Progress®



Dump and Load a Database

OpenEdge®

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

Updated: 2021/06/28

Table of Contents

Preface....7

Determining whether to dump and load a database....9

Table analysis report....10
Index analysis report....12

Overview of dumping and loading a database....13

Dumping data definitions....17

Procedure: Dumping data definitions using a Data tool....19

Dumping database contents....21

Procedure: Dumping table contents using a Data tool....23
Dumping table contents using a binary dump....24
Using PROUTIL DUMP to binary-dump table contents....25
Dumping table contents using a multi-threaded binary dump....26
Using PROUTIL to perform a multi-threaded binary dump....27
Using a parameter file to perform a multi-threaded binary dump in Windows....27
Dumping table contents using a selective binary dump....28
Using PROUTIL to perform a selective binary dump....28
Procedure: Dumping sequence values using a Data tool....29

Loading data definitions....31

Modifying a data definitions file....33
Procedure: Loading data definitions using a Data tool....34

Loading database contents....35

Procedure: Loading table contents using a Data tool....36
Loading table contents using a bulk load....38
Procedure: Creating a bulk loader description file....39
Invoking the bulk loader....39
Loading table contents using a binary load....40
Using PROUTIL LOAD to binary-load table contents....41
Loading table contents using a binary load with a dump list file....42
Using PROUTIL to binary-load with a dump list file....43
Procedure: Loading sequence values using a Data tool....43

Rebuilding indexes.....45
Using the PROUTIL IDXBUILD utility.....47
Allocating memory for sorting and merging indexes.....48
Specifying sort locations for index sorting.....50
Guidelines for choosing sort locations.....51
Creating a sort file specification file.....51
Procedure: Rebuilding indexes using PROUTIL IDXBUILD.....52

Preface

Purpose

This guide explains how to dump and load OpenEdge® databases. The procedures in this guide help you analyze tables and indexes, choose the best tool for dumping table contents or data definitions, and perform dump and load operations. This guide assumes that you are familiar with the OpenEdge RDBMS planning concepts discussed in *OpenEdge Getting Started: Database Essentials*.

Audience

This book is intended for OpenEdge database administrators.

For the latest documentation updates go to https://docs.progress.com.

Organization

Determining whether to dump and load a database on page 9

Explains how to generate database analysis reports for statistics about tables and indexes.

Overview of dumping and loading a database on page 13

Defines dumps and loads, suggests a dump and load schedule, and recommends related tasks.

Dumping data definitions on page 17

Explains how to dump data definitions from a source database to a data definition file.

Dumping database contents on page 21

Explains how to dump table contents and sequence values from a source database.

Loading data definitions on page 31

Explains how to revise the structure description of the source database, create a target database, and use a Data tool to load data definitions.

Loading database contents on page 35

Describes ways to load table contents and sequence values.

Rebuilding indexes on page 45

Describes how to allocate memory, manage index sorting, and rebuild indexes.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

1

Determining whether to dump and load a database

To determine whether you need to dump and load a database, you should get statistics about tables and indexes of application data in your database. To do this, you use the PROUTIL DBANALYS utility to generate a database analysis report. You can run PROUTIL DBANALYS online or offline.

A database analysis report contains individual reports about tables, indexes, and chains. Of these, the most important reports are about tables and indexes. You should analyze the statistical information to determine whether it is time to dump and load.

The command to generate a database analysis is:

```
proutil db-name -C dbanalys > output-file
```

Note: Because PROUTIL DBANALYS generates heavy I/O, you should run it during off-hours or against a copy of the production database.

Here is an example of how to generate a database analysis report for the sports database and output the results to the sports.txt file:

```
proutil sports -C dbanalys > sports.txt
```

For details, see the following topics:

- Table analysis report
- Index analysis report

Table analysis report

```
RECORD BLOCK SUMMARY FOR AREA "Inventory": 8
-------------------------------------------------------


RECORD BLOCK SUMMARY FOR SHARED TABLES
---------------------------------------------


                                                        -Record Size (B)-
Table                                   Records    Size   Min   Max  Mean
PUB.Bin                                     770   26.2K    33    35    34
PUB.InventoryTrans                           72    3.4K    45    51    49
PUB.Item                                     55    8.9K   128   258   166
PUB.POLine                                 5337  217.5K    40    44    41
PUB.PurchaseOrder                          2129   76.9K    37    40    37
PUB.Supplier                                 10    1.2K    96   168   121
PUB.SupplierItemXref                         56    1.1K    18    20    19
PUB.Warehouse                                14    1.3K    82   104    92

                                 ---------------------------------------------
Subtotals:                                 8443  336.6K    18   258    40




                                 ---------------------------------------------
Summary for AREA "Inventory": 8
Subtotals:                                 8443  336.6K    18   258    40
```

The table analysis part of the report provides information about the amount of record fragmentation and scatter for each table in each storage area of a database. For each table, the report shows the total number of records, table size, the minimum, maximum, and mean of the record size (in bytes), record fragments count and factor, and scatter factor, as shown in the graphic.

The following table lists each field with its description:

Field	Description
Records	Displays the number of records in each table. This is important in determining the effect of record fragmentation. For example, if a table has 10 or fewer records, you can safely ignore its fragmentation. But if it has 100,000 or more records, you must consider its fragmentation.
Fragments Count	Shows the total number of record fragments for each table. The optimal count is the same as the total number of records.

Field	Description
Fragments Factor	Displays the degree of record fragmentation for each table. • If the fragments factor is 1.0, there is no fragmentation. • If the fragments factor is equal to or greater than 2.0, then the table has a significant amount of fragmentation.
Scatter Factor	Indicates the degree of distance between database blocks in each table. • If the scatter factor is 1.0, it indicates the records occupy contiguous database blocks and there is no scatter. • If the scatter factor is between 2.0 and 4.0, monitor closely. • If the scatter factor is 4.0 or greater, there is significant scatter in the table.

Index analysis report

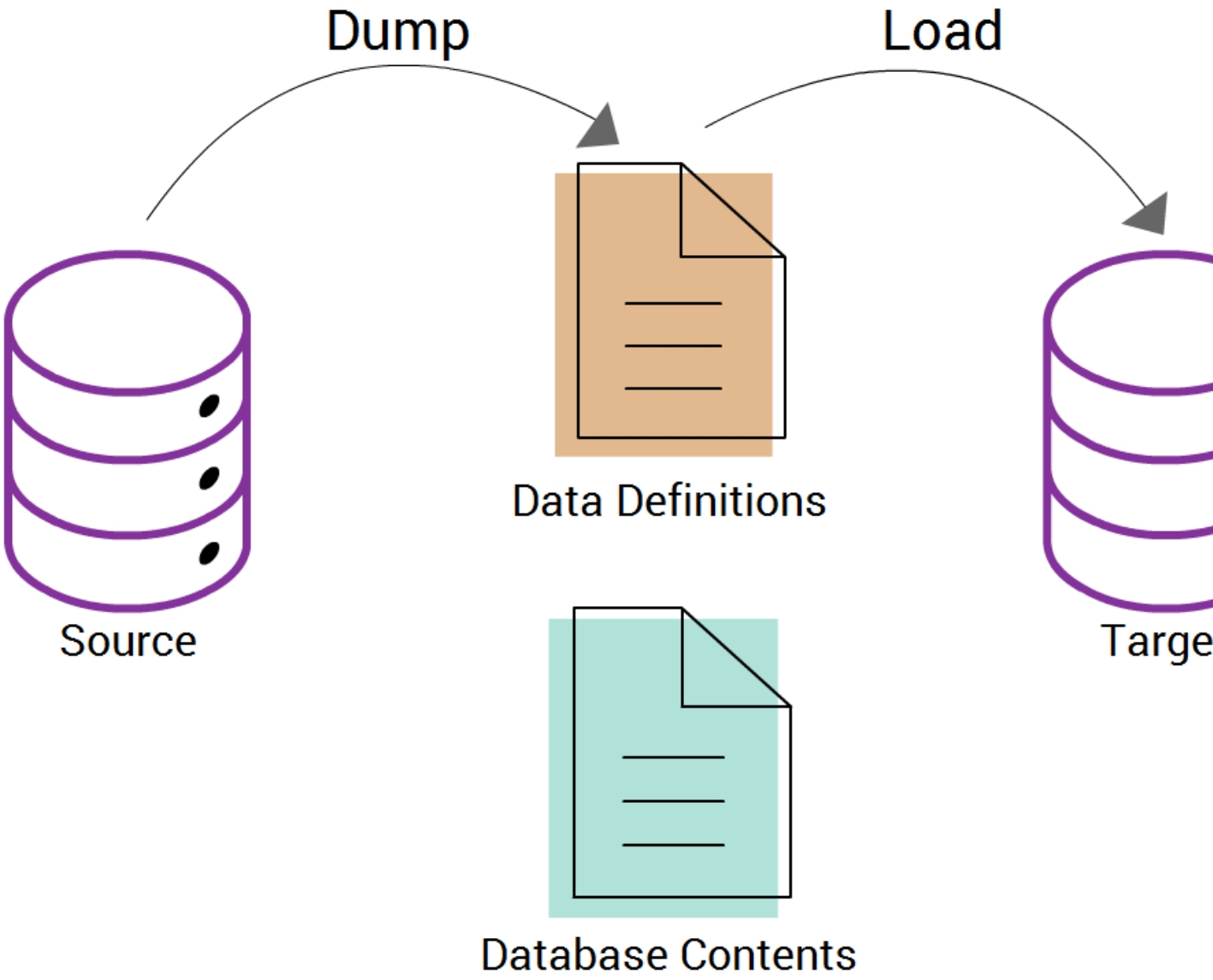
```
INDEX BLOCK SUMMARY FOR AREA "Inventory": 8
------------------------------------------------------


INDEX BLOCK SUMMARY FOR SHARED OBJECTS:
-----------------------------------------
Table                       Index  Fields Levels          Blocks    Size  % Util
PUB.Bin
  BinNum                       10       1      2               3    6.8K    57.1
  ItemNum                      11       1      1               1    1.8K    46.2
  WarehouseNumItemNum          12       2      2               3    6.8K    57.4
PUB.Item
  CatDescription               34       1      2               3    4.1K    34.5
  Category2ItemName            35       2      1               1  958.0B    23.5
  CategoryItemName             36       2      1               1    1.0K    26.1
  ItemName                     37       1      1               1  934.0B    23.0
  ItemNum                      33       1      1               1  503.0B    12.4
PUB.POLine
  PONumLinenum                 47       2      2              15   51.3K    86.1
PUB.PurchaseOrder
  PONum                        48       1      2               6   18.8K    78.9
PUB.Supplier
  SupplierID                   56       1      1               1   97.0B     2.4
PUB.SupplierItemXref
  ItemNumSupplierID            58       2      1               1  624.0B    15.3
  SupplieridItemNum            57       2      1               1  542.0B    13.3
PUB.Warehouse
  warehousename                62       1      1               1  307.0B     7.5
  warehousenum                 61       1      1               1  133.0B     3.3

                            ------------------------------------------------------
Subtotals:                                                     40   94.8K    59.6
```

The index analysis part of the report provides statistics about indexes associated with each table in each storage area of a database. For each index, the report displays the number of levels, number of blocks, index size, utilization percentage, and factor, as shown in the graphic.

