Information Management software

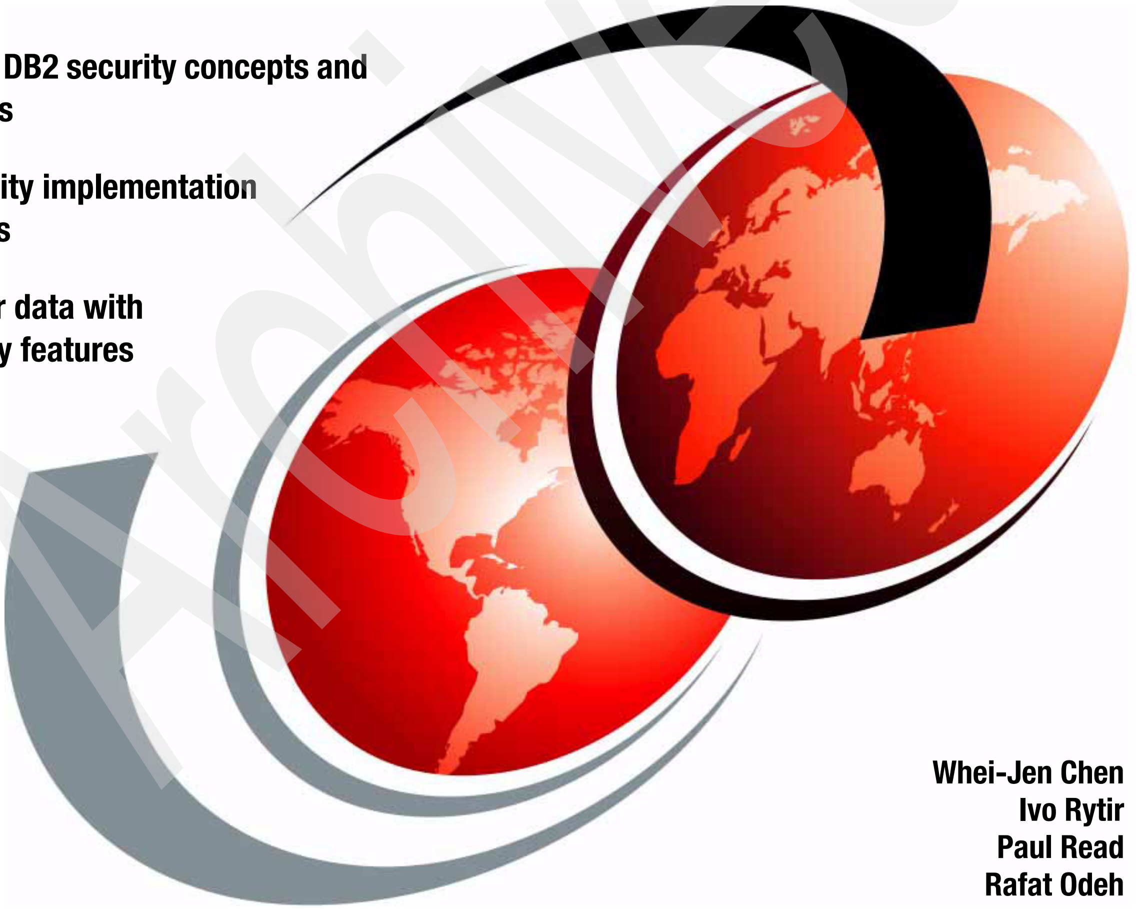
IBM

DB2 Security and Compliance Solutions for Linux, UNIX, and Windows

Understand DB2 security concepts and technologies

Learn security implementation by examples

Protect your data with DB2 security features



Whei-Jen Chen
Ivo Rytir
Paul Read
Rafat Odeh

Redbooks

ibm.com/redbooks

International Technical Support Organization

DB2 Security and Compliance Solutions for Linux, UNIX, and Windows

March 2008

SG24-7555-00

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (March 2008)

This edition applies to DB2 9.5 for Linux, UNIX, and Widows.

