Key_Word_Value,'B') = ''

Chkrmk1 = Pos('/*',Key_Word_Value,1)
If Chkrmk1 > 0 then
    Do
        lenkwv = Length(Key_Word_Value)
        Chkrmk2 = Pos('*/',Key_Word_Value,Chkrmk1)
        If Chkrmk2 > 0 then
            Do
                TrlLen = lenkwv - Chkrmk2 - 1
                Chkrmk2 = Chkrmk2 + 2
                Key_word_Value1 = Substr(Key_Word_Value,1,Chkrmk1-1)
                Key_word_Value2 = Substr(Key_Word_Value,Chkrmk2,TrlLen)
                Key_word_Value = Key_word_Value1||Key_word_Value2
            End
        Else
            Do
                Key_word_Value1 = Substr(Key_Word_Value,1,Chkrmk1-1)
                Key_word_Value = Key_word_Value1
            End
        End
    End Select
When Key_Word = 'BUILD' then
    Do
        BILD = Strip(Key_Word_Value,'B')
        IBILD = BILD
        T2BILD = BILD
        GOTBILD = 'Y'
    End
When Key_Word = 'TABLEOWNER' then
  Do
    TABCRTR = Strip(Key_Word_Value,'B')
    IOWNR   = TABCRTR
  End

When Key_Word = 'DELOADCRDS' then
  Do
    DELCRDS = Strip(Key_Word_Value,'B')
  End

When Key_Word = 'SSID' then
  Do
    LOCSSID = Strip(Key_Word_Value,'B')
    ILOC   = LOC
    GOTHOST = 'Y'
  End

When Key_Word = 'BASEPDS' then
  Do
    PDSMODEL = Strip(Key_Word_Value,'B')
    GOTPDS  = 'Y'
  End

When Key_Word = 'OUT_HLQ' then
  Do
    OHLQ  = Strip(Key_Word_Value,'B')
    GOTHLQ = 'Y'
  End

When Key_Word = 'TABLEINP' then
  Do
    TABLE_INP = Strip(Key_Word_Value,'B')
    GOTTABIN = 'Y'
  End

OTHERWISE
  Do
    Err_Msg = 'Invalid Parameter 'Key_Word
    StatsChk = 'ERR'
  End
End

Parse Var Sysin_Parms_Rem ,
  Key_Word = 'Key_Word_Value $'$ Sysin_Parms_Rem
End
Appendix D. REXX procedures for table space partitioning

Call Display_Parameter

Return

/*---------------------------------------------------------------------*/
Display_Parameter:

Say '#'Copies('-',78)'
Say '#'left('UNLD Parameters',78)''
Say '#'Copies('-',78)'
Say '#'left('BUILD      ='IBILD,78)'
Say '#'left('SSID       ='LOCSSID,78)'
Say '#'left('TABLEOWNER ='TABCRTR,78)'
Say '#'left('DELLOADCRDS='DELCRDS,78)'
Say '#'left('OUT_HLQ    ='OHLQ,78)'
Say '#'left('TABLEINP   ='TABLE_INP,78)'
Say '#'left('BASEPDS    ='PDSMODEL,78)'
Say '#'Copies('-',78)'