While you may not need the index analysis report data to determine whether it is time to dump and load your database, the information can help you determine if it is time to compact or rebuild indexes.

2

Overview of dumping and loading a database



How frequently you should dump a database depends on what types of storage areas are used in the database:

- If Type I areas are used, dump and load every 12 to 18 months.
- If Type II areas are used, dump and load every 3 to 5 years.

When you dump a database, you export data definitions and database contents from a source database to external files. When you load a database, you import data definitions and database contents from the external files into a target database.

Here is what you dump and load:

- Data definitions

- Table definitions
- Sequence definitions
- Database contents
- Table contents
- Sequence values

Depending on how your database is used, you can also dump and load the following:

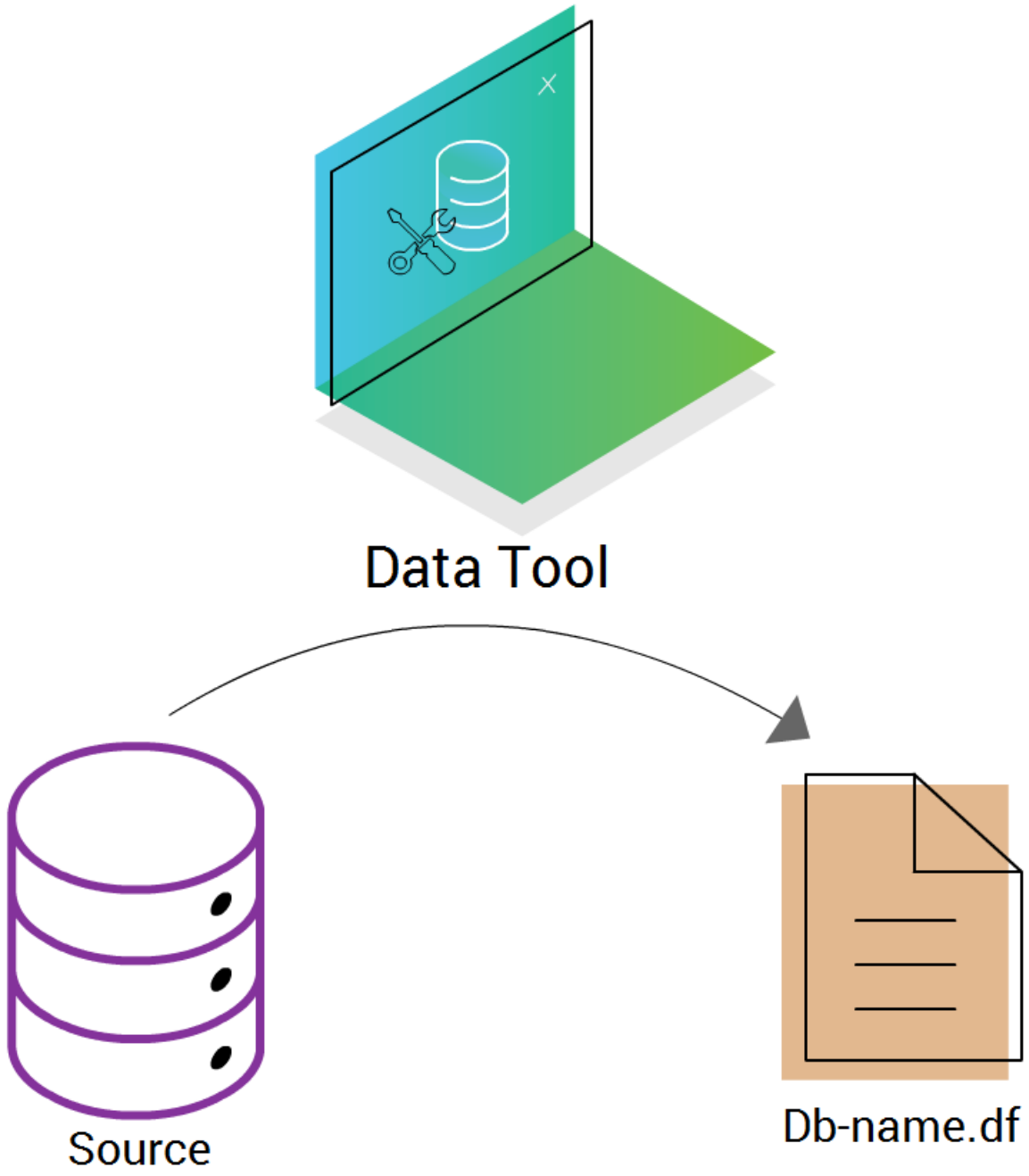
- If security is enforced, you can dump and load user table contents, security domains, security permissions, and database identification properties.
- If auditing is used, you can dump and load audit policies and audit data.
- If OpenEdge Table Partitioning is used, you can dump and load definitions and data for table partitions, partition policies, and partition policy details.
- If OpenEdge Multi-tenancy is used, you can dump and load definitions and data for tenants and tenant groups.
- If OpenEdge SQL is used, you can dump and load SQL objects and data.

When you perform a dump and load, you can take this opportunity to:

- Ensure that you have the optimal number of records per block, the optimal number of blocks per cluster, and the right database block size.
- Change the physical placement of tables and indexes in the storage areas of the target database.
- Migrate your database from an older Progress® OpenEdge® RDBMS version to a newer version or from one operating system (OS) to another.

3

Dumping data definitions



To begin the dump process, you dump data definitions from the source database. If you are able to schedule downtime, you can dump data definitions offline. If your business environment requires that your database be available 24x7, then you can dump data definitions online.

When you dump data definitions of an entire database, OpenEdge RDBMS creates a data definition (.df) file, which contains the definitions of all tables, fields, indexes, and sequences in the database. No data is dumped.

You use a Data tool (the Data Administration tool in Windows or the Data Dictionary tool in UNIX) to dump data definitions of a database.

For details, see the following topics:

- Procedure: Dumping data definitions using a Data tool

Procedure: Dumping data definitions using a Data tool

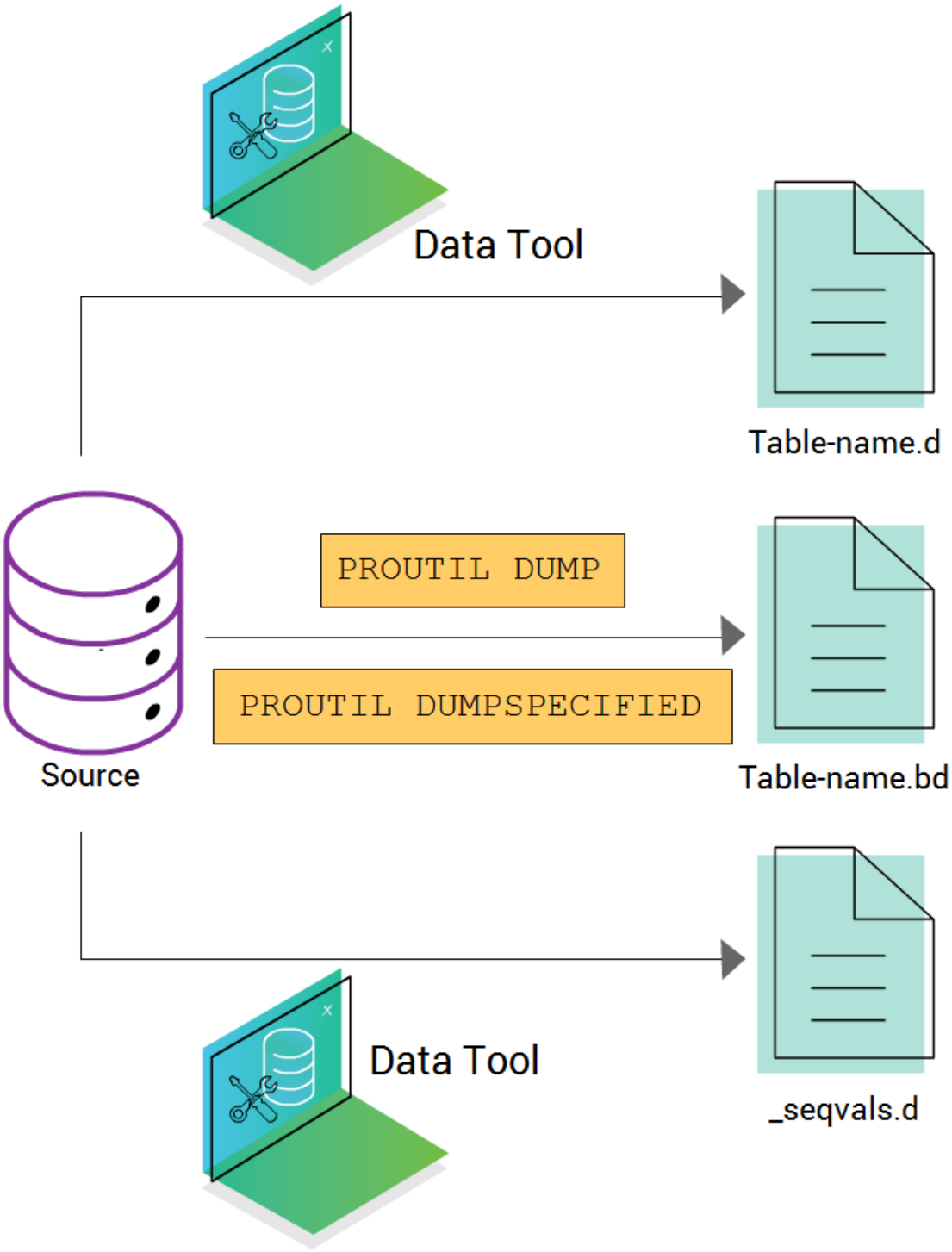
Follow these steps to dump data definitions from a database offline or online, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the database: • Click **Database > Connect** and then click **Browse** and navigate to the directory where your database resides. • Double-click the .db file of the database to select it. • To access the database in single-user mode, click **OK**. • To access the database in multi-user mode, click **Options**, select the **Multiple Users** check box, and then click **OK**. You are connected to the database as indicated by the database's status in the lower-left corner.
3	Click **Admin > Dump Data and Definitions >Data Definitions (.df file)**. The **Select Tables** dialog box appears, showing an alphabetical list of all the tables in the database.
4	To dump definitions of the entire database, click the **Select Some** button. Then, in the **Select Tables by Pattern Match** dialog box, ensure that * is entered in the **Table Name** field, and click **OK**.
5	In the **Select Tables** dialog box, click **OK**. The Dump Data Definitions dialog box appears. The default name for the data definitions file with the .df extension is displayed in the Output File field. By default, the .df file is named after the database.
6	By default, the .df file is created in the current working directory. If you want to change the default name and location of the data definitions file, enter a fully qualified filename in the **Output File** field or click **Files** and then specify the new location and filename.

