Quick answers to common problems

IBM DB2 9.7 Advanced Administration Cookbook

Over 100 recipes focused on advanced administration tasks to build and configure powerful databases with IBM DB2

Adrian Neagu Robert Pelletier

[PACKT] PUBLISHING enterprise professional expertise distilled

IBM DB2 9.7 Advanced Administration Cookbook

Over 100 recipes focused on advanced administration tasks to build and configure powerful databases with IBM DB2

Adrian Neagu

Robert Pelletier

[PACKT] PUBLISHING enterprise professional expertise distilled

BIRMINGHAM - MUMBAI

IBM DB2 9.7 Advanced Administration Cookbook

First published: March 2012

Production Reference: 1200212

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-332-6

www.packtpub.com

Cover Image by Sandeep Babu (sandyjb@gmail.com)

Credits

Authors

Adrian Neagu

Robert Pelletier

Reviewers

Nadir Doctor

Marius Ileana

Nivasreddy Inaganti

Nitin G. Maker

Drazen Martinovic

Eldho Mathew

Acquisition Editor

Rukshana Khambatta

Lead Technical Editor

Hithesh Uchil

Technical Editor

Arun Nadar

Project Coordinator

Leena Purkait

Copy Editor

Brandt D'Mello

Proofreader

Aaron Nash

Indexer

Monica Ajmera Mehta

Production Coordinator

Shantanu Zagade

Cover Work

Shantanu Zagade

About the Authors

Adrian Neagu has over 10 years of experience as a database administrator, mainly with DB2 and Oracle databases. He has been working with IBM DB2 since 2002.

He is an IBM DB2 Certified Administrator (versions 8.1.2 and 9), Oracle Database Administrator Certified Master 10g, Oracle Certified Professional (9i and 10g), and Sun Certified System Administrator Solaris 10. He is an expert in many areas of database administration, such as performance tuning, high availability, replication, and backup and recovery.

In his spare time, he enjoys cooking, taking photos, and catching big pikes with huge jerkbaits and bulldawgs.

I would like to give many thanks to my family, to my daughter Maia-Maria, and my wife Dana, who helped and supported me unconditionally, and also to my colleagues, my friends, to Rukshana Khambatta, my acquisition editor, for her patience, and finally to Robert Pelletier and Marius Ileana, who have provided invaluable advice, helping me to climb up the cliffs of authoring.

Robert Pelletier is a Senior DBA Certified Oracle 8i, 9i, 10g, and DB2. He has 12 years of experience as DBA, in production/development support, database installation and configuration, and tuning and troubleshooting. He has more than 30 years of IT experience in application development in mainframe central environments, client-server, and UNIX. More recently, he has added expertise in Oracle RAC 11gR2, 10gR2, 9i, DB2 UDB DBA, ORACLE 9iAS, Financials, PeopleSoft, and also SAP R/2 & R/3. He is renowned for his expertise among many major organizations worldwide and has a solid consulting background in well-known firms.

I would like to thank my wife, Julie, and son, Marc-André, for their positive and unconditional support, and also to Adrian Neagu, who helped me a lot for coauthoring this book, and all the Packt publishing team for making this possible. I would also like to thank my clients and colleagues who have provided invaluable opportunities for me to expand my knowledge and shape my career.

About the Reviewers

Marius Ileana is an OpenGroup Certified IT specialist currently working in banking industry.

Working for six years in IBM Romania as a part of middleware team and also being a two-year support specialist, he has been involved in various IBM-related technologies and enterprise grade deployments.

He holds many IBM certifications including IBM Certified DBA for DB2 9 on LUW. Since Java development is one of his hobbies, he is also a Sun Certified Programmer for Java™ v1.4. His areas of expertise include AIX, HACMP, WebSphere Application Server, DB2 UDB, and design and development of J2EE™ applications.

His current focus areas include the architecture and development of a general-purpose monitoring solution, Portal solutions, and data visualization.

Nitin G. Maker is an IBM Certified DB2 UDB DBA with around 11 years of IT experience, primarily in IBM DB2 Universal Database Technologies. He has demonstrated excellent capabilities in various roles as Data Architect/Database Administrator/DataWarehouse Architect, Applications Administrator, Upgrade Specialist, and Technical Team Leader.

Nitin has worked with many leading software houses in India and also completed assignments in the USA, UK, and Sri Lanka. He is currently based in Pune, with his family, and enjoys making new friends, listening to music, and following sports.

Drazen Martinovic graduated at the Faculty of Electronics, Machinery and Shipbuilding, Split, Croatia, in 1996. He worked in DHL international d.o.o. as a Unix administrator—IT support administrator—for 11 years. He then started to work as a database administrator for DB2 for LUW. He has been an IBM Certified Database Administrator (DB2 9 for Linux, UNIX, and Windows), since last year.

He works in the Raiffeisenbank Austria d.d. Zagreb bank as a Database Administrator for DB2. It has over 2000 employees.

Eldho Mathew is a DB2 LUW, Linux and AIX certified administrator with 8 years of proven expertise in various aspects of building, administrating, and supporting highly complex 24x7 operational and warehouse database servers. He has handled highly complex and critical systems for many top branded customers in UK.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following `@PacktEnterprise` on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface

IBM DB2 LUW is a leading relational database system developed by IBM. DB2 LUW database software offers industry leading performance, scale, and reliability on your choice of platform on various Linux distributions, leading Unix systems, such as AIX, HP-UX, and Solaris, and also MS Windows platforms. With lots of new features, DB2 9.7 delivers one the best relational database systems on the market.

IBM DB2 9.7 Advanced Administration Cookbook covers all the latest features with instance creation, setup, and administration of multi-partitioned databases.

This practical cookbook provides step-by-step instructions to build and configure powerful databases, with scalability, safety, and reliability features, using industry standard best practices.

This book will walk you through all the important aspects of administration. You will learn to set up production-capable environments with multi-partitioned databases and make the best use of hardware resources for maximum performance.

With this guide, you can master the different ways to implement strong databases with high-availability architecture.

What this book covers

Chapter 1, *DB2 Instance—Administration and Configuration*, covers DB2 instance creation and configuration for non-partitioned database and multipartitioned database environments.

Chapter 2, *Administration and Configuration of the DB2 Non-partitioned Database*, contains recipes that explain how to create a database and get operational in simple and easy steps. In this chapter, you will also learn how to configure your database for its mission and prepare it for automatic maintenance, so its operation is worry-free.

Chapter 3, *DB2 Multipartitioned Databases—Administration and Configuration*, contains recipes that explain how to create and configure a multipartitioned database and its related administration tasks. This chapter will also teach us how to add and remove new partitions, how to perform add, remove, and redistribute operations on database partition groups, and much more.

Chapter 4, *Storage—Using DB2 Table Spaces*, covers physical aspects of storage, the foundation of a database. In this chapter, we will cover configuring SMS and DMS table spaces, altering table spaces, and dropping table spaces.

Chapter 5, *DB2 Buffer Pools*, covers caching. Here, you will learn how data is read from the disk, to buffer pools. And as reading from memory is faster than reading from disk, the buffer pools play an important part in database performance.

Chapter 6, *Database Objects*, covers Multidimensional Clustering (MDC), Materialized Query Tables (MQT), and Partitioning as the key techniques used for efficient data warehousing. Combined with database partitioning, these deliver a scalable and effective solution, reduce performance problems and logging, and provide easier table maintenance.