Return

SQLERR:
trace('o')
SAY "-------------------- SQL ERROR -----------------------"
SAY "SQLSTATE="SQLSTATE
SAY "SQLWARN ="SQLWARN.0
SAY SQLWARN.1"," 
SAY SQLWARN.2","
SAY SQLWARN.3","
SAY SQLWARN.4","
SAY SQLWARN.5","
SAY SQLWARN.6","
SAY SQLWARN.7","
SAY SQLWARN.8","
SAY SQLWARN.9","
SAY SQLWARN.10","
SAY "SQLERRD="SQLERRD.1
SAY SQLERRD.2","
SAY SQLERRD.3","
SAY SQLERRD.4","
SAY SQLERRD.5","
SAY SQLERRD.6"," 
SAY "SQLERRP="SQLERRP
SAY "SQLERRMC="SQLERRMC
SAY "SQLCODE="SQLCODE
SELECT
WHEN SQLCODE="-198" THEN
   SAY ">>OBJECT OF PREPARE (SQL STATEMENT) IS INVALID."
WHEN SQLCODE="-204" THEN
```sql
SAY ">>UNDEFINED NAME SPECIFIED:"SQLERRMC
WHEN SQLCODE='-923' THEN
  SAY ">>COMMS CONNECTION TO "SSID" COULD NOT BE ESTABLISHED."
WHEN SQLCODE='-924' THEN
  SAY ">>SUBSYSTEM IS NOT ACTIVE."
WHEN SQLCODE='-991' THEN
  SAY ">>SUBSYSTEM IS NOT AVAILABLE. UNABLE TO CONNECT."
END
END
if batchin = 'Y' THEN
  DO
    exitrc = 12
    RETURN
  END
ELSE
  EXIT 12
END

Example: D-7  DSNTIAUI member

)SEL &DELSN = Y
  //DELSN EXEC PGM=IEFBR14
  //DD1 DD DISP=(MOD,DELETE,DELETE),SPACE=(TRK,(1)),UNIT=SYSDA,
    // DSN=&DSNC
)ENDSEL
)SEL &CLEANUP = N
  // JCLLIB ORDER=(SIEBEL1.XUAN.SIEBEL.PDS)
  // &MBRNM. EXEC TBLUNLD
  //&JCLLIB ORDER=(SIEBEL1.XUAN.SIEBEL.PDS)
  // &MBRNM. EXEC TBLUNLD
  //******************************************************************
  // UNLOAD FROM DB=&DBNME TS=&TSNME TBL=&TBLENME
  // TO FILE &DSNC
  // TO FILE &DSNC
  //* LOADCARDS IN &DSND
  //* LOADCARDS IN &DSND
  //******************************************************************
  //DSNTIAUL.SYPUNCH DD DISP=SHR,
    // DSN=&DSNC
  //DSNTIAUL.SYSPUNCH DD DISP=SHR,
    // DSN=&DSNC
  //DSNTIAUL.SYSRECOO DD DISP=(NEW,CATLG,DELETE),
    // DSN=&DSNC,
    // SPACE=(TRK,(1000,1000),RLSE)
  // DATACLAS=COMPACT
  //DSNTIAUL.SYSTSIN DD *
  //DSN SYSTEM(&SSID)
  //DSN SYSTEM(&SSID)
    RUN PROGRAM(DSNTIAUL) PLAN(&PLAN) PARM('SQL') -
      LIB('DB7RJ.RUNLIB.LOAD')
  */
  //SYSIN DD *
)TBA 10 47
)SET A = SELECT
```
Example: D-8  DSNTIAUX member

```sql
*/
//&MBRNM.  EXEC TBLUNLD
//--*------------------------------------------------------------------
//--*      UNLOAD FROM DB=&DBNME TS=&TSNME TBL=&TBLENME
//--*      TO FILE &DSNC
//--*      LOADCARDS IN &DSND
//--*------------------------------------------------------------------
//--*DSNTIAUL.SYSPUNCH DD DISP=SHR,
//--*     DSN=&DSND
//--*DSNTIAUL.SYSREC00 DD DISP=(NEW,CATLG,DELETE),
//--*     DSN=&DSNC,
//--*     SPACE=(TRK,(1000,1000),RLSE),DATACLAS=COMPACT
//--*DSNTIAUL.SYSTSIN  DD  *
---DSN SYSTEM(&SSID)
RUN  PROGRAM(DSNTIAUL) PLAN(&PLAN)  -
     LIB('DB7RU.RUNLIB.LOAD')
/
/*
//SYSIN  DD *
 &TABCNR..&TBLENME. ;
)ENDSEL
/*
Example: D-9  TBLUNLD member

```
Example: D-10  LOADJOBC member

Example: D-11  LOADGENJ member

D.3 Load jobs and procedures

We used the following load jobs and procedures.

Example: D-10  LOADJOBC member

Example: D-11  LOADGENJ member

//DSNTIAUL  EXEC PGM=IKJEFT1B,REGION=4M
//*
//STEPLIB  DD DISP=SHR,DSN=&SDSNEXIT
//   DD DISP=SHR,DSN=&SDSNLOAD
//   DD DISP=SHR,DSN=&RUNLIB
//*
//SYSPRINT DD  SYSOUT=*  
//SYSTSPRT DD  SYSOUT=*  
//SYSUDUMP DD  SYSOUT=*  
//SYSOUT  DD  SYSOUT=*  
//SYSPUNCH DD  DUMMY  
//SYSRECOO DD  DUMMY  
//SYSTSIN  DD  DUMMY

Example: D-10  LOADJOBC member

//&JOBNM JOB  (ACCOUNT),'IM',REGION=48M,MSGLEVEL=(1,1),CLASS=A,  
// LINES=(500,WARNING)
/*JOBPARM S=&LPAR

D.3 Load jobs and procedures

We used the following load jobs and procedures.