Contents

Notices

Trademarks

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

Redbooks (logo) ®	Domino®	IMS™
z/OS®	DB2 Connect™	Lotus®
zSeries®	DB2®	Redbooks®
AIX 5L™	DRDA®	System i™
AIX®	Informix®	System x™
BetaWorks™	IBM®	Tivoli®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Java, JDBC, JDK, JRE, Solaris, Sun, Sun Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Excel, Expression, Microsoft, SQL Server, Windows Vista, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

In this IBM® Redbooks® publication we discuss the existing and new DB2® security features introduced in DB2 9.5 for Linux®, UNIX®, and Windows®. These enriched DB2 security features provide you with the capability to protect your data and comply with regulatory requirements.

We describe how you can control data access through DB2 authentication and authorization functions. The role feature provides new options for tighter security, more granularity, and flexibility in administrating data access. Data encryption offers the capability to protect sensitive data in the database, critical database files, and data transferred over the network. Trusted contexts and trusted connections allow you to have more control over when a data access privilege becomes available to a user. Using label-based access control (LBAC), you can control read and write access of users to individual rows and columns at the table level. The enhanced audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events for analysis and identifying system misuse.

At the end, we introduce other DB2 data security solutions including IBM Database Encryption Expert, DB2 Audit Management Expert, and IBM Optim Enterprise Data Management.

The team that wrote this book

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

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT Specialist.

Ivo Rytir is an IBM Certified Solutions Expert with the DB2 Advanced Support Team at the IBM Toronto Software Lab. He has extensive application development experiences on Windows, Linux, and UNIX platforms with different programming languages. He holds a master's degree in Computer Science from the Brno University of Technology, Czech Republic.

Paul Read is a Relational Database Specialist with over 25 years of experience in BI and data management. Paul is the lead EMEA technical professional for Information Management products on the distributed platforms in the IBM BetaWorks™ team. He runs the beta and early support programs for data servers and associated products. He has also provided technical consultancy for the DB2 family and data management software products across all platforms.

Rafat Odeh has worked at IBM since 2000. He has been involved in many aspects of DB2 application development on LUW. Rafat is a member of the DB2 Tools and Connectivity Advanced Tech Support team. His area of expertise includes DB2 Authentication and DB2 DRDA®. Before joining IBM, Rafat worked as an advanced Informix® support analyst for nine years, where he became a Dialup Certified Analyst. During that time, he provided bug fixes, served as a subject matter expert, and taught various classes on IDS.

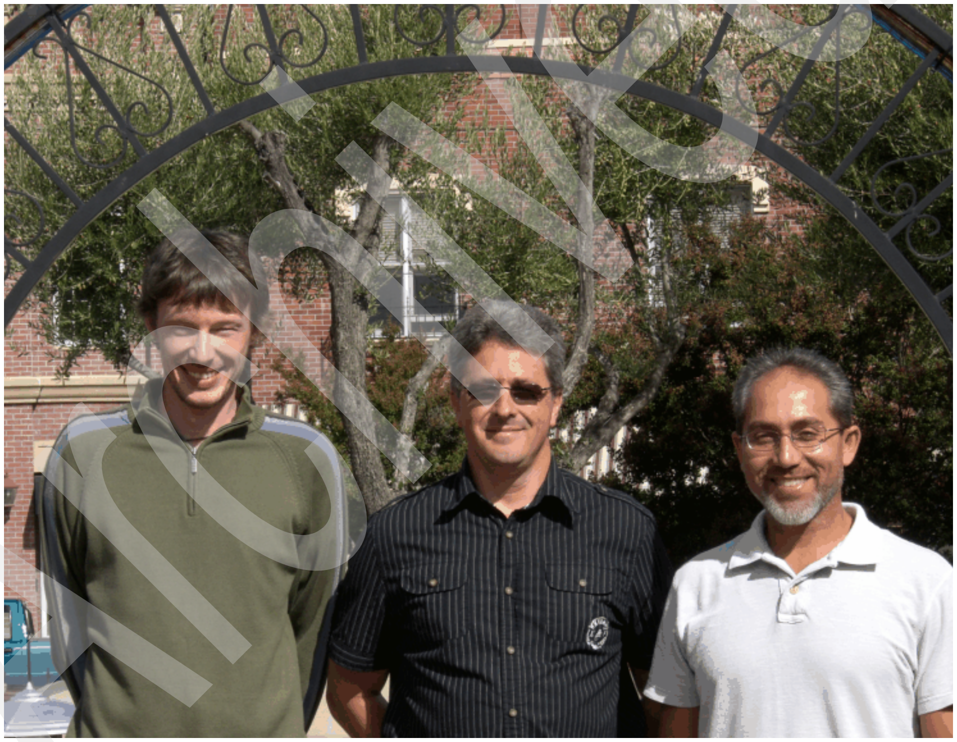


Figure 1 Left to right: Ivo, Paul, and Rafat

Acknowledgements

The authors would like to express their deep gratitude for the help received from **Walid Rjaibi**, who contributed advice, support, and written content.