Steps	Action
7	Click **OK** in the Dump Data Definitions dialog box to create the data definitions file. After the data definitions file is created, the Information box displays a message confirming that the dump of data definitions is completed.
8	Click **OK** to close the Information box.
9	Verify that the .df file exists on your file system. If you have time, you should back it up.

4

Dumping database contents



So far, you have learned how to dump data definitions. Data definitions enable you to create new database objects. After you create objects in a target database, you populate them with values. To do this, you must dump database contents from the source database and then load the contents into the target database.

To speed up the dump, you should first dump the contents of static tables. You should also dump static portions of dynamic tables using a selective binary dump.

The database contents that you dump include:

- Table contents—You can dump table contents as text dump (.d) files or binary dump (.bd) files. You use a Data tool to create .d files. You use the PROUTIL utility with the DUMP or DUMPSPECIFIED qualifier to create .bd files.
- Sequence values—You dump sequence values as a .d file. You use a Data tool to dump sequence values.

For details, see the following topics:

- Procedure: Dumping table contents using a Data tool
- Dumping table contents using a binary dump
- Dumping table contents using a multi-threaded binary dump
- Dumping table contents using a selective binary dump
- Procedure: Dumping sequence values using a Data tool

Procedure: Dumping table contents using a Data tool

If you have a small database, an easy way to dump table contents from a source database online or offline is to use a Data tool. A text dump (.d) file is created for each table in the source database.

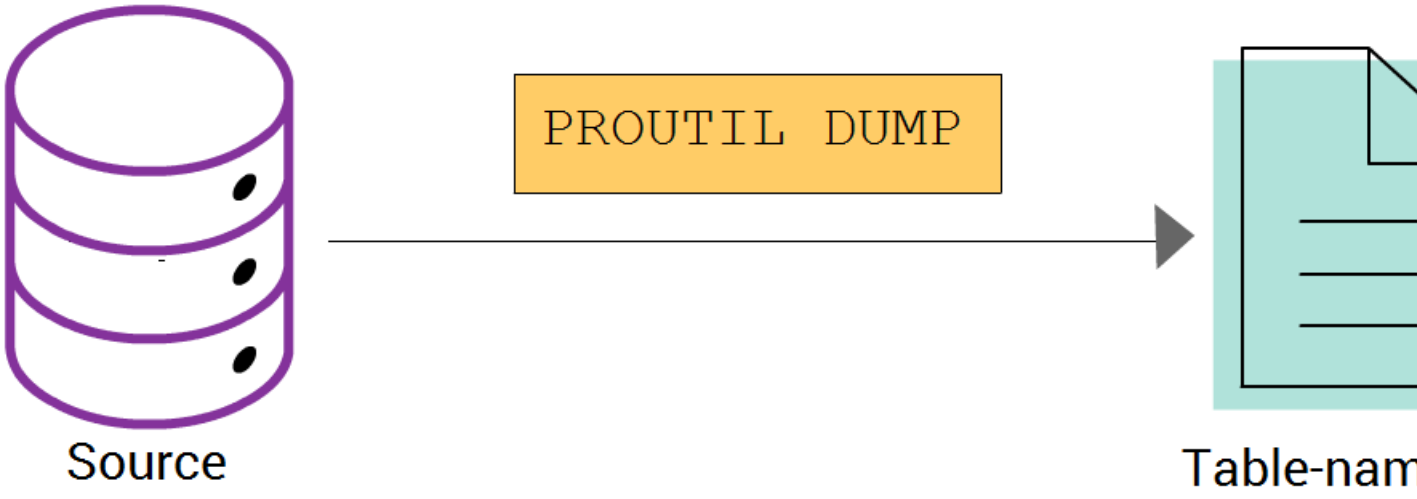
Note: When you dump table contents to .d files, keep in mind that the .d files must observe the 2GB limit even if your database has been enabled for large files.

Follow these steps to dump table contents online or offline, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the database.
3	Click **Admin > Dump Data and Definitions > Table Contents (.d file)**. The **Select Tables** dialog box appears, showing an alphabetical list of all the tables in the database.
4	To dump the contents of all the tables in the database, click the **Select Some** button. Then, in the **Select Tables by Pattern Match** dialog box, ensure that ***** is entered in the **Table Name** field, and click **OK**.

Steps	Action
5	In the **Select Tables** dialog box, click **OK**. The Dump Data Contents dialog box appears.
6	By default, the output directory is the current working directory. If you want to change it, specify the location in the **Output Directory** field or click **Dir** and browse to the location.
7	If your database contains large objects (LOBs), select the **Include LOB** check box. By default, the LOB directory is the current working directory. If you want to change it, specify the location in the LOB Directory field or click Dir and browse to the location.
8	Click **OK** in the **Dump Data Contents** dialog box to create the .d files. The Dump Data dialog box displays the tables that are being dumped, the respective .d files used for dumping, and the number of records dumped. After the dump files are created, the Information box displays a message stating the number of tables for which dump files have been created.
9	Click **OK** to close the Information box.
10	Verify that the required .d files exist on your file system. If you have time, you should back them up.

Dumping table contents using a binary dump



A binary dump is the fastest way to dump table contents. If you have a large database, it is the recommended way. A binary dump generates binary data, which is more compact than text data. A binary dump (.bd) file is created for each table in the source database.

You can dump table contents to .bd files online or offline, using the PROUTIL DUMP utility. To speed up the dump, you can use batch files to binary-dump multiple tables in parallel. To do this, you should divide all the tables into similarly sized groups and dump each group separately. For best performance, you should also dump the groups to non-database disks.

Note: If you have multiple computers available, a quick way to perform binary dump and load is to dump to one machine, transfer the .bd files to another machine, and then load them into the target database.

- When you perform a binary dump of a table greater than 2GB on a file system without a 2GB limit, a single .bd file is created.
- However, on a file system with a 2GB limit, dumping a table greater than 2GB creates multiple .bd files. If multiple .bd files are created, you must ensure that all the .bd files are loaded into the target database.

Note: When PROUTIL DUMP creates multiple .bd files, the files are named using this convention: tablename.bdn, where n is an integer starting from 2. For example, assume the Orderline table of the sports database is greater than 2GB. When Orderline is binary-dumped, three .bd files could be created: orderline.bd, orderline.bd2, and orderline.bd3.

Using PROUTIL DUMP to binary-dump table contents

You use the PROUTIL utility with the DUMP qualifier to dump the contents of a table to a .bd file online or offline. PROUTIL DUMP also supports dumping binary large objects (BLOBS).

The command to binary-dump the contents of a table online or offline is:

```
proutil db-name –C dump table-name target-directory
```

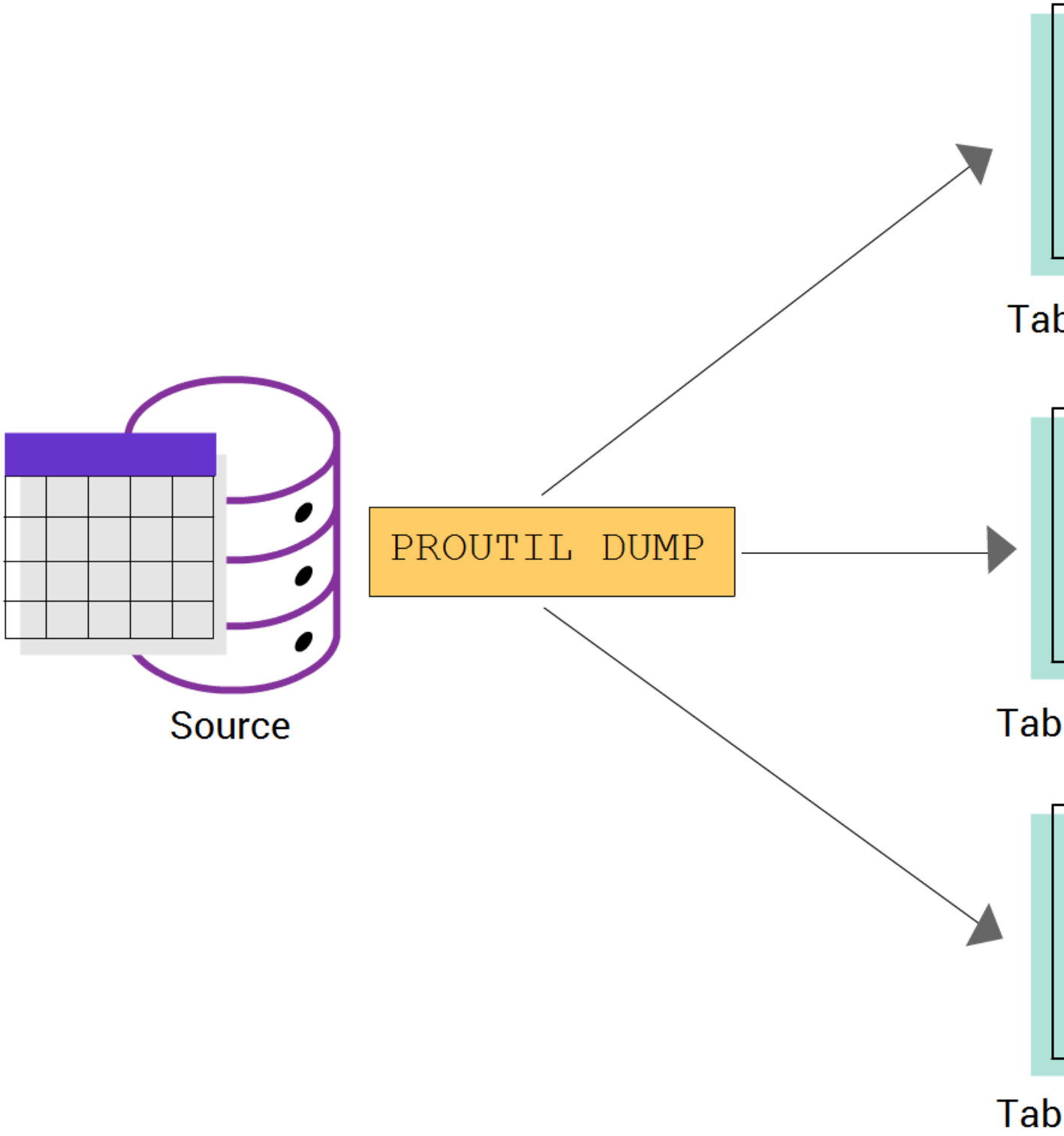
Where:

`target-directory`	The name of the directory for the .bd file. Use a dot (.) to denote the current working directory.