Example: D-11  LOADGENJ member

//LOADGENJ JOB  (ACCOUNT),'IM',REGION=24M,MSGLEVEL=(1,1),CLASS=A,  
// COND=(4,LT),LINES=(500,WARNING),NOTIFY=&SYSUID
/*JOBPARM S=SC04
// JCLLIB ORDER=(SIEBEL1.XUAN.SIEBEL.PDS)
//*------------------------------------ GENERATE LOAD JOBS -----------------
//REFCREXX EXEC ISPBAT
//ISPFBACH.SYSTSIN DD *
//ISPSTART CMD(LOADGEN)
/*
//SYSIN DD *
BUILD=SP19054
SSID=D7R1
TABLEOWNER=SIEBTBO
MACRO_NAME=CARDEDIT
TABLEINP=XATRAN.TAB_PART_LST
LOADPDS=SIEBEL1.SP15054.UNLDDATA.LOADCRDS
DELOUTPDS=Y
/

Example: D-12  LOADGEN member

/*************************************************************/
/* Language : REXX - LOADGEN */
/* ----------------------------------------------- */
/* This REXX program reads a list of tables and generate */
/* LOAD REPLACE jobs, one for each table. */
/* */
/* The program will first allocate and copy all load cards */
/* to a PDS with the SSID as the last node */
/* */
/* The generated load jobs will be put in member JCL2LOAD in */
/* the same library as the copied loadcard PDS */
/* */
/* */
/* SYSIN parameters: */
/* */
/* SSID: Source Sub system ID */
/* TABLEOWNER: Table Owner */
/* BUILD: BUILD number */
/* LOADCRD: Template partition dataset PDS */
/* DELOUTPDS: Delete any existing output loadcard ds flag */
/* MACRO_NAME: Edit macro to edit load card */
/* TABLEINP: DB2 input table that contains list of tables */
/* to be unloaded plus function to extract data */
/* for an additional partition column */
/* */
/*************************************************************/
Address ISPEXEC
'CONTROL ERRORS RETURN'
DTTM = DATE('W')||', '||DATE()||' 'TIME()

/-----------------------------------------------*/
/* INITParms */
/-----------------------------------------------*/
/* trace('i') */
JOBCT=0
lopCT=0
CSRCT=0
TOTCRT=0
BILD = 'B15051'
DELCRDS = 'N'
Initiator_Class_Big = 'S'
Initiator_Class_Small = 'A'
lpar = 'SC04'

Call Get_Util_Sysin_Parameters

Last_node_pos = LASTPOS('.',LDCard)
Pds_i = "LDCard"

CALL Allocate_output_pds

Legend=out_dsn||"($README)"

/*--------------------------------------------------------------*/
/* setup dB2 addressability.                                  */
/*--------------------------------------------------------------*/
Address TSO "SUBCOM DSNREXX"
if rc then
   S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
else
   say rc || 'dsnrexx command env not available'

/*--------------------------------------------------------------*/
/* Update reference tables with index count number            */
/*--------------------------------------------------------------*/
Address DSNREXX
"CONNECT "SSID
If rc <> 0 then
   Do
      SAY '+*********************************************+'
      SAY '+*    CONNECTION TO DB2 FAILED             *+'
      SAY '+*      RETURN CODE IS ' RC '           *+'
      SAY '+*********************************************+'
      EXIT RC = RC
   End

"EXECSQL UPDATE " TABLE_INP " TB ", " SET IDX_CNT = " , " ( SELECT COUNT(*) ) " , " FROM SYSIBM.SYSINDEXES IX ", " WHERE CREATOR = "CREATR" " , " AND TB.TBNAME = IX.TBNAME " 

IF SQLCODE < 0 then call SQLERR

/*--------------------------------------------------------------*/
/* get a total count of indexes to work out sortkeys          */
/*--------------------------------------------------------------*/
OPENRC0=SQLCODE
IF OPENRC0 = 0 THEN
DO
  X1= TIME('E')
END

/*------------------------------------------------------------------------*/
/* get cross reference of member names and table names used to */
/* build the LOAD JCL. cross ref is in legend. */
/* Load it into an ispf table so lookups can be done later. */
/*------------------------------------------------------------------------*/

ADDRESS TSO
"ALLOC FI(INDD) DA("LEGEND") SHR "
ALLOCRC= RC
If ALLOCRC = 0 Then
DO
  "EXECIO * DISKR INDD (STEM XREF. OPEN FINIS"
  "free fi(indd) "
END