Chapter 7, *DB2 Backup and Recovery*, covers the major aspects of backup and recovery, as is practiced industry-wide, the preferred solutions, and how we can implement some of these methods.

Chapter 8, *DB2 High Availability*, mainly covers High Availability Disaster Recovery as a HA solution and DB2 Fault Monitor, which is used for monitoring and ensuring the availability of instances that might be closed by unexpected events, such as bugs or other type of malfunctions. The reader will learn how to implement HADR using command line and Control Center, about synchronization modes, how to initiate takeover and takeover by force, how to configure and open a standby database in read-only mode, and more.

Chapter 9, *Problem Determination, Event Sources, and Files*, has recipes for various tools used for diagnostics, inspection, and performance problem detection, such as `db2mtkr`, for gathering memory-related information, `db2pd`, a very powerful tool used for problem determination, `db2dart`, also a very powerful tool with wide applicability, that can be used for virtually any problem that may arise, `db2ckbkp`, for backup image checking, and `db2support`, used mainly for automating diagnostic data collection.

Chapter 10, *DB2 Security*, speaks about the main security options used to harden and secure DB2 servers. It is about instance-level and database authorities, data encryption, roles, and securing and hiding data using Label Based Access Control.

Chapter 11, *Connectivity and Networking*, covers many network-related configurations that apply to DB2 servers and clients, such as node cataloging, setting up connections to DRDA serves, and how to tune and monitor the Fast Communication Manager.

Chapter 12, *Monitoring*, covers an important part of a DBA's work, ensuring the database is available and that nothing hinders its functionality.

Chapter 13, *DB2 Tuning and Optimization*, provides general guidelines, as well as insightful details, on how to dispense the regular attention and tuning that databases need, using a design-centered approach. Our tips, based on best practices in the industry, will help you in building powerful and efficient databases.

Chapter 14, *IBM pureScale Technology and DB2*, represents mainly an introduction to pureScale technology. We will cover the principal administration tasks related to members, instances, and caching facilities. The reader will also learn about monitoring, backup and recovery methods, and special features that exist only in pureScale configurations.

What you need for this book

Unless you have access to a facility that has DB2 installed, you can install a trial version of DB2 on your own PC for learning purposes. Make sure you have the required hardware and operating system.

We must stress the importance of using a sandbox environment in order to duplicate the recipes in this book. Some recipes are intended for demonstration purposes and should not be done in a production environment.

Who this book is for

If you are a DB2 Database Administrator who wants to understand and get hands-on with the underlying aspects of database administration, then this book is for you.

This book assumes that you have a basic understanding of DB2 database concepts, and sufficient proficiency in the Unix/Linux operating system.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "Partitioned indexes facilitate data maintenance by making `rollin` and `rollout` operations easier."

A block of code is set as follows:

```
SELECT DISTINCT
  STORE, INTEGER(SALESDATE)/100
FROM POS.SALES
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
db2 "CREATE TABLE POSP.MQT_REFTBLS AS ( … )
 ...
 MAINTAINED BY SYSTEM
 DISTRIBUTE BY REPLICATION"
```

Any command-line input or output is written as follows:

```
CREATE GLOBAL TEMPORARY TABLE TMP_INVCDET
LIKE POSP.INVCDET
ON COMMIT DELETE ROWS
NOT LOGGED
IN POSTEMP8K;
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Navigate to **Database partition groups**, right-click, and choose **Create...**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
DB2 Instance—Administration and Configuration

In this chapter, we will cover:

- Creating and configuring instances for non-partitioned environments
- Creating and configuring a client instance
- Creating and configuring instances for multipartitioned environments
- Starting and stopping instances
- Configuring SSL for client-server instance communication
- Listing instances
- Attaching to instances
- Dropping instances

Introduction

The main focus of this chapter is DB2 instance creation and configuration, for non-partitioned database and for multipartitioned database environments.

Creating and configuring instances for non-partitioned environments

A DB2 instance can be defined as a logical container or as a logical context for databases. It can also be described as a layer between DB2 software binaries, a database, and its objects. Also it provides a level of isolation between databases; for example, it is possible to have two or more databases on the same environment, with the same name, but under different instances. It also provides and ensures the communication layer between clients and databases.

Getting ready

For this recipe (and almost all recipes in this book), we will use two servers running Red Hat Enterprise Linux Server x64 release 5.5 (Tikanga), named `nodedb21` and `nodedb22`. The hostnames are optional, but our recommendation is to set up an identical environment to avoid confusion during reading and applying the recipes.

As install location for the IBM DB2 9.7 Enterprise Server Enterprise software product, we will use the directory `/opt/ibm/db2/V9.7` on `nodedb21`. On `nodedb22`, we will install DB2 Client software to location `/opt/ibm/db2/V9.7_clnt`. The instance owner will be `db2inst1` on `nodedb21` and `db2clnt1` as client instance owner on `nodedb22`. Also, on `nodedb21`, we will create a second instance owner user named `db2inst2`, to demonstrate how to create an instance manually.

How to do it...

The default method to create an instance is during the IBM DB2 9.7 Enterprise Server Edition software installation. The other possible option is to use the `db2icrt` command.

In Linux and Unix, every instance is created under a dedicated user, called the instance owner. To create an instance in Linux and UNIX you have to be the `root` user; on these platforms, we are limited to one instance per user. On Microsoft Windows platforms, you may have more than one instance created under the same user.

Usually, if you set up the software in graphical mode you do not have to create the users manually—you can do this using the wizard. In our recipes, we want to reuse the same groups (`db2iadm1` and `db2fadm1`) for the non-partitioned and the multipartitioned instance and database setup. For the multipartitioned setup we will have the same groups defined on both servers; because we have to deal with security regarding permissions, here, we should create the groups with the same group ID (GID):

1. Create primary groups with the same GID on both servers:

```
[root@nodedb21 ~]# groupadd -g 1103 db2iadm1
[root@nodedb21 ~]# groupadd -g 1102 db2fadm1
[root@nodedb21 ~]#
```

```
[root@nodedb22 ~]# groupadd -g 1103 db2iadm1
[root@nodedb22 ~]# groupadd -g 1102 db2fadm1
[root@nodedb22 ~]#
```

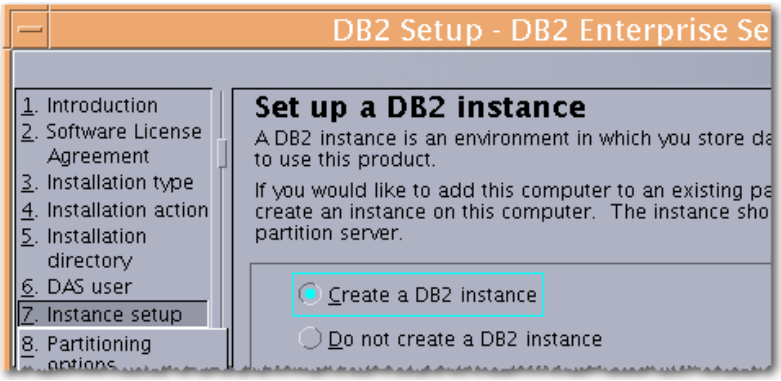
Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

2. Run `db2setup` from the IBM DB2 9.7 Enterprise Server Edition software installation kit.

Instance owner user `db2inst` and fenced user `db2fenc` will be created during installation. The groups `db2iadm1` and `db2fadm1` will automatically fill in on the screen.

3. To create a new instance during the installation with `db2setup` in graphical mode, navigate through configuration steps 1 to 6 and, at step 7 you will find **Create a DB2 instance** option checked; this is the default option.and let as it is.Click **Next**.



4. At step 8—**Partitioning options**—you will find **Single partition instance** option checked ; this is the default option and let as it is. Click **Next** and finalize installation. If installation was successful, we have a new instance named `db2inst1` created.

   Another way to create an instance is to use the `db2icrt` command. This method is suitable in the case that you install the DB2 software with `db2_install` (manual installation), or that you do not check the **Create a DB2 instance** option during installation with `db2setup`. Other scenarios would be if you drop an instance and want to create a new one, or if you want to create an additional instance.

5. As mentioned previously, in Linux and Unix, every instance has to be created under an instance owner user. As a `root` user, we will create the user `db2inst2` as instance owner and `db2fenc2` as fenced user; set passwords identical to the individual usernames:

```
[root@nodedb21 ~]# useradd -g db2iadm1 db2inst2
[root@nodedb21 ~]# useradd -g db2fadm1 db2fenc2
[root@nodedb21 ~]# passwd db2inst2
Changing password for user db2inst2.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb21 ~]# passwd db2fenc2
Changing password for user db2fenc2.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb21 ~]#
```

6. At this step, set the communication protocol to TCP/IP. The instance communication protocol is set up using the `DB2COMM` variable. We can set this variable no protocol managers will be started and will lead to communication errors at the client side.

```
[db2inst2@nodedb21 ~]$ db2set DB2COMM=TCPIP
[db2inst2@nodedb21 ~]$
```

7. Next, as user `root`, edit `/etc/services` and add `db2c_db2inst2 50002/tcp` entry (highlighted in bold in the listing bellow). Port `50002` will be assigned to `db2inst2` instance. Port `50001` corresponds to the `db2c_db2inst1` service name and was added at `db2inst1` instance creation. Port names prefixed with DB2 are reserved for inter-partition communication, a subject that we're going to discuss later on.

```
db2c_db2inst1   50001/tcp
DB2_db2inst1    60000/tcp
DB2_db2inst1_1  60001/tcp
DB2_db2inst1_2  60002/tcp
DB2_db2inst1_END        60003/tcp
db2c_db2inst2 50002/tcp
```

If you choose to use only port numbers for SVCENAME database manager parameter you do not need to edit this file.

8. As `root` user, create instance `db2inst2`, using the previously created users as instance owner and fenced user:

```
[root@nodedb21.~]# /opt/ibm/db2/V9.7/instance/db2icrt -a SERVER
ENCRYPT -p db2c_db2inst2 -u db2fenc2 db2inst2

DBI1070I  Program db2icrt completed successfully.

[root@nodedb21 ~]#
```

We need to explain a little bit about the options used for creating instance `db2inst2`:

- The `-a` option indicates the authentication type; the default is `SERVER`. Using the `-a` option, the following authentication modes are available: `SERVER`, `CLIENT`, and `SERVER ENCRYPT`. We may change it later by modifying the `AUTHENTICATION` or the SRVCONN_AUTH instance parameter.
- The `-u` switch is used to set the fenced user.
- The `-p` option is used to specify the port or its corresponding service name used for client communication, as defined in `/etc/services`. The port or service name may be changed later by modifying the `SVCENAME` database manager parameter
- For MS Windows platforms, we don't have the `-a` option to specify the authentication mode. The `-p` option in Windows has a different meaning; it is used to specify the instance profile. The `-u` option is for specifying the account name and password used that will be included in the Windows service definition associated with the instance.

To use the **Control Center** for managing an instance locally or remotely, you need to have DB2 Administration Server (DAS) up and running, on the server.

To check the status of DAS, execute the following command, as DAS owner user, which is in our case `dasusr1`:

```
[dasusr1@nodedb21 ~]$ db2dascfg get dasstatus
ACTIVE
[dasusr1@nodedb21 ~]$
```

Usually, it is installed and created during IBM DB2 software installation. If there is no DAS created, you should create it using the `dascrt` command. The steps are similar to those for creating an instance—create a group and a user. It has to be created by specifying the owner.

For example, `/opt/ibm/db2/V9.7/instance/dascrt -u dasusr1`.

How it works...

In Linux or Unix, when an instance is created, the `db2icrt` command *builds up* under the instance owner home directory, the `sqllib` directory, as a collection of symbolic links pointing to the IBM DB2 software installation home directory. If you want to see what is executing `db2icrt` in the background, you need to include the `-d` option to enable debug mode. This explains what happens behind the scenes for the steps mentioned earlier. Usually, this switch is used for detailed diagnostics, and should be activated at the request of IBM support.

Almost all files and directories from sqllib directory are symbolic links to the corresponding installation path (DB2HOME). A short listing inside `sqllib` directory looks like this:

```
[db2inst1@nodedb21]/home/db2inst1/sqllib>symlinks -v .
other_fs: /home/db2inst1/sqllib/map -> /opt/ibm/db2/V9.7/map
other_fs: /home/db2inst1/sqllib/bin -> /opt/ibm/db2/V9.7/bin
other_fs: /home/db2inst1/sqllib/ruby64 -> /opt/ibm/db2/V9.7/dsdriver/
ruby64
```

On MS Windows platforms, the `db2icrt` command creates a service. The binaries are actually copied and a service associated with the instance is created.

On a generic Windows machine we'll create an instance named `db2win`. Initially, the associated service has the status set to stopped and the startup type set to manually. If you want the service to start automatically at system boot, you have to change its startup type to automatic.

To create instance db2win, execute the following command under a privileged user:

```
C:\Windows\system32>db2icrt db2win
DB20000I  The DB2ICRT command completed successfully.
```

To find the associated Windows service with db2win instance, execute the following command:

```
C:\Windows\system32>sc query state= all  | findstr "DB2WIN"
SERVICE_NAME: DB2WIN
DISPLAY_NAME: DB2 - DB2COPY1 - DB2WIN
C:\Windows\system32>
```

There's more...

The `db2isetup` graphical tool might be used also for creating instances; this tool is available only on the Linux and Unix platforms.

On Linux and Unix you have the possibility to create a non-root type instance using the installer. You are limited to only one non-root instance per server.

Updating instances using the db2iuptd command

Usually this command is used to update an instance after an upgrade to a higher version, or migrate an instance from a lower product level such as Workgroup Edition to Enterprise Edition. Also it might be used for instance debug using the `-d` option. Like `db2icrt`, this command has its own particularities on MS Windows operating systems. To find the available options and related descriptions of this command issue `db2iuptd -h`. For non-root type instances exists a variant of this command named `db2nruptd`.

Creating and configuring a client instance

Usually, this special type of instance is used for cataloging nodes and databases to which you want to connect using this client. Compared to server instances there are some limitations, as it cannot be started or stopped, and you cannot create databases under it. Mainly, it is used by the DB2 Client and DB2 Connect products.

Getting ready...

On `nodedb22` we will create the instance owner `db2clnt1` and fenced user named `db2fenc1`. For creating a client instance, we'll use the `-s` option of the `db2icrt` command.

How to do it...

1. Install DB2 Client in the `/opt/ibm/db2/V9.7_clnt` location on `nodedb22`, without creating an instance; to do this during installation, check at step 6—**Instance setup**—*Defer this task until after installation is complete*.
2. Next, create users on `nodedb22`—`db2clnt1` as the client instance owner and `db2fenc1` as fenced user—and set passwords identical to the usernames:

```
[root@nodedb22 ~]# useradd -g db2iadm1 db2clnt1
[root@nodedb22 ~]# useradd -g db2fadm1 db2fenc1
[root@nodedb22 ~]# passwd db2clnt1
Changing password for user db2clnt1.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb22 ~]# passwd db2fenc1
Changing password for user db2fenc1.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb22 ~]#
```

3. As user `root`, create the client instance `db2clnt1`:

```
[root@nodedb22 ~]# /opt/ibm/db2/V9.7/instance/db2icrt -s client -u
db2fenc1 db2iclnt1
DBI1070I  Program db2icrt completed successfully.
[root@nodedb22 ~]#
```

How it works...

Mainly you need to setup a client instance when you have plans to administer DB2 servers remotely with tools that are using non-Java based connections such as **Control Center** or Toad for DB2. The same scenario is applicable when you are using CLI for remote administration or command execution and also in this category are non-java based application clients.

There's more...

In the previous section we used the term non-java clients. However, this not totally exact for older type JDBC or JDBC-ODBC bridge connections using type 1 and 2 drivers. Type 3 and 4 JDBC drivers have implemented internally the entire network communication stack; this is the main reason for their independence from client instances and external network libraries. A good example for a tool that is relying only on JDBC type connections is the new Optim Database Administrator recommended by IBM to be used in future for database administration.

See also

The *Communication with DRDA servers (z/OS and i/OS)* recipe in *Chapter 11*, *Connectivity and Networking*

Creating and configuring an instance for multipartitioned environments

The IBM DB2 database multipartitioned feature offers the ability to distribute a large database onto different physical servers or the same SMP server, balancing the workload onto multiple databases that are working as one, offering a very scalable way of data processing. We may have all the database partitions reside on the same server, this method of database partitioning is called logical partitioning. There is another scenario when the database partitions are spanned on different physical servers; this partitioning method is called physical partitioning.

An instance in a multipartitioned configuration is not very different by a non-partitioned instance, if it is running on a logical partitioning scheme. To use only physical partitioning, or physical partitioning combined with logical partitioning, an instance must be configured as shared across all the database partitions. In this recipe, we will use the last scenario.

The instance is created once on one node; on the other participant nodes, you have to create just the instance owner user with the same user ID (UID) and GIDs and the same home directory as on the instance owner node. In the following recipe, we will configure servers for the purpose of multipartitioning and will create a new instance named `db2instp`.

Notice that in this recipe we will use node and partition termsinterchangeably

Getting ready

To install a multipartitioned instance, we need to prepare a suitable environment. For this recipe, we will use the two Linux servers named `nodedb21` and `nodedb22`, mentioned before. `nodedb21` will contain the instance home and will export it through NFS to the `nodedb22` system. We will also use a new disk partition, defined on `nodedb21`, for instance home `/db2partinst`, which, in our case, is a Linux LVM partition. We will create users on both servers with the same UID, and will install IBM DB2 ESE in a new location or DB2HOME—`/opt/ibm/db2/V9.7_part` on `nodedb21` with the **create a response file** option. On `nodedb22`, we'll also install IBM DB2 ESE, in the location `/opt/ibm/db2/V9.7_part`, using the response file created during installation on `nodedb21`.

How to do it...

1. Because this is not a Linux book, we do not cover how to install NFS or how to create a new Linux partition. As a preliminary task, you should check if you have NFS and portmap installed and running on both servers.
2. As user `root`, execute the following commands on both servers:

   To check if we have NFS and portmap on `nodedb21`:

```
[root@nodedb21 ~]# rpm -qa | grep nfs
nfs-utils-lib-1.0.8-7.6.el5
nfs-utils-1.0.9-44.el5
[root@nodedb21 ~]# rpm -qa | grep portmap
portmap-4.0-65.2.2.1
[root@nodedb21 ~]#
```

   To check their current status on `nodedb21`:

```
[root@nodedb21 ~]# service nfs status
rpc.mountd (pid 3667) is running...
nfsd (pid 3664 3663 3662 3661 3660 3659 3658 3657) is running...
rpc.rquotad (pid 3635) is running...
[root@nodedb21 ~]#
[root@nodedb21 ~]# service portmap status
portmap (pid 3428) is running...
[root@nodedb21 ~]#
```

Set up NFS for sharing the instance home

1. To automatically export `/db2partinst` on system boot, add your hostnames or the corresponding IP numbers to the `/etc/exports` file. On `nodedb21`, add the following line in `/etc/exports`:

```
/db2partinst       10.231.56.117(rw,no_root_squash,sync)
  10.231.56.118(rw,no_root_squash,sync)
```

2. To export the partition immediately, execute the following command:

```
[root@nodedb22 ~]# exportfs –ra
[root@nodedb22 ~]#
```

3. On `nodedb22`, as user `root`, create a directory `/db2partinst`, used as mount point for `/db2partinst`, exported from `nodedb21`:

```
[root@nodedb22 ~]# mkdir /db2partinst
[root@nodedb22 ~]#
```

4. In `/etc/fstab` on `nodedb22`, to mount `/db2partinst` on system boot, add the following line:

```
nodedb21:/db2partinst /db2partinst nfs
  rw,timeo=300,retrans=5,hard,intr,bg,suid
```

5. To mount the partition immediately on `nodedb22`, issue the following command:

```
[root@nodedb22 ~]# mount nodedb21:/db2partinst /db2partinst
[root@nodedb22 ~]#
```

Creating the instance owner and fenced user

1. On `nodedb21`, create the instance owner `db2instp` and the fenced user `db2fencp`. Instance home will be located in `/db2partinst/db2instp`:

```
[root@nodedb22 ~]# useradd -u 1316 -g db2iadm1 -m -d /db2partinst/
db2instp db2instp
[root@nodedb22 ~]# useradd -u 1315 -g db2fadm1 -m -d /db2partinst/
db2fencp db2fencp
[root@nodedb22 ~]# passwd db2instp
Changing password for user db2instp.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb21 ~]# passwd db2fencp
Changing password for user db2fencp.
New UNIX password:
```

```
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@nodedb21 ~]#
```

2. Repeat step 1 on `nodedb22` and ignore any warnings.

Set up SSH for client authentication

In a physical multipartitioned environment, any instance owner user has to be able to execute commands on any participant node. To ensure this, we need to establish user equivalence or host equivalence between nodes. Actually, we have two methods: one is with RSH, which is less secure and the other is using SSH, which is secure. With SSH, there are two methods: one is host-based authentication and the other is client-based authentication. Next, we will implement client-based authentication; this method fits better with a small number of partitions, as in our example.

1. As user `db2instp` on `nodedb21`, execute the following commands:

```
[db2instp@nodedb21 ~]$ cd ~
[db2instp@nodedb21 ~]$ mkdir .ssh
[db2instp@nodedb21 ~]$ chmod 700 .ssh
[db2instp@nodedb21 ~]$ cd .ssh
[db2instp@nodedb21 .ssh]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/db2partinst/db2instp/.ssh/
id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /db2partinst/db2instp/.ssh/
id_rsa.
Your public key has been saved in /db2partinst/db2instp/.ssh/id_
rsa.pub.
The key fingerprint is:
2b:90:ee:3b:e6:28:11:b1:63:93:ba:88:d7:d5:b1:14 db2instp@nodedb21
[db2instp@nodedb21 .ssh]$ cat id_rsa.pub >> authorized_keys
[db2instp@nodedb21 .ssh]$ chmod 640 authorized_keys
```

2. As user `db2instp` on `nodedb22`, execute the following commands:

```
[db2instp@nodedb22 .ssh]$ cd ~/.ssh
[db2instp@nodedb22 .ssh]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/db2partinst/db2instp/.ssh/
id_rsa):
```

```
/db2partinst/db2instp/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /db2partinst/db2instp/.ssh/
id_rsa.
Your public key has been saved in /db2partinst/db2instp/.ssh/id_
rsa.pub.
The key fingerprint is:
87:36:b4:47:5a:5c:e5:3e:4e:e9:ce:5b:47:2c:ce:6b db2instp@nodedb22
[db2instp@nodedb22 .ssh]$ cat id_rsa.pub >> authorized_keys
[db2instp@nodedb22 .ssh]$
```

3. Go back on `nodedb21` and issue the following commands to set up a host trust relationship:

```
[db2instp@nodedb21 ~]$ cd ~/.ssh
[db2instp@nodedb21 .ssh]$ ssh-keyscan -t rsa
nodedb21,10.231.56.117 >> known_hosts
# nodedb21 SSH-2.0-OpenSSH_4.3
[db2instp@nodedb21 .ssh]$ ssh-keyscan -t rsa
nodedb22,10.231.56.118 >> known_hosts
# nodedb22 SSH-2.0-OpenSSH_4.3
[db2instp@nodedb21 .ssh]$
```

4. Verify that the client authentication is working; on `nodedb21`, issue `ssh nodedb22 date` (do it the other way around—now it should work without asking for a password):

```
[db2instp@nodedb21 .ssh]$ ssh nodedb22 date
Thu Jun  9 16:42:33 EEST 2011
[db2instp@nodedb21 .ssh]$ ssh nodedb22
[db2instp@nodedb22 ~]$ ssh nodedb21 date
Thu Jun  9 16:42:48 EEST 2011
[db2instp@nodedb22 ~]$ ssh nodedb22 date
Thu Jun  9 16:42:55 EEST 2011
[db2instp@nodedb22 ~]$ ssh nodedb21
[db2instp@nodedb21 ~]$ ssh nodedb21 date
Thu Jun  9 16:43:07 EEST 2011
[db2instp@nodedb21 ~]$
```

Install DB2 ESE software with a response file option

A response file is a text file containing installation and configuration information such as paths, installation options etc. It can be created and recorded using interactive installation and replayed by other installations to perform the same steps.

1. Launch `db2setup`, and, at step 4 of the installation wizard (**Install action**), check the **Install DB2 Enterprise Server Edition on this computer and save my setting in a response file** option. Provide the complete path to the response file.
2. At step 5, specify `/opt/ibm/db2/V9.7_part` for **Installation directory**.
3. At step 7 (**Partitioning option**), check **Multiple partition instance**.
4. Next, for DB2 instance owner, choose `db2instp` and, for fenced user, choose `db2fencp`. On the next screen, choose **Do not create tools catalog**. At the end of installation, we will find (in the directory chosen at step 4 of installation wizard) two files with `.rsp` extension; you need to copy just `db2ese_addpart.rsp` to `nodedb22` and issue on `nodedb22`, from the installation directory:

```
./db2setup -r <your path>db2ese_addpart.rsp
DBI1191I  db2setup is installing and configuring DB2 according to
the response file provided. Please wait.
```

Configuring communication for inter-partition command execution

1. The communication method of inter-partition command execution is controlled by DB2RSCHCM registry variable. Because our choice is SSH for inter-partition command execution, you must next set the `DB2RSHCMD` variable to point to SSH executable `DB2RSHCMD=/usr/bin/ssh`. If this variable is not set, the `rsh` method is used by default:

```
[db2instp@nodedb21 ~]$ db2set DB2RSHCMD=/usr/bin/ssh -i
```

2. To verify the current DB2 registry variables, issue the following command:

```
[db2instp@nodedb21 ~]$ db2set -all
[i] DB2RSHCMD=/usr/bin/ssh
[i] DB2COMM=tcpip
[i] DB2AUTOSTART=YES
[g] DB2FCMCOMM=TCPIP4
[g] DB2SYSTEM=nodedb21
[g] DB2INSTDEF=db2instp
```

Configuring the nodes

In the `db2nodes.cfg` file, database partition configuration file, located in `$INSTANCEHOME/sqllib`, set the participant nodes. Define three nodes—two on `nodedb21`, partion number `0` with logical port `0` and partition number `2` with logical port `1` and one on `nodedb22`, partition `1` with logical port `0`. After adding the nodes we should have the following structure:

```
0 nodedb21 0
1 nodedb22 0
2 nodedb21 1
```

How it works...

Instance `db2instp` knows about the current nodes by reading their definition from `db2nodes`.cfg database partition configuration file. The logical ports and number of maximum partitions per server are limited by the range defined within `/etc/services` file as follows:

```
DB2_db2inst1
60000/tcp DB2_db2inst1_1
60001/tcp DB2_db2inst1_2
60002/tcp DB2_db2inst1_END 60003/tcp
```

The structure of `db2nodes.cfg`, in some cases, can be further elaborated with optional information such as `resourcenames` or `netnames`; in our case being a simple setup used for demonstration purpose we have defined only the nodes, hostnames, and the logical ports.

Under Unix and Linux, `db2nodes` has the following complete format:

```
dbpartitionnum hostname logicalport netname resourcesetname
```

Under MS Windows, `db2nodes` has the following complete format:

```
dbpartitionnum hostname computername logicalport netname resourcesetname
```

There's more...

DB2 has two utilities to verify that communication between nodes is working: `db2_all` and rah. You can also issue practically any administrative command (backup, restore, setting parameters, and so on) across the database partitions with these utilities.

An example of using `db2_all` for verification:

```
[db2instp@nodedb21 ~]$ db2_all uptime
  11:54:02 up 17:11,       1 user,       load average: 0.07, 0.03, 0.00
nodedb21: uptime completed ok
  11:54:03 up 17:11,       0 users,      load average: 0.10, 0.03, 0.01
nodedb22: uptime completed ok
```

```
  11:54:03 up 17:11,       1 user,       load average: 0.07, 0.03, 0.00
nodedb21: uptime completed ok
```

The same using `rah`:

```
[db2instp@nodedb21 ~]$ rah uptime

  14:56:19 up 35 days, 18:09,     1 user,   load average: 0.08, 0.02, 0.01
nodedb21: uptime completed ok
  14:56:20 up 35 days, 18:09,     0 users,  load average: 0.00, 0.00, 0.00
nodedb22: uptime completed ok
  14:56:20 up 35 days, 18:09,     1 user,   load average: 0.08, 0.02, 0.01
nodedb21: uptime completed ok
```

Obviously, there is also a possibility of using a shared disk, formatted with a concurrent file system, such as, IBM's GPFS or Red Hat GFS, for instance home, and used for sharing across the nodes instead of using NFS exports.

On Windows, it is not recommended to edit the db2nodes.cfg file manually; use the

The following commands instead:

- `db2nlist`—to list database partitions
- `db2ncrt`—to add a database partition server to an instance
- `db2ndrop`—to drop a database partition server to an instance
- `db2nchg`—to modify a database partition server configuration

See also

The *Converting a non-partitioned database to a multipartitioned database on MS Windows* recipe in *Chapter 3*, *DB2 Multipartitioned Databases—Administration and Configuration*

Starting and stopping instances

There are several situations in which an instance must be stopped and started, for example, after you change some parameters that are not dynamic, or after applying a fixpack.

Getting ready

We have, at disposal, a couple of different ways to start or stop an instance. We can use, say, `db2start` for starting and `db2stop` for stopping; these commands are available for execution in the command line or from DB2 CLI. We can also start or stop an instance from the **Control Center**. In Windows, you can also start and stop an instance by starting and stopping the service associated with it.

How to do it...

1. The current instance is set by the environment variable `DB2INSTANCE` or the global registry variable `DB2INSTDEF`, in case `DB2INSTANCE` is not set. This is applicable mostly for Microsoft Windows platforms where there could be more than one instance per user.

   - On Microsoft Windows:

     ```
     C:\Documents and Settings>db2ilist
     DB2_02
     DB2WIN
     C:\Documents and Settings>set DB2INSTANCE
     DB2INSTANCE=DB2_02
     ```

   Now, if we issue `db2stop` or `db2start`, only instance `DB2_02` will be affected.

   - On our Linux server `nodedb21`:

     ```
     [db2inst1@nodedb21 ~]$ echo $DB2INSTANCE
     db2inst1
     ```

2. As the `db2inst1` instance owner, stop instance `db2inst1` with the `db2stop` command, and start it with `db2start`:

   ```
   [db2inst1@nodedb21 ~]$ db2stop
   06/09/2011 17:55:21     0   0   SQL1064N  DB2STOP processing was
     successful.
   SQL1064N  DB2STOP processing was successful.
   [db2inst1@nodedb21 ~]$ db2start
   06/09/2011 17:55:29     0   0   SQL1063N  DB2START processing was
     successful.
   SQL1063N  DB2START processing was successful.
   ```

3. As the multipartitioned instance owner `db2instp`, stop instance `db2instp` with the `db2stop` command, and start it with `db2start`:

   ```
   [db2instp@nodedb21 sqllib]$ db2stop
   06/09/2011 19:03:47     1   0   SQL1064N  DB2STOP processing was
     successful.
   06/09/2011 19:03:48     0   0   SQL1064N  DB2STOP processing was
     successful.
   06/09/2011 19:03:49     2   0   SQL1064N  DB2STOP processing was
     successful.
   SQL1064N  DB2STOP processing was successful.
   [db2instp@nodedb21 sqllib]$ db2start
   ```

```
06/09/2011 19:04:02     1   0   SQL1063N  DB2START processing was
  successful.
06/09/2011 19:04:06     2   0   SQL1063N  DB2START processing was
  successful.
06/09/2011 19:04:06     0   0   SQL1063N  DB2START processing was
  successful.
SQL1063N  DB2START processing was successful.
```

4. Using the **Control Center**, right-click on `db2inst1` and issue *stop* and *start*.

How it works...

In the process of starting an instance, memory structures are allocated and the instance starts listening for connections on the ports assigned by the `SVCENAME` database manager configuration parameter. At *stop*, existing connections are disconnected and memory is deallocated.

There's more...

Other options that can be used to start and stop an instance are the DB2 CLI commands, `START DATABASE MANAGER` and `STOP DATABASE MANAGER`. For Windows, we have as alternate option to start or stop the service associated with the instance. To set the instance for automatic start on Linux or Unix, at system boot, you can use the instance-level registry variable `DB2AUTOSTART=YES` or the `db2iauto –on <instance name>` command.

Configuring SSL for client-server instance communication

Databases can contain sensitive information; these days, the main concern is related to the security of data stored in tables as well as those sent over the network. One method of securing network communication between server and client is **SSL**, which is actually an abbreviation for **Secure Socket Layer**. We do not delve further into too much theory. Mainly, SSL addresses the following important security considerations: authentication, confidentiality, and integrity. Mainly SSL encryption and other network communication or also named data in transit encryption methods protects against unauthorized packet interception and analysis performed by an interposed person between a client and a server, also known as eavesdropping.

The DB2 instance has built-in support for SSL. DB2 relies on Global Security Kit for implementing SSL. GSKit is included in the IBM DB2 ESE software installation kit or is downloadable for free from IBM's website. Next, we'll show how to implement a secure connection between a DB2 server and a DB2 client.

Getting ready

For the next recipe, we will use `nodedb21` (`db2inst1` instance) as server and `nodedb22` (`db2clnt1` instance) as client, where we have installed DB2 Client in previous recipes. You need to ensure that you have GSKit libraries in `LD_LIBRARY_PATH`. In our case, the libraries that are located in `/home/db2inst1/sqllib/lib64` are pointing to the `/opt/ibm/db2/V9.7/lib64` location.

How to do it...

1. The first step is to add the `gsk8capicmd_64` executable in our `PATH`.

   Include the following in `.bash_profile`:

   ```
   PATH=$PATH:$HOME/bin:$HOME/sqllib/gskit/bin
   ```

   Execute `source .bash_profile` to reinitialize the user environment.

2. To create a key database on the server, execute the following (for more information about `gsk8capicmd_64`, execute `gsk8capicmd_64 –help`):

   ```
   [db2inst1@nodedb21 ~]$ gsk8capicmd_64 -keydb -create -db "/home/
   db2inst1/keystoredb2inst1.kdb" -pw "db2cookbook" -stash
   [db2inst1@nodedb21 ~]$
   ```

3. Create a self-signature and self-sign the key database on the server:

   ```
   [db2inst1@nodedb21 ~]$ gsk8capicmd_64 -cert -create -db "/
   home/db2inst1/keystoredb2inst1.kdb" -pw "db2cookbook" -label
   "db2cookbooksignature" -dn "CN=www.packtpub.com,O=Packt
   Publishing,OU=Packt Publishing"
   [db2inst1@nodedb21 ~]$
   ```

4. Extract the signature for signing the client key database:

   ```
   [db2inst1@nodedb21 ~]$ gsk8capicmd_64 -cert -extract -db "/home/
   db2inst1/keystoredb2inst1.kdb" -label "db2cookbooksignature"
   -target "/home/db2inst1/db2cookbook.arm" -format ascii -fips -pw
   "db2cookbook"
   [db2inst1@nodedb21 ~]$
   ```

5. Next, create the client key database:

   ```
   [db2inst1@nodedb21 ~]$ gsk8capicmd_64 -keydb -create -db "/home/
   db2inst1/keystoreclientdb2inst1.kdb" -pw "db2ckbk" –stash
   [db2inst1@nodedb21 ~]$
   ```

6. Import the self-signed certificate into the client key database:

```
[db2inst1@nodedb21 ~]$ gsk8capicmd_64 -cert -add -db "/home/
db2inst1/keystoreclientdb2inst.kdb" -pw "db2ckbk" -label
"db2cookbooksignature" -file "/home/db2inst1/db2cookbook.arm"
-format ascii –fips
[db2inst1@nodedb21 ~]$
```

7. To enable SSL as communication protocol on `nodedb21`, execute the following:

```
[db2inst1@nodedb21 ~]$ db2set DB2COMM=tcpip,ssl –i
[db2inst1@nodedb21 ~]$
```

8. Enable SSL as communication protocol also on the client side:

```
[db2clnt1@nodedb21 ~]$ db2set DB2COMM=tcpip,ssl –i
[db2clnt1@nodedb21 ~]$
```

9. Next, on `nodedb21`, set SSL-related parameters on the server instance; then, stop and start the instance:

```
[db2inst1@nodedb21 ~]$ db2 "update dbm cfg using ssl_svr_keydb /
home/db2inst/keystoredb2inst1.kdb"
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
completed
successfully.
[db2inst1@nodedb21 ~]$ db2 "update dbm cfg using ssl_svr_stash /
home/db2inst/keystoredb2inst1.sth"
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
completed
successfully.
[db2inst1@nodedb21 ~]$ db2 "update dbm cfg using ssl_svr_label
db2cookbooksignature"
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
completed
successfully.
[db2inst1@nodedb21 ~]$ db2 "update dbm cfg using ssl_svcename
50004"
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
completed
successfully.
[db2inst1@nodedb21 ~]$ db2stop
06/09/2011 19:08:39     0   0   SQL1064N  DB2STOP processing was
successful.
SQL1064N  DB2STOP processing was successful.
[db2inst1@nodedb21 ~]$ db2start
06/09/2011 19:08:45     0   0   SQL1063N  DB2START processing was
successful.
SQL1063N  DB2START processing was successful.
```

Description of SSL-related parameters used on the server side:

- `SSL_SVR_KEYDB` specifies a fully qualified filepath of the key file to be used for SSL setup at server side
- `SSL_SVR_STASH`—specifies a fully qualified filepath of the stash file to be used for SSL setup at server side
- `SSL_SVR_LABEL`—specifies a label of the personal certificate of the server in the key database
- `SSL_SVCENAME`—specifies the name of the port that a database server uses to await communications from remote client nodes using SSL protocol
- Be careful to set the correct paths, otherwise SSL won't work.

10. Copy `/home/db2inst1/keystoreinstclient.kdb` and `/home/db2clnt1/keystoreinstclient.sth` to `nodedb22`.

11. On `nodedb22`, set SSL DB2 client instance-related parameters:

```
[db2clnt1@nodedb22 ~]$ db2 "update dbm cfg using SSL_CLNT_KEYDB /
home/db2clnt1/keystoreclientdb2inst.kdb"

DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
  completed successfully.

[db2clnt1@nodedb22 ~]$  db2 "update dbm cfg using SSL_CLNT_STASH /
home/db2clnt1/keystoreclientdb2inst.sth"

DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command
  completed successfully.
```

Description of SSL-related parameters on the client side:

`SSL_CLNT_KEYDB` specifies the fully qualified filepath of the key file to be used for SSL connection at the client side

`SSL_CLNT_STASH` specifies the fully qualified filepath of the stash file to be used for SSL connections at the client side

12. Next, copy GSKit libraries to the client's `DB2HOME/lib64` directory:

```
[root@nodedb22 ~]# cp /opt/ibm/db2/V9.7_part/lib64/libgsk8* /opt/
ibm/db2/V9.7/lib64/

[root@nodedb22 ~]#
```

How it works...

SSL establishes the connection between client and server using a mechanism called **handshake**. There is a lot of information on the Internet about SSL and its working. Briefly, these are the steps for SSL handshake:

1. The client requests an SSL connection, listing its SSL version and supported cipher suites.
2. The server responds with a selected cipher suite.
3. The server sends its digital certificate to the client.
4. The client verifies the validity of the server's certificate (server authentication).
5. Client and server securely negotiate a session key.
6. Client and server securely exchange information using the key selected previously.

There's more...

In this recipe, we used a self signed certificate, which is fine for testing or internal use. For production environments, you should use trusted certificates signed by a third-party certification authority.

Other methods for encrypting data in transit can be implemented by using `DATA_ENCRYPT` and `DATA_ENCRYPT_CMP` as authentication methods. Also using port forwarding with SSH tunnels is a good option.

See also

Chapter 10, DB2 Security

Listing and attaching to instances

On a server environment, you may have many instances belonging to one DB2 installation or DB2HOME; obviously, you need to know about them and their name. For this purpose, you have the ability to use some specific commands to list them.

You also need to connect to these instances from remote locations to perform administration tasks; this, in the DB2 world, is called attaching.

Getting ready

In this recipe, we'll show how to list instances and attach to local and remote instances. Again, we'll use `nodedb21` as server and `nodedb22` as client.

How to do it...

Commands related to creating an instance are performed by the `root` user; listing is no exception and must be performed as `root`.

Listing instances

1. The command to list current instances is `db2ilist`. It lists the instances that belong to one DB2 copy. List instances created in DBCOPY1:

```
[root@nodedb21 ~]# /opt/ibm/db2/V9.7/instance/db2ilist
db2inst1
db2inst2
```

2. The same command from multipartitioned DB2HOME or DBCOPY2:

```
[root@nodedb21 ~]# /opt/ibm//db2/V9.7_part/instance/db2ilist
db2instp
```

Attaching to instances

1. On `nodedb22`, catalog `db2inst1` both as TCPIP and SSL, on our client instance `db2clnt1`, created before. Because we set up SSL as a separate communication method for the `db2inst1` instance, we have to specify it as the security method when cataloging the node (security SSL) with the SSL dedicated port. Catalog the nodes, as follows:

```
[db2clnt1@nodedb22 db2dump]$ db2 "CATALOG TCPIP NODE NODE21_S
REMOTE nodedb21 SERVER 50004 SECURITY SSL REMOTE_INSTANCE
db2inst1 SYSTEM  nodedb21 OSTYPE  LINUXX8664"
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the
  directory cache is refreshed.

[db2clnt1@nodedb22 db2dump]$ db2 "CATALOG TCPIP NODE NODE21_1
REMOTE nodedb21 SERVER 50001 REMOTE_INSTANCE  db2inst1 SYSTEM
nodedb21 OSTYPE  LINUXX8664"
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the
  directory cache is refreshed.
```

2. List the cataloged nodes:

```
[db2clnt1@nodedb22 ~]$ db2 "list node directory"

 Node Directory

 Number of entries in the directory = 2
```

```
Node 1 entry:
Node name                      = NODE21_S
 Comment                        =
 Directory entry type           = LOCAL
 Protocol                       = TCPIP
 Hostname                       = nodedb21
 Service name                   = 50004
 Security type                  = SSL
 Remote instance name           = db2inst1
 System                         = nodedb21
 Operating system type          = LINUXX8664

 Node 2 entry:

  Node name                      = NODE21_1
  Comment                        =
  Directory entry type           = LOCAL
  Protocol                       = TCPIP
  Hostname                       = nodedb21
  Service name                   = 50001
  Remote instance name           = db2inst1
  System                         = nodedb21
  Operating system type          = LINUXX8664
```

3. Attach to instance `db2inst1`, using first the SSL port, and next the TCP/IP port:

```
[db2clnt1@nodedb22 ~]$ db2 "attach to NODE21_S user db2inst1
using db2inst1"