We also thank the following people for their support and contributions to this project:

Paul Bird
Greg Stager
Ro Omrow
Kevin See
Raul F. Chong
IBM Software Group, Toronto Laboratory, Canada

Alice Ma
Sarbinder Kallar
Ted J. Wasserman
IBM Software Group, Silicon Valley Laboratory, USA

Marie Stephens
Saunnie Dunne
Bruce Fischer
IBM Software Group, Optim products

David Peters
IBM Software Group, Information Management

Rick Butler
Bank of Montreal (BMO) Financial Group, Toronto Canada

Emma Jacobs, Sangam Racherla, Deanna Polm
International Technical Support Organization

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

1

DB2 security overview

In this chapter we provide an overview of the security capabilities of DB2 for Linux, UNIX, and Windows, as well as a look into the new security enhancements in DB2 9.5 for Linux, UNIX, and Windows.

In this chapter we describe the following topics:

- DB2 security compliance
- DB2 security model
- DB2 administrator roles
- DB2 trusted context and connections
- DB2 data encryption
- DB2 audit enhancements
- DB2 security tools

1.1 DB2 security compliance

In today's business world, it is inevitable that a company has to hold both personal and business information. Data privacy and data security are becoming a major concern for not only the individual but also the business organizations. Data privacy refers to the evolving relationship between technology and the legal right to, or public expectation of, privacy in the collection and sharing of data. Data security is the means of ensuring that data is kept safe from corruption and that access to it is suitably controlled. Thus, data security helps to ensure privacy.

The ongoing demand to protect companies' and clients' data has increased the focus on information security and protection. Securing information assets and restricting data access to only those with the need to know are becoming a significant challenge for many organizations today. Clients always demand new and improved security strategies to be incorporated within their environment to provide robust internal control while maintaining system integrity and performance.

In recent years, the number of identity theft and security breaches have increased, causing businesses and organizations not only to loose a great deal of money, but also putting their reputations at stake. This prompted government agencies and financial institutions to push for more restrictions and legislation to govern the exchange and storage of data to protect client and personal information. As a result, key requirements relevant to database security emerged requiring organizations to enforce unique user ID and passwords usage, protect stored data, prevent data intrusion, implement more stringent auditing and monitoring of all access to data, regularly test security systems and processes, store data using security encrypted software, and limit access of sensitive and private data to those with the need to know.

Faced with these challenges, organizations spend a significant amount of resources to secure information assets and to ensure compliance with the new legislation and rules to avoid being penalized for non-compliance and prevent sanctions, lawsuits, possible imprisonment, and above all loss of client and investor confidence.

In this book, we describe the existing and new DB2 security features and tools that enable you to have security control over your DB2 data, your DB2 database system, and who can access them. DB2 9.5 for Linux, UNIX, and Windows has been enriched with security capabilities to help your organization comply with regulatory demands. These capabilities include authentication and authorization designed to allow only eligible users access to their applications and data, intrusion detection services to help detect and prevent unauthorized access to data, and encryption of data being archived or transmitted over the network. Figure 1-1 illustrates DB2 security capabilities.