ADDRESS TSO
"FREE FI(ISPFILE) "
ADDRESS ISPEXEC
'LIBDEF ISPFILE DATASET ID('Pds_o')'
'FTOPEN '
DO i = 1 to xref.0
  PARSE VAR xref.i MBRNM TBLENME .
  sqlstmt= ,
    "SELECT CARD, IDX_CNT",
    "FROM " TABLE_INP,
    "WHERE TBNAME = "tblenme" "
Address DSNREXX
"EXECSQL DECLARE C1 CURSOR FOR S1"
"EXECSQL PREPARE S1 FROM :sqlstmt"
"EXECSQL OPEN C1"
"EXECSQL FETCH C1 INTO ",
  ":card, :ixcnt "
  If SQLCODE = 0 | SQLCODE = -305 then
    Do
      ixcnt = strip(ixcnt)
      tablbl= STRIP(mbrnm)
      PARSE VAR tablbl 'TBL' nbr
      TBNM = strip(TBLENME)
      jobname = SSID||RIGHT(nbr,4,'0')
      if ixcnt > 0 then
        sortkeys = ixcnt * card
      else
        sortkeys = card
      address ispexec "VPUT (CREATR sortkeys tbnm) SHARED"
    End
if sortkeys > 2000000 then
    Do
        Class = Initiator_Class_Big
        Rgn   = '0M'
    End
Else
    Do
        Class = Initiator_Class_Small
        Rgn   = '256M'
    End
end

ADDRESS ISPEXEC
"EDIT DATASET("EditDS") MACRO("MACNAME")"
"FTINCL LOADSKEL"
CSRCT = CSRCT + 1
End
ADDRESS DSNREXX
"EXECSQL CLOSE C1"
End

ADDRESS ispexec
'FTCLOSE NAME(JCL2LOAD)'
say "process: "  CSRCT " members"
ADDRESS ISPEXEC
X2= TIME('E')
/* trace('i') */
SAY 'IT TOOK 'X2 'SECONDS.'
ADDRESS TSO
"FREE FI(ISPFILE) "
END
Exit

/*-------------------------------------------------------------*/
/* Allocate output PDS based on a supply pds model            */
/*-------------------------------------------------------------*/
Allocate_output_pds:
/*
*/

newdsn= substr(LDCard,1,Last_node_pos) || SSID
out_dsn = newdsn
Pds_o = ""newdsn"
Address tso
IF SYSDSN(""||newdsn||""") = "OK" THEN
    DO
        IF DELCRDS = 'N' THEN
            DO

198  Siebel 7 with DB2 for z/OS: Database Implementation and Administration Guide
Appendix D. REXX procedures for table space partitioning

SAY 'NEWDSN' EXISTS ALREADY. ',
   'TERMINATING UNTIL CORRECTED.'
EXITRC = 9
END
ELSE
   DO
      "DELETE '"NEWDSN"'
   END
END
IF SYSDSN('"'||newdsn||'"') = "DATASET NOT FOUND" THEN
   DO
      "alloc da('"newdsn") like('"LDCard")"
      "free dataset('"newdsn")"
   END
ADDRESS ISPEXEC
"LMINIT DATAID(FDID) DATASET("PDS_i") ENQ(SHR)"
"LMINIT DATAID(TDID) DATASET("PDS_o") ENQ(EXCLU)"
IF RC <> 0 THEN
   DO
      SAY 'UNABLE TO GET EXCLU ENQUEUE ON 'PDS_o
      SAY 'LOADGEN WILL TERMINATE NOW.'
      EXIT 11
   END
"LMCOPY FROMID("FDID") FROMMEM(*)",
   " TODATAID("TDID") REPLACE "
IF RC <> 0 THEN
   DO
      SAY 'Unable to copy loadcards to output pds' PDS_o
      SAY 'LOADGEN WILL TERMINATE NOW.'
      EXIT 12
   END
Return

/*-----------------------------------------------------------------*/
Get_Util_Sysin_Parameters:
/*                                             */
Address tso
'EXECIO * DISKR SYSIN (STEM UTIL_SYSIN.'
   /*
If RC > 0 Then
   Do
      Err_Msg = 'Unable to Read Input Parameters'
   End
Else
   Do
      SysinParms = ''
Do Sysin_Rec_Cnt = 1 To Util_Sysin.0
Sysin_Parms = Sysin_Parms||,
Strip(Util_Sysin.Sysin_Rec_Cnt,'B')||'$$'
End
End