To use PROUTIL DUMP to dump the contents of the Customer table in the sports database to a .bd file in the data directory, enter:

```
proutil sports –C dump customer ../data
```

Dumping table contents using a multi-threaded binary dump



If you need to dump online, it can take a lot of time to dump a very large table. However, if you have an OpenEdge Enterprise RDBMS license and your database is running on a multiprocessor system, you can perform a multi-threaded binary dump of a very large table using PROUTIL DUMP.

A multi-threaded binary dump creates multiple .bd files, each generated by a specific thread of your multiprocessor system. Because of this, you can make full use of your multiprocessor system and reduce the dump time from hours to just minutes.

Using PROUTIL to perform a multi-threaded binary dump

If you must perform a binary dump online, you can perform a multi-threaded binary dump on large tables. You use PROUTIL DUMP with the multi-threaded qualifiers to perform a multi-threaded binary dump of the contents of a table to one or more .bd files.

The command to perform a multi-threaded binary dump of the contents of a table online is:

```
proutil db-name –C dump table-name target-directory –thread n –threadnum nthreads
–dumplist dumplist-file
```

Where:

n	Specifies whether your database machine is threaded. • Use 0 for a single-threaded dump. • Use 1 for a multi-threaded dump.
`nthreads`	The maximum number of threads to create in a multi-threaded dump. The default value is the number of system CPUs.
`dumplist-file`	The name of a dump list file that lists all .bd files created by the multi-threaded binary dump. You can use this file later to perform a binary load of those .bd files into the target database.

Assume that the sports database is running.

Here is an example of how to use PROUTIL DUMP to perform a multi-threaded binary dump of the contents of the Order table. The command invokes a two-thread dump to create two .bd files in the data directory.

```
proutil sports –C dump order ../data -thread 1 -threadnum 2
```

The contents of the Order table are dumped to order.bd and order.bd2.

Using a parameter file to perform a multi-threaded binary dump in Windows

In Windows, PROUTIL supports only nine tokens, but a full multi-threaded binary dump command exceeds this limit. As a result, a PROUTIL DUMP command might skip some parameters when it is executed.

To avoid this problem, you should create a parameter (.pf) file to hold the multi-threaded parameter values and invoke the .pf file when you perform the multi-threaded binary dump.

Here is an example of a .pf file, orderline.pf, that specifies a three-threaded binary dump and a dump list file, orderline.txt:

```
-thread 1-threadnum 3-dumplist orderline.txt
```

To invoke orderline.pf to perform a multi-threaded binary dump of the Orderline table in the sports database to the working directory, enter:

```
proutil sports -C dump orderline . -pf orderline.pf
```

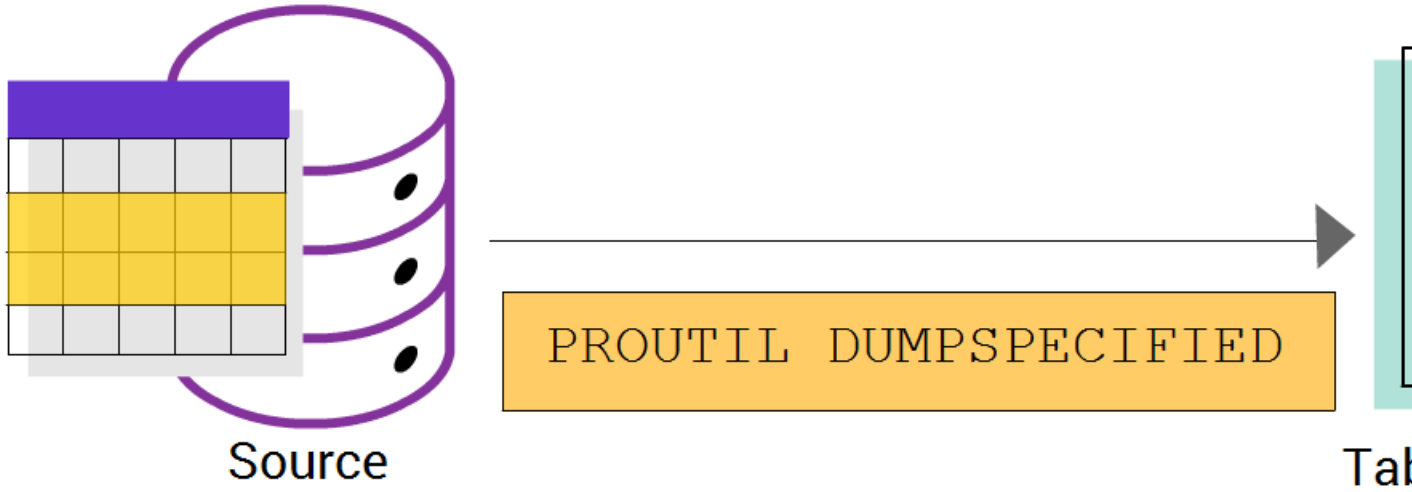
Three .bd files, orderline.bd, orderline.bd2, and orderline.bd3, and a dump list file, orderline.txt, are created.

The dump list file, orderline.txt, may look like this:

```
C:\OpenEdge\WRK\dba\db/orderline.bdC:\OpenEdge\WRK\dba\db/orderline.bd2C:\OpenEdge\WRK\dba\db/orderline.bd3
```

You can use this file later as an input parameter for a binary load of the .bd files into the target database.

Dumping table contents using a selective binary dump



To speed up a dump, you should first dump static portions of dynamic tables using a selective binary dump. A selective binary dump dumps a subset of a table based on specified criteria to a .bd file. You can do this online or offline.

With this method, you can dump large amounts of data of very large tables without having to do a full table dump, thereby reducing dump time from hours to just minutes. To speed up the dump, you can also use batch files to selectively dump multiple large tables in parallel.

Using PROUTIL to perform a selective binary dump

You use the PROUTIL utility with the DUMPSPECIFIED qualifier to perform a selective binary dump of the contents of a table.

The command to selectively binary-dump the contents of a table online or offline is:

```
proutil db-name –C dumpspecified table-name.field-name operator1 low-value [AND operator2
 high-value] target-directory
```

Where:

`table-name`	The name of the table that contains the data you want to dump.
`field-name`	The name of the index field used as the criterion in the selective dump.
`operator1`	The first operator in the range. This can be EQ (equals to), GT (greater than), LT (less than), GE (greater than or equal to), or LE (less than or equal to).
`low-value`	The low value against which the field value will be compared.
`AND`	Specifies a range for the binary dump. The first operator and value combined with the second operator and value define a subset. Note: You cannot use AND to dump two sets of non-contiguous results.
`operator2`	The second operator in the range. This can only be LT (less than) or LE (less than or equal to).
`high-value`	The high value against which the field value will be compared.
`target-directory`	The name of the directory for the .bd file. Use a dot (.) to denote the current working directory.

Here is an example of how to selectively binary-dump the contents of the Order table in the sports database. The command dumps the records with order dates after December 31, 2015 and before October 1, 2016 to the working directory.

```
proutil sports –C dumpspecified order.orderdate gt 12-31-2015 and lt 10-01-2016
.
```

Procedure: Dumping sequence values using a Data tool

If your source database has CREATE TRIGGER statements that generate and assign sequence values to key fields, then the target database must begin with the same sequence values. To ensure this happens, you must dump sequence values from the source database and load them into the target database.

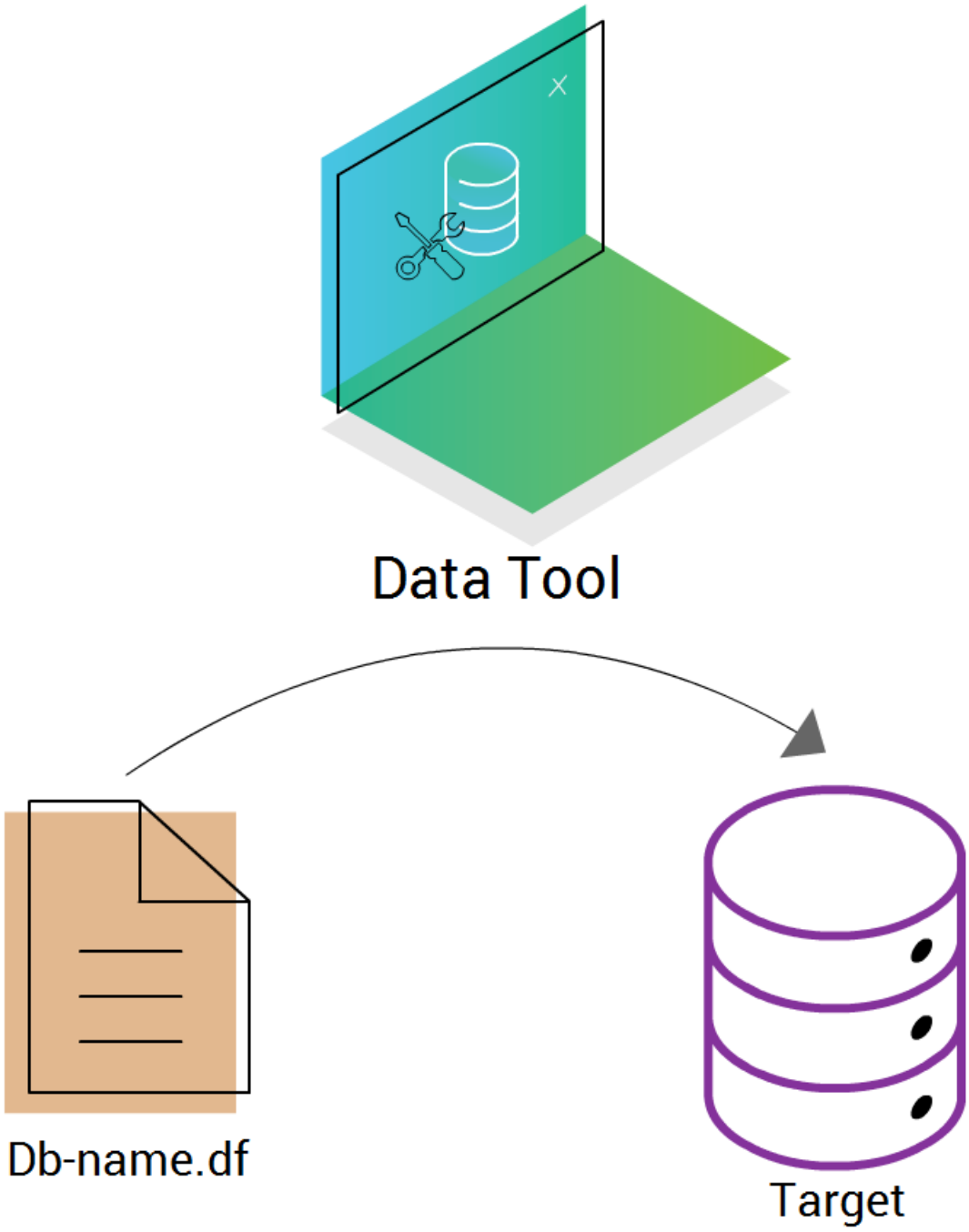
Although sequence definitions of a database are dumped as part of the data definitions of the database, you must dump sequence values to a text dump (.d) file separately.