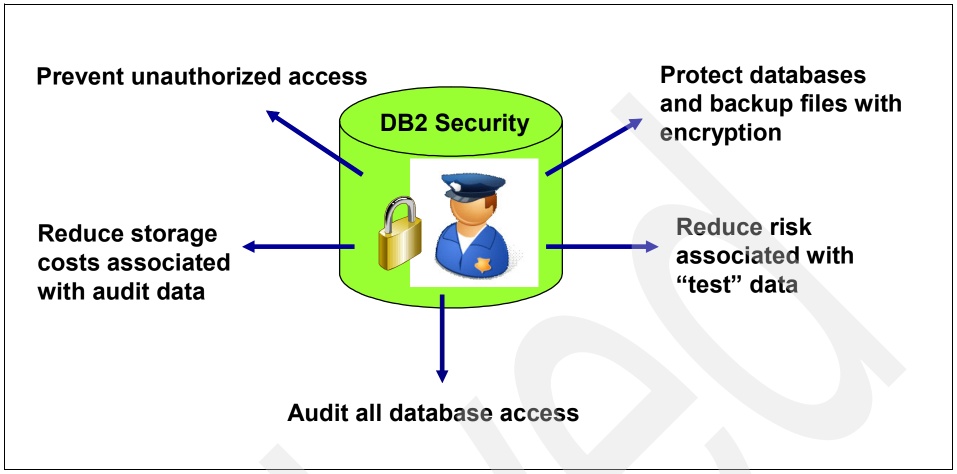

Prevent unauthorized access
DB2 Security
Protect databases and backup files with encryption
Reduce storage costs associated with audit data
Reduce risk associated with "test" data
Audit all database access

Figure 1-1   DB2 security capability

By the global nature of the Web, data privacy and data security compliance include both US and non-US security legislation. DB2 security features and functions help to enable you to ensure compliance with regulations. Examples of security legislation include:

- Payment Card Industry (PCI) Data Security Standard (DSS): This regulation is owned and distributed by the PCI Security Standards Council. It is a collaborative effort between major credit card companies and is designed to protect customers' personal information.
- California Senate Bill 1386: This is an amendment that provides consumers with notice of security breaches involving compromised personal information.
- Health Insurance Portability and Accountability Act of 1996 (HIPAA): This act focuses on health care and is designed to protect all forms of personal health information by defending patients' rights to have their health information kept private.
- Gramm-Leach-Bliley Act of 1999 (GLBA): This act defines that the financial institutions must comply with the privacy provisions that mandate controls over customers' non-political personal information (NPI) with respect to usage, protection, and distribution.
- Sarbanes-Oxley Act: This act redesigned federal regulation of public company corporate governance and reporting obligations by demanding accountability and assurance of financial reporting by executives, auditors, securities analysts, and legal counsel.

- Personal Information Protection and Electronic Documents Act (IPIPEDA): This act is a Canadian law that incorporates and makes mandatory provisions of the Canadian Standards Association's Model Code for the protection of personal information. PIPEDA contains various provisions to facilitate the use of electronic documents and governs how private-sector organizations collect, use, and disclose personal information in the course of commercial business.
- Data Protection Act (DPA): This is a United Kingdom Act of Parliament that defines a legal basis for the handling in the UK of information relating to living people. It is the main piece of legislation that governs protection of personal data in the UK. Compliance with the act is overseen by an independent government authority, the Office of the Information Commissioner (OIC).

1.2 DB2 security model

The DB2 security mechanism is divided into two main operations: authentication and authorization. The operations work together to provide secure access to DB2 resources.

The strategy for authentication within DB2 has been to rely on strong security mechanisms outside of the database, such as an operating system, to provide authentication facilities. Using these mechanisms allows you to leverage new developments in security and to reduce the administrative overhead of requiring users and groups to be defined to the database. DB2 relies on the configured security mechanism to provide both user and group membership information, through authentication security plug-ins. This approach also allows and supports businesses to create centralized repositories for user IDs and authentication in their enterprise, which minimizes costs for security maintenance and reduces integrity risks.

The strategy for authorization within DB2 is to maintain information about privileges and authorities within the database. Because database objects can be dependent on this information, maintaining privileges and authority information within the database provides tight access control by allowing us to react immediately to changes, to the point of invalidating objects that are impacted by such changes. New database entities are added into the existing hierarchy with appropriate privileges as required. Integrating with the configured authentication mechanism, privileges and authorities can be granted to either users or to group identities. Because group definitions are specified by an external mechanism, there are controls placed on their use to ensure that a change in membership externally is reflected in the objects within the database. DB2 offers a finer granularity of privileges together with configured database roles to allow for greater flexibility and ease of administration. Membership and acquisition of roles are controlled within the database, allowing an alternative to the existing group support. In addition, a role eliminates the group privilege restrictions when creating a static SQL object. This strategy provides robust internal control while maintaining high standards for system integrity and performance.