Call Check_and_Reformat_Parms

If GOTBILD = 'Y' & GOTPOS = 'Y' & GOTMACNM = 'Y' &,
GOTHOST = 'Y' & GOTTABIN = 'Y' then
GOTPARMS = 'Y'
Else
GOTPARMS = 'N'

Return

/*------------------------------------------------------------------*/

Check_and_Reformat_Parms:

Parse Var Sysin_Parms Key_Word '=' Key_Word_Value '$$' Sysin_Parms_Rem

GOTBILD = 'N'
GOTCREATR = 'N'
GOTSSID = 'N'
GOTLOAD = 'N'
GOTRUN = 'N'
GOTMACNM = 'N'
GOTPARMS = 'N'
SYSIN_CNT = 0
StatsChk = 'OK'
BILD = '?'
LOC = '?'

Do Until Strip(Key_Word_Value,'B') = ''

Chkrmk1 = Pos('/*',Key_Word_Value,1)
If Chkrmk1 > 0 then
Do
lenkwv = Length(Key_Word_Value)
Chkrmk2 = Pos('*/',Key_Word_Value,Chkrmk1)
If Chkrmk2 > 0 then
Do
Trllen = lenkwv - Chkrmk2 - 1
Chkrmk2 = Chkrmk2 + 2
Key_word_Value1 = Substr(Key_Word_Value,1,Chkrmk1-1)
Key_word_Value2 = Substr(Key_Word_Value,Chkrmk2,Trllen)
Key_word_Value = Key_word_Value1||Key_word_Value2
End
Else
End
Do
    Key_word_Value1 = Substr(Key_Word_Value,1,Chkrmk1-1)
    Key_word_Value = Key_word_Value1
End
End

Select
When Key_Word = 'BUILD' then
    Do
        BILD        = Strip(Key_Word_Value,'B')
        IBILD      = BILD
        T2BILD     = BILD
        GOTBILD    = 'Y'
    End
When Key_Word = 'TABLEOWNER' then
    Do
        TABCRTR = Strip(Key_Word_Value,'B')
        CREATR  = TABCRTR
    End
When Key_Word = 'MACRO_NAME' then
    Do
        MACNAME  = Strip(Key_Word_Value,'B')
        GOTMACNM = 'Y'
    End
When Key_Word = 'SSID' then
    Do
        LOCSSID  = Strip(Key_Word_Value,'B')
        SSID  = LOCSSID
        GOTHOST    = 'Y'
    End
When Key_Word = 'LOADPDS' then
    Do
        LDCard  = Strip(Key_Word_Value,'B')
        GOTPDS  = 'Y'
    End
When Key_Word = 'TABLEINP' then
    Do
        TABLE_INP = Strip(Key_Word_Value,'B')
        GOTTABIN  = 'Y'
    End
When Key_Word = 'DELOUTPDS' then
    Do
        DEL_T2  = Strip(Key_Word_Value,'B')
        GOTTABIN  = 'Y'
    End
DELCRDS  = Strip(Key_Word_Value,'B')
End

OTHERWISE
Do
   Err_Msg  = 'Invalid Parameter 'Key_Word
   StatsChk = 'ERR'
End
End

Parse Var Sysin_Parms_Rem,
   Key_Word = Key_Word_Value '$$' Sysin_Parms_Rem
End

Call Display_Parameter
Return

/****************************************************************************
Display_Parameter:
Say '#Ccopies(-,78)'#
Say '#lleft('LOADGEN Parameters','78)'#'
Say '#Ccopies(-,78)'#'
Say '#lleft('--> BUILD   ='BILD,'78)'#'
Say '#lleft('--> CREATOR =''CREATR,'78)'#'
Say '#lleft('--> SSID    =''SSID,'78)'#'
Say '#lleft('--> MACRO_NAME='MACNAME,'78)'#'
Say '#lleft('--> TABLEINP =''TABLE_INP,'78)'#'
Say '#lleft('--> DELOUTPDS=''DELCRDS,'78)'#'
Say '#lleft('--> LOADPDS =''LDCard,'78)'#'
Say '#Ccopies(-,78)'#'
Return

/*****************************************************************************/
SQLERR:
trace('o')
SAY "------------------- SQL ERROR -------------------"
SAY "SQLSTATE="SQLSTATE
SAY "SQLWARN ="SQLWARN.0
SAY SQLNARN.1",
SAY SQLNARN.2",
SAY SQLNARN.3",
SAY SQLNARN.4",
SAY SQLNARN.5",
SAY SQLNARN.6",
SAY SQLNARN.7",
SAY SQLNARN.8",
SAY SQLNARN.9",