Follow these steps to dump sequence values from a database online or offline, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the database.
3	Click **Admin > Dump Data and Definitions > Sequences Current Values**. The **Dump Sequence Current Values** dialog box appears.
4	By default, the output file takes the **name _seqvals.d**, and is stored in the current working directory. If you want to change the default name or location of the sequence values file, enter a fully qualified filename in the **Output File** field or click **Files** and then specify the location and filename.
5	Click **OK** in the **Dump Sequence Current Values** dialog box to create the sequence values file. After the sequence values file is created, the Message box confirms that the dump of the sequence values is completed.
6	Click **OK** to close the Message box.
7	Verify that the sequence values file exists on your file system. If you have time, you should back it up.

5

Loading data definitions



After you complete dumping data definitions, table contents and sequence values from a source database, you are ready to begin the load process. But first you need to create a target database.

If you want to modify the number of records per block, the number of blocks per cluster, or the location, size, and number of the BI, application data, and AI extents, you need to copy and revise the structure description (.st) file of the source database for the target database.

After you create a target database, you begin the load process by loading data definitions.

For details, see the following topics:

- Modifying a data definitions file
- Procedure: Loading data definitions using a Data tool

Modifying a data definitions file

If you want to change the physical placement of tables and indexes in the storage areas for the target database, you need to modify the data definitions (.df) file from the source database.

The .df file is a text file that contains the definitions of tables, fields, indexes, and sequences in a database. The definitions include different properties and their values. For example, the definition of a table contains properties such as AREA, DUMP-NAME, and TABLE-TRIGGER.

One of the important properties that you may choose to modify is AREA. This property specifies the storage area in which a table or index is to be stored.

Here is an example of a portion of a data definitions file, sports.pf:

```
ADD SEQUENCE "NextCustNum" INITIAL 1000 INCREMENT 5 CYCLE-ON-LIMIT no MIN-VAL
 1000
...
ADD TABLE "Customer"
 AREA "Cust_Data"
 DESCRIPTION "The customer table contains customer information including balance
 and address."
 DUMP-NAME "customer"
 TABLE-TRIGGER "CREATE" NO-OVERRIDE PROCEDURE "sports3000trgs/crcust.p" CRC
"?"
 TABLE-TRIGGER "DELETE" NO-OVERRIDE PROCEDURE "sports3000trgs/delcust.p" CRC
"?"
ADD FIELD "CustNum" OF "Customer" AS integer  FORMAT ">>>>9" INITIAL "0" LABEL
 "Cust Num" POSITION 2 MAX-WIDTH 4
 VALEXP "custnum > 0"
 VALMSG "Customer number must be greater than zero"
 HELP "Please enter a customer number."
 ORDER 10
ADD FIELD "Name" OF "Customer" AS character  FORMAT "x(30)" INITIAL "" LABEL
"Name" POSITION 3 MAX-WIDTH 60
 HELP "Please enter a name."
 ORDER 30...
ADD INDEX "CustNum" ON "Customer"  AREA "Cust_Index" UNIQUE PRIMARY INDEX-FIELD
 "CustNum" ASCENDING
...
```

Procedure: Loading data definitions using a Data tool

If your business environment allows scheduled downtime, you can load data definitions into a target database offline. If your business environment requires that your database be as highly available as possible, then you can load data definitions online.

Follow these steps to load data definitions into a target database offline or online, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the target database.
3	Click Admin > Load Data and Definitions > Data Definitions (.df file). The Load Data Definitions dialog box appears.
4	Click Files. The Find Input File box appears.
5	Browse to the location of the data definitions (.df) file, select the file, and then click Open. The Load Data Definitions dialog box appears.
6	Click OK to load data definitions. Note: During the load, OpenEdge records any errors detected to a file named after the .df file with the .e extension. After the data definitions are loaded, the Information box confirms that the data definitions are loaded successfully.
7	Click OK to close the Information box.

6

Loading database contents

After you load data definitions into a target database, you next need to load table contents and sequence values from associated text dump (.d) files or binary dump (.bd) files.

OpenEdge provides different methods for loading table contents. You choose a method based on the type of dump files you have. The following table lists the different types of dump files and the method you can use to load them into the target database.

If you have …	Then use …
Text dump (.d) files	A Data tool or bulk load
Binary dump (.bd) files	PROUTIL LOAD
Multi-threaded binary dump (.bdn) files	PROUTIL LOAD or PROUTIL LOAD with the –DUMPLIST qualifier
Selective binary dump (.bd) files	PROUTIL LOAD
A sequence value dump (_seqvals.d) file	A Data tool

Next, you will learn how to use different methods to load table contents. Then you will learn how to load sequence values.

For details, see the following topics:

- Procedure: Loading table contents using a Data tool
- Loading table contents using a bulk load
- Loading table contents using a binary load
- Loading table contents using a binary load with a dump list file
- Procedure: Loading sequence values using a Data tool

Procedure: Loading table contents using a Data tool

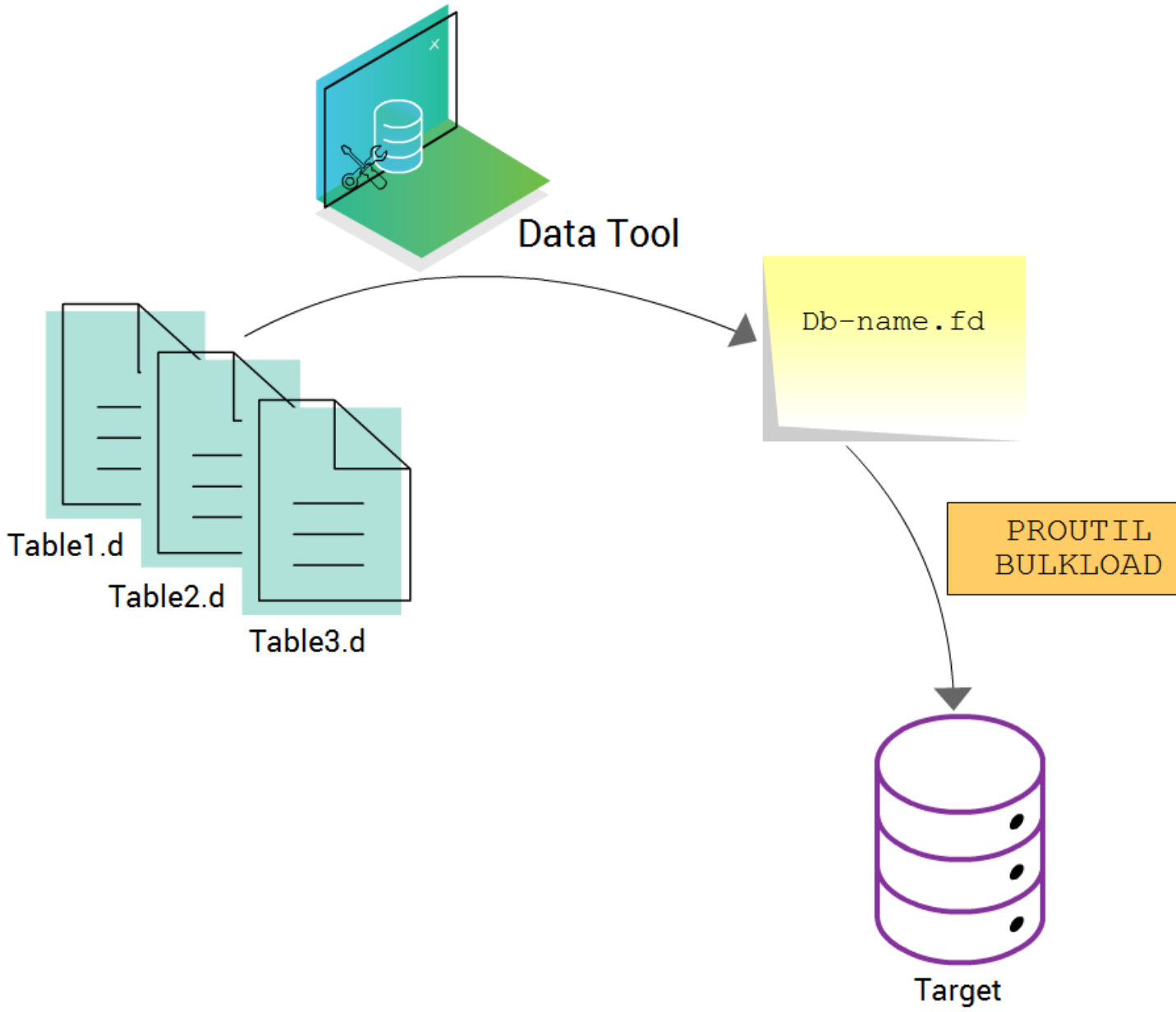
For best performance, you should deactivate all indexes before loading table contents, and then rebuild indexes after the load. Index deactivation not only makes it faster to load but also reduces record scatter, enhancing sequential read performance. This is especially useful for tables and their indexes that reside in the same storage area.

Follow these steps to load table contents from .d files into a target database online or offline, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the target database.
3	Deactivate all indexes: • Click Utilities **> Index Deactivation**. A Warning dialog box appears, stating that data for the deactivated indexes will not be accessible until you rebuild the indexes. • Click **OK** to continue with the deactivation. The **Select Tables** appears, with the **ALL** value highlighted. • Click **OK** and then click **Yes** on the Question dialog box to deactivate all indexes. After all the indexes are deactivated, the Information box confirms that the index deactivation is completed. • Click **OK** to close the Information box.
4	Click **Admin > Load Data and Definitions > Table Contents (.d file)**. The Select Tables dialog box appears, showing an alphabetical list of all the tables in the database.
5	To load table contents into all tables in the target database, click the **Select Some** button. Then, in the **Select Tables by Pattern Match** dialog box, ensure that * is entered in the **Table Name** field, and click OK.