1.2.1 Authentication

Authentication is the first DB2 security feature that you encounter when you attempt to attach to a DB2 instance or connect to a DB2 database. Authentication is the process whereby the user identities himself to the DB2 database system and provides proof of his identity before access is granted or rejected. The actual user validation is performed by a security facility outside of the DB2 database system, through an authentication security plug-in. The security facility can be part of the operating system or a third-party security facility such as Kerberos or Lightweight Directory Access Protocol (LDAP). DB2 provides some built-in plug-ins, and you can build your own plug-in as well.

The security facility requires two items to authenticate a user: a user ID and an authentication token. The user ID identifies the user to the security facility. By supplying the correct authentication token (usually a password known only to the user and the security facility), the user's identity is verified. For greater flexibility, authentication security plug-in modules can be custom built to accommodate authentication needs and allow for customized levels of security.

After DB2 authenticates a user successfully, the user’s groups (if any) are acquired using a group membership security plug-in. *Groups* are defined and managed by an external entity such as an operating system or an LDAP directory. The default group acquisition process relies on an IBM-shipped operating system based group security plug-in.

Lastly, the authenticated user ID is mapped to a DB2 authorization ID. This mapping is determined by the authentication security plug-in. When using the default IBM shipped authentication security plug-in, there are two derived authorization IDs provided by the plug-in: system and session IDs. System authorization ID is used for checking the connect privilege to establish a connection, while the session authorization ID is the primary ID for the next connection. In the default plug-in, both authorization IDs are derived in the same way from the user ID and are identical. You can provide two different ones in the plug-in.

The authorization ID is simply the user ID in uppercase. This derived authorization ID is also referred to as the system authorization ID or the primary authorization ID, and is the one checked by DB2 when performing authorization checks.

Note: DB2 for Linux, UNIX, and Windows is configured to use the default authentication security plug-ins included when DB2 is installed. These security modules invoke the operating system based APIs to provide both user and group membership information.

1.2.2 Authorization

After a user is authenticated, the DB2 database manager determines whether that user is authorized to access data or resources. *Authorization* is the process whereby the DB2 database manager obtains information about the authenticated user, indicating which database operations that user can perform, and which data objects that user can access. Without the proper authorization, users cannot successfully execute SQL or XQuery statements.

When an authenticated user tries to access data, the authorization name of the user, the groups to which the user belongs, and the roles granted to the user directly or indirectly through a group or a role are compared with the recorded permissions. Based on this comparison, the DB2 server decides whether to allow or reject the requested access. DB2 tables and configuration files are used to record the permissions associated with each authorization name.

Figure 1-2 shows the overview of authentication and authorization.