202   Siebel 7 with DB2 for z/OS: Database Implementation and Administration Guide
Appendix D. REXX procedures for table space partitioning

Example: D-13  LOADSKEL member

FILE: D-13.LOADSKEL

//&JOBNAME  JOB ACCNT#,CLASS=&CLASS.,MSGCLASS=X,MSGLEVEL=(1,1),
//      TIME=1440,REGION=&RGN.,LINES=(500,WARNING),COND=(4,LT),
//      NOTIFY=&&SYSUID
//JOBPARM  S=LPAR.
//JOBLIB  DD DISP=SHR,DSN=DB7R7.SDSNLOAD
//-------------------------------------------------------------------
// LOAD TABLE &TBNM.   TABLE #: &TABLBL.
//-------------------------------------------------------------------
//STP01  EXEC PGM=DSNUTILB,REGION=0M,PARM='&SSID.,&JOBNAME.LD'
//STEP1   DD DSN=DB7R7.SDSNLOAD,DISP=SHR
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//UTPRINT  DD SYSPUT=*
//SYSOUT  DD SYSPUT=*
//SYSIN    DD DISP=SHR,
//           DSN=&DSYSIN.
//SYSREC00 DD DISP=SHR,
//           DSN=&DSYSREC.
//SYSUT1   DD DSN=&ZUSER..&SSID..&TABLBL..SYSUT1,
//           DISP=(MOD,DELETE,CATLG),UNIT=(SYSDA,7),
//           SPACE=(CYL,(500,500),RLSE)
//SORTOUT  DD DSN=&ZUSER..&SSID..&TABLBL..SORTOUT,
//           DISP=(MOD,DELETE,CATLG),UNIT=(SYSDA,7),
//           SPACE=(CYL,(500,500),RLSE)
//-------------------------------------------------------------------
// EOM
/* REXX 2 */
ADDRESS ISPEXEC
'ISREDIT MACRO '
"ISREDIT FIND ' REPLACE NOCOPYPEND ' FIRST "
/* IF NOT edit BEFORE */
If rc <> 0 then
  Do  
    "ISREDIT DELETE 1 2 " /* delete the first 2 lines */
    "ISREDIT FIND ' PARTITION_COLUMN ' FIRST "
    If rc = 0 then
      Do  
        "ISREDIT (Partln) = LINENUM .ZCSR"
        "ISREDIT FIND ' NULLIF( 3)= ' FIRST "
        If rc = 0 then /* if loadcard contains NULLIF on part col */
          Do  
            "ISREDIT (Curln) = LINENUM .ZCSR"
            If (Curln = Partln + 2) then
              "ISREDIT DELETE " Curln /* delete unwanted NULLIF line */
            End
          End
    End
  End
END
LINE1 = 'LOAD DATA LOG NO STATISTICS TABLE(ALL) REPLACE NOCOPYPEND '
ADDRESS ISPEXEC "VGET (CREATR SORTKEYS TBNM) SHARED"
Line2 = 'SORTKEYS ' Sortkeys ' SORTDEVT SYSDA '
Line3 = CREATR."TBNM"
"ISREDIT LINE_BEFORE .ZF = '' Line1 ""
"ISREDIT LINE_AFTER .ZF = '' Line2 ""
"ISREDIT LINE_AFTER 2 = ' INDDN SYSREC00 INTO TABLE ' 
"ISREDIT LINE_AFTER 3 = '' Line3 ""
"ISREDIT END "
End
EXIT 0
Appendix E. REXX program for EXPLAIN

This appendix contains a REXX program that reads the Siebel Object Manager SQL trace and generates the EXPLAIN statements into an input file which will be read and executed by SPUFI or DSNTEP2 utilities. We include also the sample job that is to call the REXX program.

Refer to Chapter 7, “Monitoring dynamic SQL” on page 105, for information on how to use this REXX program.

Note: The REXX program and the sample job to call it can be downloaded from the Web. Instructions on how to download them can be found in Appendix F, “Additional material” on page 211.