   Instance Attachment Information

 Instance server        = DB2/LINUXX8664 9.7.4
 Authorization ID       = DB2INST1
 Local instance alias   = NODE21_S

[db2clnt1@nodedb22 ~]$ db2 " attach to node21_1 user db2inst1
using db2inst1"

   Instance Attachment Information
```

```
Instance server        = DB2/LINUXX8664 9.7.4
Authorization ID       = DB2INST1
Local instance alias   = NODE21_1
```

4. Attaching to an instance with the **Control Center**:

   In **Control Center** navigate to instance `db2inst1`, right-click, and choose **Attach**.

How it works...

Instances are registered in a file named global register. This file is always updated when an instance is created or dropped.

When you attach to an instance from a client, you can see that the port on the server is changing its status from *listening* to *established*:

```
[root@nodedb21 ~]# netstat -nlpta | grep 5000*
tcp        0      0 0.0.0.0:50001               0.0.0.0:*
LISTEN      19974/db2sysc 0
tcp        0      0 0.0.0.0:50003               0.0.0.0:*
LISTEN      26082/db2sysc 0
tcp        0      0 0.0.0.0:50004               0.0.0.0:*
LISTEN      19974/db2sysc 0
tcp        0      0 10.231.56.117:50001         10.231.56.118:49321
TIME_WAIT   -
tcp        0      0 10.231.56.117:50004         10.231.56.118:48187
ESTABLISHED 19974/db2sysc 0
```

This appears on `nodedb21`, after attaching to instance `db2inst1`, using the SSL port 50004.

There's more...

There is a straightforward method to verify that one instance is listening on its assigned port from a client. For this purpose, you can try to connect with telnet on that port:

```
[db2inst1@nodedb22 ~]$ telnet nodedb21 50004
Trying 10.231.56.117...
Connected to nodedb21.
Escape character is '^]'.
```

This means that our port assigned to SSL is listening. To detach from an instance, simply issue the `DETACH` command.

Another indirect method to list instances on a server is to use the discovery process provided by Configuration Assistant or Control Center locally or remotely.

See also

Chapter 11, *Using DB2 Discovery*

Dropping instances

There could be situations when it is necessary to drop an instance. An instance might be dropped by using the db2idrop command.

Getting ready

In this recipe, we will drop the instance `db2inst2`, created previously.

How to do it...

1. The command for dropping an instance is `db2idrop`. You have to be user `root` to drop an instance. First, we need to ensure that the instance is not active. If the instance has active connections and it is active, the `db2idrop` command fails.

2. Stop the instance by force:

```
[db2inst2@nodedb21 ~]$ db2stop force
07/12/2011 16:38:27     0   0   SQL1064N  DB2STOP processing was
successful.
SQL1064N  DB2STOP processing was successful.
[db2inst2@nodedb21 ~]$
```

If the instance hangs for some reason, the `db2_kill` command might be used. It will bring down the instance abruptly. However, be careful running this, because your databases running under this instance remain in an inconsistent mode.

3. As the user `root`, issue the following command to drop `db2inst2`:

```
[root@nodedb21 ~]# /opt/ibm/db2/V9.7/instance/db2idrop db2inst2
DBI1070I  Program db2idrop completed successfully.
```

How it works...

On Linux and Unix, `db2idrop` actually deletes the `sqllib` directory from the instance owner home. Therefore, it is recommended to save anything you have placed in this directory such as UDFs or external programs.

On Windows, `db2idrop` removes the service associated with the instance.

There's more...

As a best practice, before the instance is dropped, it is recommended to save the information related to that instance in a server profile file. In case you plan to recreate the instance and configure it as before, you can simply import the server profile after the instance is created again.

To export the instance profile, use **Control Center** | **Tools** | **Configuration assistant** | **Export profile** | **Customize**.

In the **Export** tab, you have plenty of options to export; choose anything you consider worth being saved.

2

Administration and Configuration of the DB2 Non-partitioned Database

In this chapter, we will cover:

- Creating and configuring DB2 non-partitioned databases
- Using configuration advisor
- Creating a database from an existing backup
- Configuring automatic database maintenance
- Managing federated databases, connecting to Oracle and MSSQL
- Altering databases
- Dropping databases

Introduction

This chapter provides recipes in order to create a database and get operational in simple and easy steps. We will do so in a manner that respects best practices in the industry.

You have created an instance named `db2inst1` on `nodedb21`, in the previous chapter. You will prepare for available disk space. You will then be able to configure your database for its mission and prepare it for automatic maintenance, so its operation is worry free.

While the nature of these recipes makes them useful right away, it is strongly recommended that they be attempted in a test environment first. We suggest you execute the commands individually, so you can learn as you go along.

Creating and configuring DB2 non-partitioned databases

We will discuss here how to create a single-partitioned database, which is sufficient for most database applications and is the most common configuration for small- to medium-sized databases.

If you plan on having a **Business Intelligence** (**BI**) database, you should be planning for a partitioned database. You can estimate one processor core for every 300 GB of data. We will cover this topic in *Chapter 3*, *DB2 Multi-partitioned Databases—Administration and Configuration*.

Getting ready

Gather as much technical information as you can about the hardware or virtual machine(s) you have at your disposal, for this database. Identify in which instance you will create your database, and ensure you will have enough memory and disk space for what you need.

Identify the location where you will create the table spaces (filesystems for Unix platforms, disk drives on Windows servers) and how much available space you will have. Make sure the instance owner has read and write permission in the directory that you will specify for your new database.

Best practices in the industry recommend separating data, indexes, lobs, and transaction logs on separate filesystems (or disk drives, on Windows systems). Depending on your installation, a filesystem can be defined as a single virtual disk on a NAS/SAN RAID 5 device, or a logical volume, spread on many physical disk drives. Check with your storage administrator for the best configuration—get as many disks/spindles as possible.

Decide on a data strategy—consider the database's mission and growth potential. Allow for possible partitioning, or table partitioning. MDC could also be a possible avenue. Decide on a naming convention for mount points and databases. The effort you spend on planning will save much time and money down the road.

Now perhaps you just want to get to the matter right away. We'll create a simple database; I'll explain the details as we go along.

How to do it...

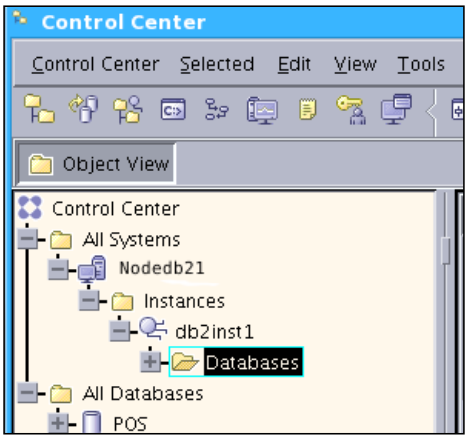
1. Log in as the target instance owner.

   Start **Control Center** and make sure the instance is started. Or, start the instance from the command line:

   ```
   [db2inst1@nodedb21 ~]$ db2start
   SQL1063N  DB2START processing was successful.
   ```

2. Choose the instance.

   Expand the **All Systems** node in the left pane of the **Control Center**. Choose the node and instance.



3. Create the database.

   Right-click on the **Databases** folder. A pop-up menu appears; select **Create Database** and start with the **Standard** option. We'll use default options for now.