Steps	Action
6	In the **Select Tables** dialog box, click OK. The Load Data Contents dialog box appears.
7	By default, the input directory is the current working directory. If the .d files that you want to load are stored in another location, specify the location in the Input Directory field or click Data Dir and browse to the location.
8	If you have large objects (LOBs), select the **Include LOB** check box. By default, the LOB directory is the current working directory. If the LOBs are stored in another location, specify the location in the LOB Directory field or click Lob Dir and browse to the location.
9	Click **OK** in the **Load Data Contents** dialog box to load table contents from the .d files. The Load Table Contents dialog box displays the tables that are being loaded, the respective .d files used for loading, the number of records loaded, and errors, if any. **Note:** During the load, OpenEdge records any errors detected to a file named after a corresponding .d file with the .e extension. After the contents of all the tables are loaded, the Information box confirms that the load of table contents is completed.
10	Click **OK** to close the Information box.

Loading table contents using a bulk load



An alternative to loading table contents using a Data tool is to use a bulk load. A bulk load automatically deactivates all indexes.

Note: Use this method only if your business environment allows sufficient downtime so that you can perform the bulk load offline and rebuild indexes after the load.

To bulk-load .d files into a target database, you perform the following tasks:

1. Create a bulk loader description (.fd) file.
2. Invoke the bulk loader.

Next, you will learn how to perform each of these tasks.

Procedure: Creating a bulk loader description file

You must use a Data tool to create a bulk loader description (.fd) file.

Follow these steps to create a bulk loader description file:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the target database.
3	Click **Admin > Create Bulk Loader Description File**. The Select Tables dialog box appears, showing an alphabetical list of all the tables in the database.
4	To load contents into all tables in the target database, click the **Select Some** button. Then, in the **Select Tables by Pattern Match** dialog box, ensure that ***** is entered in the **Table Name** field, and click OK.
5	In the **Select Tables** dialog box, click OK. The **Make Bulk Load Description File** dialog box appears. The default name for the bulk load description file with the .fd extension is displayed in the Output File field. By default, the .fd file is named after the database.
6	By default, the .fd file is created in the current working directory. If you want to change the default name and location of the bulk loader description file, enter a fully qualified filename in the Output File field or click Files and then specify the new name and location.
7	Click **OK** in the Make Bulk Load Description File dialog box to create the .fd file. After the bulk loader description file is created, the Information box displays a message confirming that the file is created.
8	Click **OK** to close the Information box.

Invoking the bulk loader

The bulk loader uses the .d files and the .fd file to load all table contents into the target database offline. First, you must shut down the target database, and then invoke the bulk loader. You use the PROUTIL utility with the BULKLOAD qualifier to run the bulk loader.

The command to invoke the bulk loader offline is:

```
proutil db-name –C bulkload .fd-file -i
```

Where:

`-i`	To make a bulk load run faster, first back up the database and then use this No Crash Protection (-i) parameter to perform the bulk load in the no-integrity mode, rather than in the default full-integrity mode. Note: If your system crashes during the load, you must restore the database from the most recent backup and resume the load.

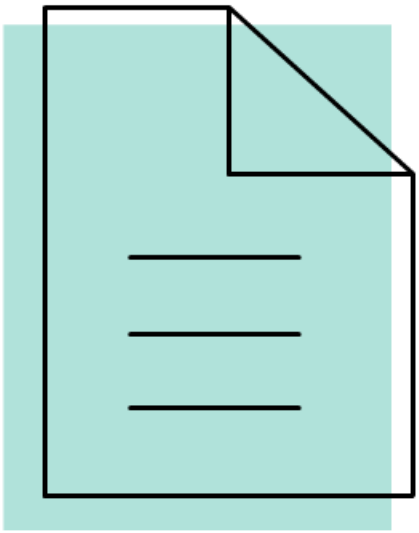
After the bulk load is completed, you should check the database log (.lg) file for errors.

Suppose you have dumped all table contents from the sports database to .d files. And suppose you have created a bulk loader description file, sports.fd, to load all .d files into the newsports database.

To use the bulk loader to load all .d files into the newsports database using sports.fd and the No Crash Protection (-i) parameter, enter:

```
proutil newsports –C bulkload sports.fd -i
```

Loading table contents using a binary load



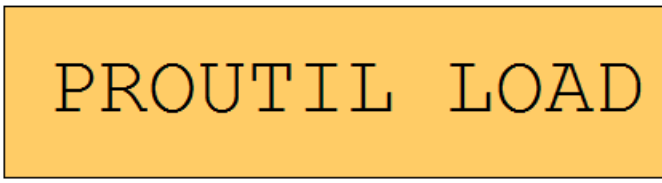


Table-name.bd

If you binary-dumped table contents from a source database to .bd files, you must binary-load those .bd files into a target database, using the PROUTIL LOAD utility. PROUTIL LOAD also supports loading binary large objects (BLOBS).

A binary load is the fastest way to load table contents because it:

- Bypasses all CREATE TRIGGER statements.
- Deactivates all indexes.
- Loads binary data, which is more compact than text data.

To speed up the load, you can also use batch files to binary-load data into multiple tables in parallel. This is especially useful if each of your tables resides in a separate storage area.

Using PROUTIL LOAD to binary-load table contents

You use the PROUTIL utility with the LOAD qualifier to binary-load the contents of a table.

The command to binary-load table contents online or offline is:

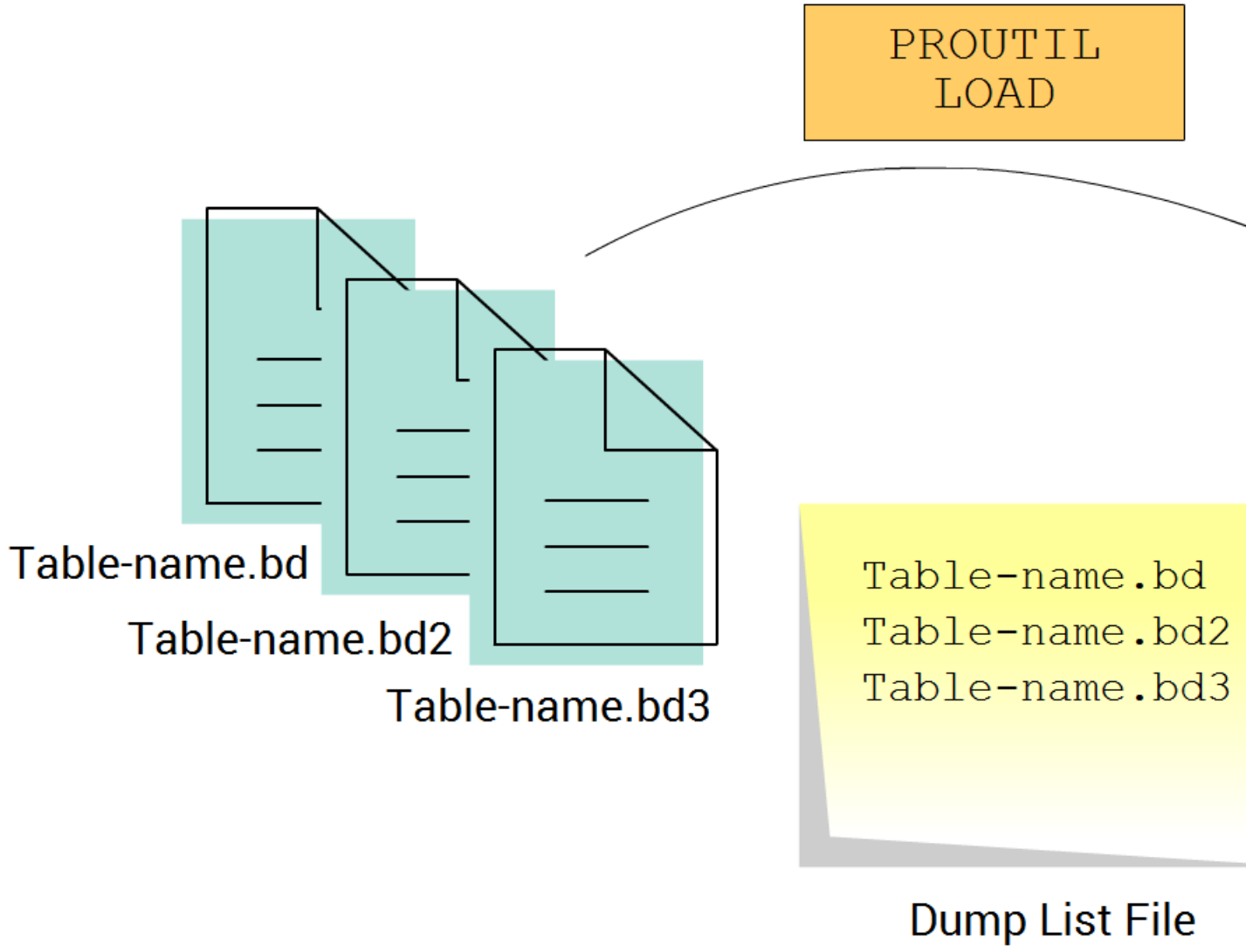
```
proutil db-name –C load .bd-file
```

Note: PROUTIL LOAD allows you to simultaneously load contents and build indexes of a target table. As a best practice, you should not build indexes when you invoke PROUTIL LOAD. Instead, you should build indexes after the load.

To use PROUTIL LOAD to load the customer.bd file in the data directory into the Customer table, enter:

```
proutil newsports –C load ../data/customer.bd
```

Loading table contents using a binary load with a dump list file



Recall that when you perform a multi-threaded binary dump of a table, you can use the –DUMPLIST parameter in PROUTIL DUMP to specify a dump list file. A dump list file is generated, listing all .bd files that the dump created. This file can be used as an input parameter for a binary load of those .bd files into the target database.

If you used a multi-threaded binary dump to dump a table into multiple .bd files and generated a dump list file in the process, you can load those .bd files into a target database using binary load with the same dump list file. This ensures that all the .bd files are loaded.

This method enables you to leverage your multiprocessor system to load data into very large tables quickly. To speed up the load, you can also use batch files to binary-load data into multiple tables in parallel.

Using PROUTIL to binary-load with a dump list file

You use PROUTIL LOAD with the –DUMPLIST qualifier to perform binary load with a dump list file.