Example 8-2  eimrexx REXX program

/*  rexx  get sql from eim log and explain it       */
/*  call x'05' ' ' in the input file              */
/*                                              */
/*  ODBC Functions like { ts '2002-10-15 12:21:06' }  */
/*  need to be replaced with                     */
/*  TIMESTAMP('2002-10-15-12.21.06')             */
/*  in order for explain to execute correctly    */
/*                                              */
/*  M.Andrade 2001 Siebel Systems UK            */
/*                                              */
arg eimdts
timestart = time()
eimdts = "#eimdts"

say "Starting processing... " eimdts
say ''
found = sysdsn(eimdts)
if found ^= 'OK' then
   Call NOGO

"alloc f(eimin) DA("eimdts") SHR"
"execio * diskr eimin (stem record. finis"
"Free f(eimin)"

eimout="sysvar('SYSUID').eim.out"
x=sysdsn(eimout)
if x ^= 'OK' then
   do
      "alloc f(eimout)",
      "da("sysvar('SYSUID').eim.out) mod reuse space(10,50)",
      "tracks dsorg(ps) recfm(f b)",
      "lrecl(80) release"
      "free f(eimout)"
   end
else nop
   "alloc f(eimout) da("sysvar('SYSUID').eim.out) old reuse"

explnum = 0

Do i = 1 to record.0
   line = record.i

   If Word(line,6) = 'UPDATE'
      then
         Do
            Call ProcessSql
            Say ''
            Say ''
         End
   ;
   If Word(line,6) = 'INSERT'
      then
         Do
            Call ProcessSql
            Say ''
            Say ''
         End
   ;
If Word(line,6) = 'SELECT'
    then
        Do
            Call ProcessSql
            Say ' '
            Say ' '
        End
    ;
If Word(line,6) = 'DELETE'
    then
        Do
            Call ProcessSql
            Say ' '
            Say ' '
        End
    ;
End

"Execio" Queued() "Diskw eimout (FINIS"
"Free f(eimout)"

timefins = time()
say "Total Explains : " explnum
say " =
say "Start  :" timestart
say "Finish :" timefins

exit 0

/* Sub Routines */
ProcessSql:

explnum = explnum + 1

queue 'DELETE FROM PLAN_TABLE WHERE QUERYNO = ' explnum
queue '/'
queue 'COMMIT'
queue '/'
queue 'DELETE FROM DSN_STATEMNT_TABLE WHERE QUERYNO = ' explnum
queue '/'
queue 'COMMIT'
queue '/'
queue 'EXPLAIN ALL SET QUERYNO = ' explnum ' FOR '

data = ''
Do while words(line) ^= 5
    charcount = length(line)
    if wordpos('?1:',line) = 0 then
        Do
            ...
data = substr(line,31,charcount-30)
Call ProcessLine
End
else nop
i = i + 1
line = record.i
End

queue '/'
queue 'COMMIT'
queue '/'
queue '
return

ProcessLine:
numWords = words(data)
StringToPrint = ''
EndPos = 0
Do ForLoop = 1 to NumWords
    StringToPrint = StringToPrint || word(data,ForLoop) || ' '
    EndPos = EndPos + 1
    If EndPos = 5 then
        do
            queue StringToPrint
            EndPos = 0
            StringToPrint = ''
        end
    else nop
end

If EndPos > 0 then
    do
        queue StringToPrint
        EndPos = 0
        StringToPrint = ''
    end
else nop
return

NOGO:
Say "EIM Log dataset "eimdts" does not exist"
exit 8
Example 8-3  sample job to call the eimrexx REXX program

//EIMREXJ JOB REGION=0M,
// MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=ANY
//******************************************************************************
//******************************************************************************
//SUBMIT   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DO SYSOUT=*   
//SYSPRINT DO SYSOUT=*      
//SYSUDUMP DO SYSOUT=*      
//SYSTSIN DO *              
    EX 'SIEBEL.REXX.CNTL(EIMREXX)' 'SIEBEL.EIM.LOG'
    */
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/SG246868

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, SG246868.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition.zip</td>
<td>Zipped code samples of table partitioning method</td>
</tr>
</tbody>
</table>
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.
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 214.

- *Implementing Siebel eBusiness Applications with DB2 UDB on AIX/NT,* SG24-6211
- *Siebel 2000 Database Implementation on OS/390 Using NT Siebel Servers,* SG24-5953
- *Migrating Siebel Database from DB2/Oracle for NT to DB2 for OS/390,* SG24-6236
- *Squeezing the Most Out of Dynamic SQL with DB2 for z/OS and OS/390,* SG24-6418
- *DB2 for z/OS and OS/390 Tools for Performance Management,* SG24-6508
- *Implementing ESS Copy Services on S/390,* SG24-5680

Other resources

These publications are also relevant as further information sources:

- *Implementing Siebel eBusiness Applications on DB2 UDB for OS/390 and z/OS, Version 7.0.4,* April 2002 (product publication)
- *DB2 UDB for z/OS and OS/390: Installation Guide,* Version 7, GC26-9936
- *DB2 UDB for z/OS and OS/390 V7 Administration Guide,* Version 7, SC26-9931
- *DB2 UDB for z/OS and OS/390 V7 Utility Guide and Reference,* Version 7, SC26-9945
- *DB2 UDB for OS/390 and z/OS V7 Release Planning Guide,* Version 7, SC26-9943
IBM DB2 Connect Enterprise Edition for UNIX: Quick Beginnings, Version 7, GC09-2952


IBM DB2 Connect: Connectivity Supplement, Version 7, SDB2-CONN-SU, available at:
www.ibm.com/software/data/dbe/library/

Dropping Indexes Process, Siebel Technical Services, March 4, 2002. The Siebel site is:
www.siebel.com

Siebel 7 on DB2 UDB for OS/390 and z/OS V7: Scalability Characterization, White Paper, July 11, 2002

Referenced Web sites

These Web sites are also relevant as further information sources:

Samba
http://www.bullfreeware.com

IBM HTTP Server for Siebel installations

DB2 bibliography

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

A
AIX configuration 29
application server 43
architecture 1–2
asynchronous preformatting 8

B
buffer pools 28, 125

C
catalog row-level locking 8
compression dictionary 129
connection concentrator 12, 18
connection pooling 18, 39
COPY 129
correlated subquery 5

d
database
configuration 69
connectivity
DB2 Connect 10
Siebel connection pooling 18
implementation 3
objects 122
server 53
DB2 126
authorization 25
functions that benefit Siebel 4
asynchronous preformatting 7
catalog row-level locking 8
correlated subquery 5
FETCH FIRST n ROWS 7
monitoring indicators 10
online system parameters 6
realtime statistics 7
REORG enhancements 8
scrollable cursors 6
self-referencing subselect 5
statistics HISTORY 9
sysplex MEMBER ID 6
setup 26
buffer pool 28
databases 28
DSNZPARM 26
indexes 28
storage group 28
stored procedures 29
table spaces 28
tables 28
statistics 126
DB2 Connect 2, 13, 33, 37
configuration 39
connections 10
connection concentrator 12, 18
dedicated 11, 14
sysplex-aware 12, 16
Fix Pack 37
DB2 Performance Monitor for OS/390 105
DB2 PM 105
DB2 PM Workstation Online Monitor 106
dedicated connections 11, 14
dev2prod 76
DSC 106
DSNZPARM 26
dynamic SQL
dynamic statement cache 106
monitoring 105
dynamic statement cache 106

E
EDM pool 124
EIM 91
EXPLAIN 109, 118, 205

F
FETCH FIRST n ROWS 7
free space 124

G
gateway server 40

H
history statistics 127
schema 55, 67, 72, 74
schema.sql file 72, 74, 85
scrollable cursors 6
security 24
  DB2 authorization 25
  RACF 25
self-referencing subselect 5
Siebel
  connection pooling 18, 39
  database configuration utility 69
  EIM 91
  maintenance 55
  repository 78, 85
  schema 55, 67, 72, 74
  scripts 53
  servers 40
    application 43
    database 53, 55
    gateway 40
    Web 62, 66
SQL tracing 114
storage control file 68–69, 74
system configuration
  client tier 22
  database tier 23
  mid-tier 23
statistics history 10
storage control file 68–69, 74
storage group 28
stored procedures 29
sysplex MEMBER ID 6
sysplex-aware 11
sysplex-aware connections 12, 16

T
transaction pooling 12, 18

V
VSAM data set extents 126

W
Web server 62, 66
In this IBM Redbook, we describe the implementation, customization, and administration of the Siebel 7 database with DB2 UDB for z/OS and OS/390 V7 using AIX V5L or Win2000 Siebel Servers, and we discuss the following topics.

- Installation tasks for setting up a three-tier configuration, with the application middle tier residing on AIX/Win2000 and the database tier residing on z/OS. The DB2 Connect EE V7 setup required to connect the Siebel application server to the DB2 database on z/OS is also discussed.

- Customization of the Siebel schema for a production environment.

- Migration from a development system to a production system.

- Database administration tasks, including:
  - Monitoring dynamic SQL
  - Identifying and inactivating unused indexes
  - Table space partitioning
  - Statistics gathering
  - Reorg, backup, and recovery