When you work with a dump list file, follow these guidelines:

- Do not edit the dump list file to include the .bd files of other tables because the –DUMPLIST parameter works only with the .bd files of the table that was dumped using a multi-threaded binary dump.
- For the load to work, always specify a .bd file after the LOAD qualifier, even though that file may already be listed in the dump list file.

The command to binary-load table contents with a dump list file online or offline is:

```
proutil db-name –C load .bd-file –dumplist dumplist-file
```

Assume that you have performed a multi-threaded binary dump of the Orderline table in the sports database and generated a dump list file, orderline.txt, that looks like this:

```
C:\OpenEdge\WRK\dba\db/orderline.bdC:\OpenEdge\WRK\dba\db/orderline.bd2C:\OpenEdge\WRK\dba\db/orderline.bd3
```

To invoke PROUTIL LOAD with the –DUMPLIST qualifier to binary-load contents from the three listed .bd files into Orderline of the newsports database using orderline.txt, enter:

```
proutil newsports –C load orderline.bd -dumplist orderline.txt
```

Procedure: Loading sequence values using a Data tool

After you load all table contents into a target database, you must load the current sequence values into the sequences of the database. This ensures that the sequences in the target database begin with the same values as those in the source database.

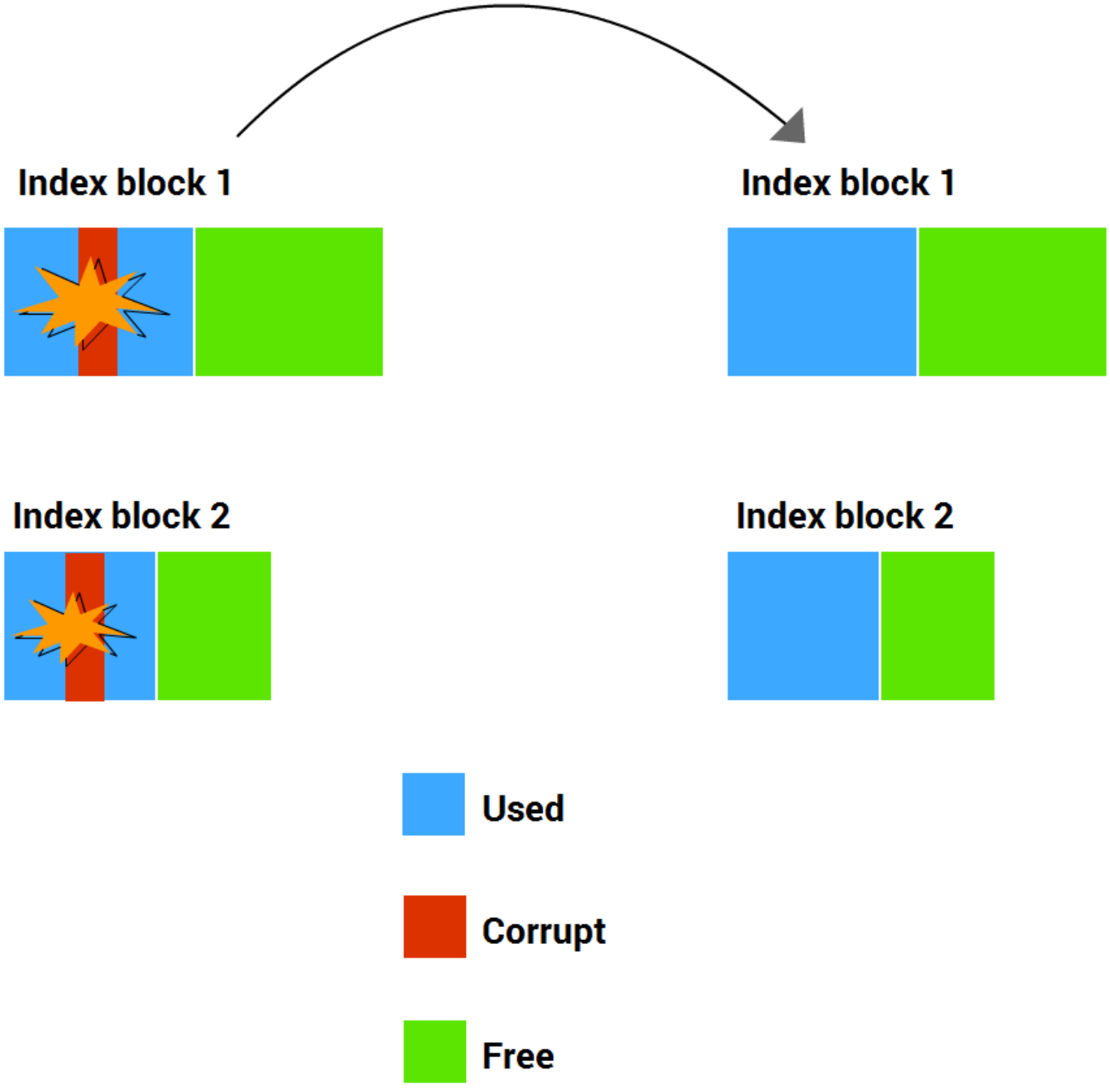
Follow these steps to load sequence values into a target database online or offline, using a Data tool:

Steps	Action
1	Launch the Data Administration tool in Windows or the Data Dictionary tool in UNIX.
2	Connect to the target database.
3	Click A**dmin > Load Data and Definitions > Sequences Current Values**. The Load Sequence Current Values dialog box appears.
4	By default, _**seqvals.d** is displayed as the input file. If the sequence values file that you want to load is in another location, enter a fully qualified filename in the Input File field or click Files and then specify the location and filename.

Steps	Action
5	Click **OK** in the Load Sequence Current Values dialog box to load the sequence values into your target database. After the sequence values are loaded, the Information box confirms that the load of the sequence values is completed.
6	Click OK to close the Information box.

7

Rebuilding indexes



After you load data definitions, table contents, and sequence values into the target database, you need to rebuild indexes using the PROUTIL IDXBUILD utility offline.

To rebuild indexes, you perform the following tasks:

1. Shut down the target database.
2. Back up the database. PROUTIL IDXBUILD does not have crash recovery built into it. In the event of a crash, you would have to restart the index rebuild. However, if the index rebuild fails to restart, you can restore the database from the most recent backup and start the rebuild again.
3. Invoke PROUTIL IDXBUILD.

Next, you will learn how to rebuild indexes using PROUTIL IDXBUILD.

For details, see the following topics:

- Using the PROUTIL IDXBUILD utility
- Allocating memory for sorting and merging indexes
- Specifying sort locations for index sorting
- Procedure: Rebuilding indexes using PROUTIL IDXBUILD

Using the PROUTIL IDXBUILD utility

The PROUTIL IDXBUILD utility builds indexes from scratch. It builds all index entries for each index, sorts them, and creates an index tree based on those index entries. It also activates indexes in the database.

The command to build indexes is:

```
proutil db-name -C idxbuild
  [ all | activeindexes | inactiveindexes |
    table [owner-name.]table-name ] |
  [ partition partition-name | composite initial] |
    area area-name |  schema schema-owner] |
  [ tenant tenant-name| group group-name | shared ]
  [ -thread n ][ -threadnum n ]
  [ -datascanthreads n ][ -mergethreads n ]
  [ -T dir-name | -SS sort-file-directory-specification ]
  [ -TB blocksize ][ -TM n ][ -TMB n ]
  [ -TF n ][ -freeTF n ][ -B n ]
  [ -SG n ]
  [ -pfactor n ][ -rusage]
```

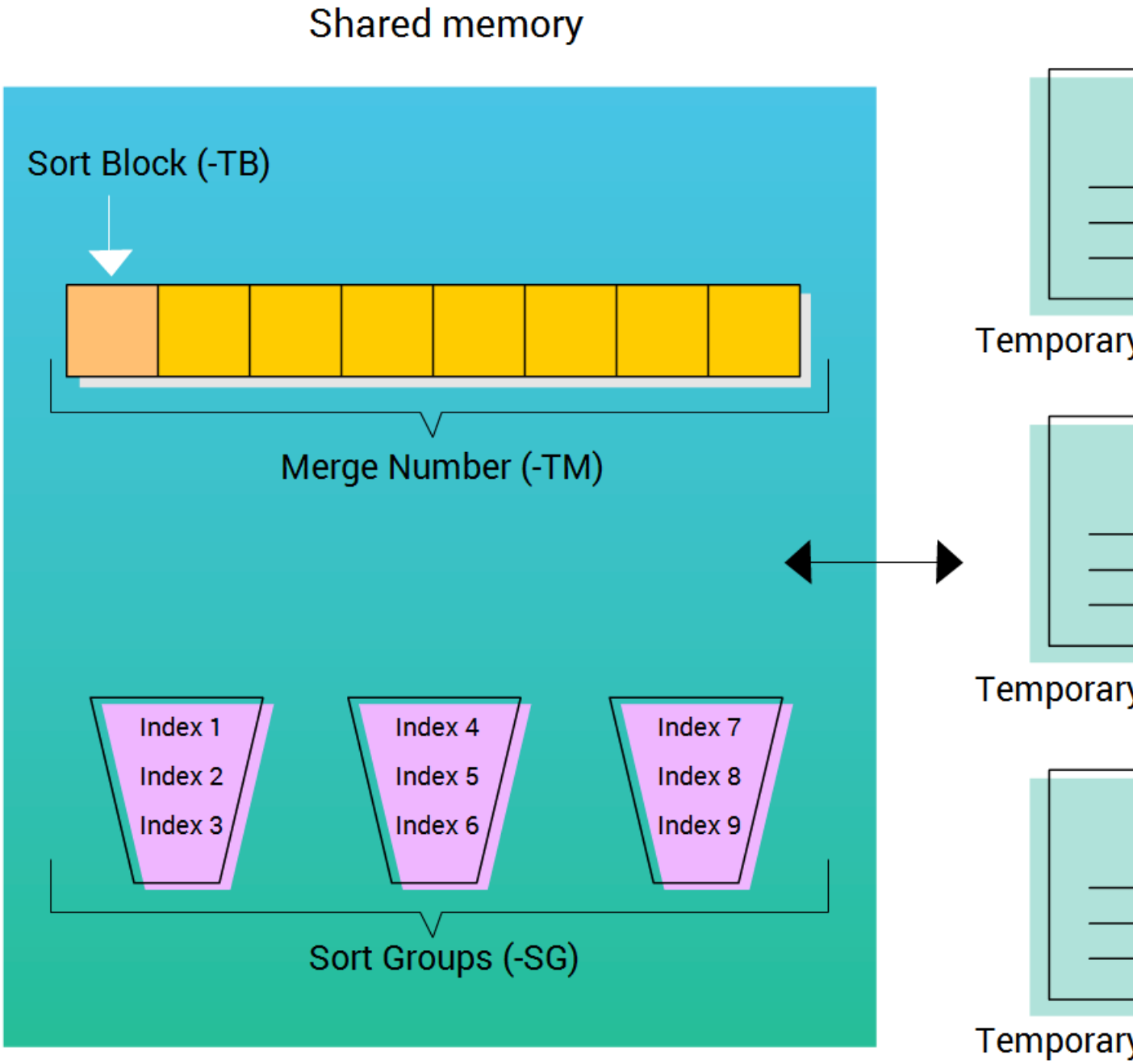
The PROUTIL IDXBUILD parameters allow you to:

- Allocate memory for sorting and merging indexes.
- Control multi-threading and the number of threads used by IDXBUILD.
- Specify sort locations for index sorting.
- Report usage statistics

Note: If you have an OpenEdge RDBMS Enterprise license and your database runs on a multiprocessor system, the index rebuild runs in a multi-threaded mode by default.

Next, you will learn about allocating memory and specifying sort locations for index sorting. Then, you will learn how to rebuild indexes.

Allocating memory for sorting and merging indexes

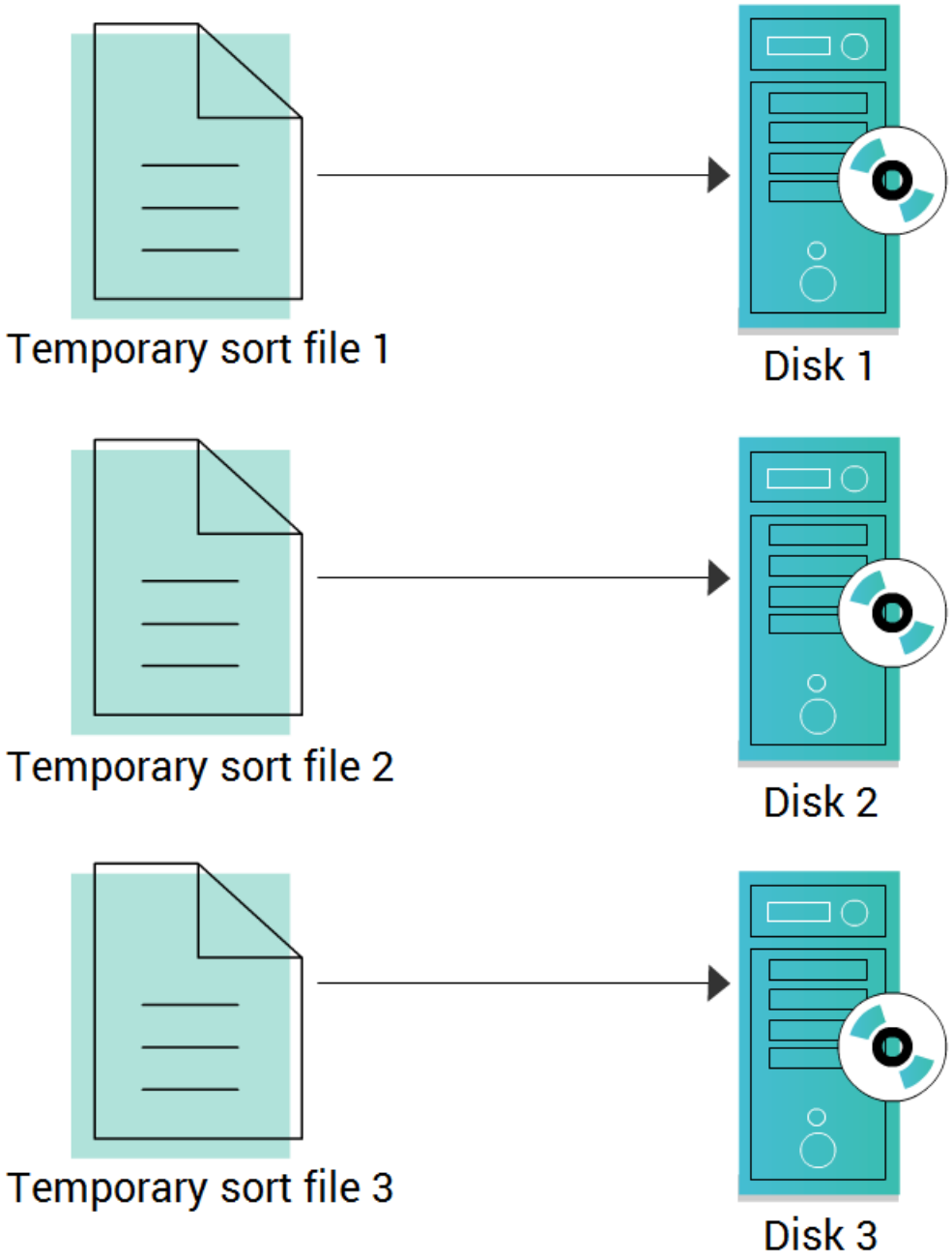


Sorting and merging indexes is the most resource-intensive phase of an index rebuild. Setting the PROUTIL IDXBUILD parameters that allocate memory for sorting and merging indexes properly enables PROUTIL IDXBUILD to do its work more quickly.

The following table lists the PROUTIL IDXBUILD parameters to allocate memory for sorting and merging indexes with their description:

Parameter	Description
-TB (Sort Block)	When PROUTIL IDXBUILD builds new index entries, it places them in sort blocks in memory. -TB specifies the size of the sort blocks, in kilobytes, that contain newly built index entries. The default -TB value is 8K. The bigger the size of the sort blocks, the more index entries they can contain, and the better the performance. For best performance, set -TB to 64K, the maximum sort block size.
-TM (Merge Number)	After the index entries have been built and sorted in each individual sort block, the sort blocks are merged together so that the index entries can be sorted across sort blocks. -TM specifies the number of sort blocks that can be merged together at a time. The default -TM value is 8. The bigger the -TM value, the more sort blocks can be merged at a time, and the better the performance. For best performance, set -TM to 32, the maximum number of sort blocks.
-SG (Sort Groups)	When the entries of an index have been sorted and merged, they are placed in a sort group. Each sort group may contain one or more indexes. During an index rebuild, several sort groups of indexes may be created. -SG determines the number of sort groups and is designed to help PROUTIL IDXBUILD use memory and disk space efficiently in sorting indexes. -SG affects both the allocation of memory and the creation of temporary sort files used for sorting and merging. -SG values range from 8 to 64. The default -SG value is 48. For best performance, set -SG to 64.

Specifying sort locations for index sorting



When you rebuild indexes, it is a best practice to use disk space rather than just memory to sort and merge indexes. During the index-sorting phase, temporary sort files are created for sorting and merging indexes. Generally, these sort files may take from 75% to 150% of the total database size in disk space. If you run out of disk space, the index rebuild operation aborts. To prevent such a situation, you should specify the sort locations for index sorting.

PROUTIL IDXBUILD can create temporary sort files for sorting and merging indexes in one of the following locations:

- The current working directory

- The location as specified by -T (directory name)
- The location as specified by -SS (sort file specification file)

If you do not specify a location, the utility stores temporary sort files in the working directory. Otherwise, it stores the files in the locations specified by -T or -SS.

For best performance, specify locations that are not currently being used to store BI, application data, or AI files.

Next, you will learn about guidelines for choosing sort locations for index sorting.

Guidelines for choosing sort locations

When you specify sort locations for index sorting, follow these guidelines:

If you have …	Then …
A small database and sufficient disk space	Specify nothing. The index rebuild utility by default uses the working directory.
A small database and insufficient disk space	Point -T to another disk.
A very large database and a lot of disk space on the same file system, and if your file system supports large files	Specify nothing. The index rebuild utility can create temporary sort files that are larger than 2GB.
A very large database and if your file system does not support large files	Create a sort file specification file and point -SS to it.
Insufficient disk space on one disk, and you need to spread the temporary sort files across multiple disks	Create a sort file specification file and point -SS to it.

Next, you will learn about a sort file specification file is and how to create it.

Creating a sort file specification file

A sort file specification file specifies the directories and the amount of space per directory that PROUTIL IDXBUILD can access during the index rebuild. The sort file specification file must have a name with a .srt extension, and must reside in the same directory as the .db file of the database. You should name the .srt file using the naming convention db-name.srt.

For best performance, you should use the sort file specification file to place the sort files on non-database disks and spread the sort files across multiple disks.

To create a sort file specification file, follow these steps:

1. Create a new text file.
2. In the text file, specify the sizes and directories of different disks on separate lines. Note: Do not separate text lines with blank lines.

3. Separate the sizes from the directories by at least one space.
4. For each file size, specify the maximum size in kilobytes. Use 0 (zero), if you want the file to expand up to your OS file-size limit.

Note: Before running the index rebuild, ensure that the sizes you specify in the sort file specification file are indeed available. Otherwise, the rebuild will abort when it encounters a full disk. The next directory is not used, even if it has enough space.

Suppose you want to rebuild indexes for the sports database and you want the index-sorting phase to have access to 1GB of space on the first disk, 1GB on the second disk, and space up to your OS file-size limit on the third disk. Then you should create a sort file specification file that looks like this:

In Windows	In UNIX
`1024000 c:\temp1024000 d:\temp0 e:\temp`	`1024000 /oetmp1/1024000 /oetmp2/0 /tmp/`

Procedure: Rebuilding indexes using PROUTIL IDXBUILD

Now that you understand the PROUTIL IDXBUILD syntax, you are ready to rebuild indexes.

Follow these steps to rebuild indexes for a database offline using PROUTIL IDXBUILD:

Steps	Action
1	Make sure that you have a valid backup.
2	Create a sort file specification file for index sorting for the database. See Creating a sort file specification file on page 51.

3	Create a parameter (.pf) file to hold the parameter values of PROUTIL IDXBUILD. For example:
	`-B 32000`
	`-TB 64`
	`-TM 32`
	`-TMB 128`
	`-TF 60`
	`-SG 64`
	`-thread 1`
	`-threadnum 4`
	`-mergethreads 4`
	`-datascanthreads 4`
	`-pfactor 90`
	`-rusage`
	`-SS <path>pcs-base.srt`

4	Shut down the database.

Steps	Action
5	Back up the database.

6	Run IDXBUILD using the .pf file.
	`proutil -db dbname -C idxbuild all -pf parameter-file.pf`
	At the end of the rebuild, you should see this message:
	`Index rebuild complete. 0 error(s) encountered. (1644)`

6	Back up the database.