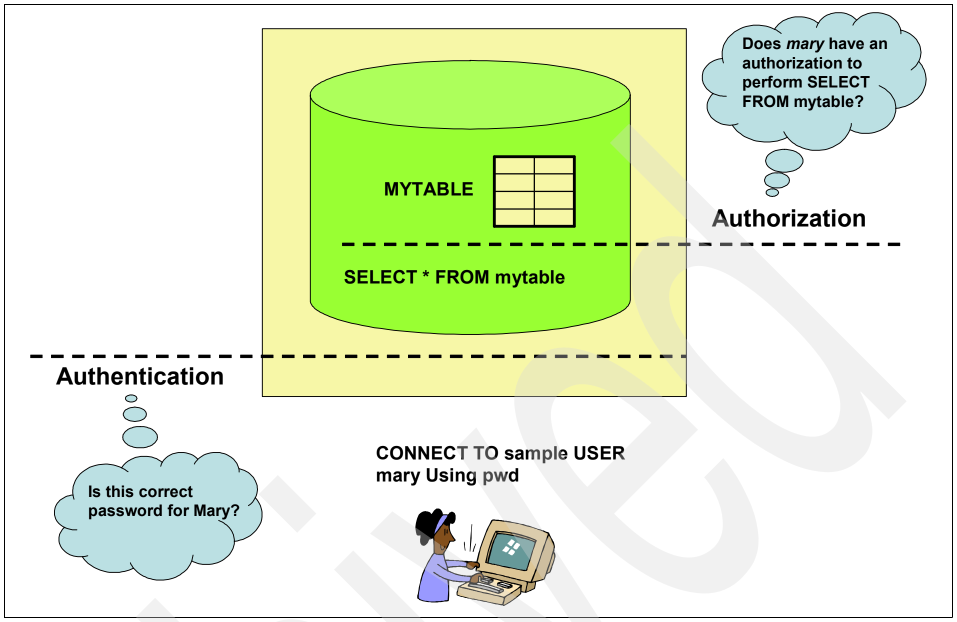

Does *mary* have an authorization to perform SELECT FROM mytable?
MYTABLE
Authorization
SELECT * FROM mytable
Authentication
CONNECT TO sample USER mary Using pwd
Is this correct password for Mary?

Figure 1-2 Authentication versus auhtorization.

There are different sources of permissions available to an authorization ID:

- Primary permissions: those granted to the authorization ID directly
- Secondary permissions: those granted to the groups and roles in which the authorization ID is a member
- Public permissions: those granted to PUBLIC, which is a special DB2 group representing everyone
- Context-sensitive permissions: those granted to a trusted context role

The database manager requires that each user be specifically authorized, either implicitly or explicitly, to use each database function needed to perform a specific task. *Explicit* authorities or privileges are granted to a user, a group, or a role. *Implicit* authorities or privileges are those acquired through groups to which the user belongs, or roles in which the user, the group, or another role is a member.

There are three forms of DB2 recorded authorization, administrative authority, privileges, and Label-Based Access Control (LBAC) credentials. A user, group, or role can have one or more of the these authorizations. LBAC credentials are introduced in 1.2.4, “LBAC” on page 16.

Administrative authority

Administrative authority gives a person the rights to be in charge of controlling the database and have the responsibility for data safety and integrity. DB2 administrative authority has a hierarchy, with SYSADM as the highest level. Under SYSADM, there are authorities for instance level and database level. These *authority levels* provide a method of grouping privileges and control over database manager or database maintenance and utility operations.

DB2 instance authorities

DB2 instance authorities operate on the instance level and have server-wide scope, that is, they apply to all databases within the instance. The instance authorities are associated with group membership, and the group names that are associated with the authority levels are stored in the database manager configuration file for a given instance.

There are four DB2 instance authorities:

- SYSADM (system administrator)

  The SYSADM authority level provides control over all the resources created and maintained by the database manager. The operating system administrator possesses all the authorities of DBADM, SYSCTRL, SYSMAINT, and SYSMON, and the authority to grant and revoke DBADM authority and SECADM authority. The user who has SYSADM authority is responsible both for controlling the database manager and for ensuring the safety and integrity of the data. SYSADM authority provides implicit DBADM authority within a database, but does not provide implicit SECADM authority within a database.

- SYSCTRL (system control)

  The SYSCTRL authority level provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority.

- SYSMAINT (system maintenance)

  The SYSMAINT authority level provides the authority required to perform maintenance operations on all databases associated with an instance. A user with SYSMAINT authority can update the database configuration, back up a database or tablespace, restore an existing database, and monitor a database. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority.

- SYSMON (system monitor)

  SYSMON provides the authority required to use the database system monitor. It operates at the instance level.

DB2 database authorities

Database authorities are linked to a specific database within the DB2 instance. Database-specific authorities are stored in the database catalogs. DB2 database authorities include:

- DBADM (database administrator)

  The DBADM authority level applies at the database level and provides administrative authority over a single database. This database administrator possesses the privileges required to create objects, issue database commands, and access table data. The database administrator can also grant and revoke CONTROL and individual privileges. DBADM, by default, has all other database authorities except SECADM.

- SECADM (security administrator)

  SECADM holds the security-related authorities that provide the ability to create, drop, grant, and revoke authorization or privileges, and transfer ownership of security objects such as roles and LBAC labels. The SECADM authority can only be granted by a user with SYSADM authority. It has no inherent privilege to access data stored in tables and has no other additional inherent privileges.

- CONNECT

  This authority allows the user to connect to the database.

- BINDADD

  This authority allows the user to create new packages in the database.

- CREATETAB

  This authority allows the user to create new tables in the database.

- CREATE_EXTERNAL_ROUTINE

  This authority allows the user to create a procedure for use by applications and other users of the database.

- CREATE_NOT_FENCED_ROUTINE

  This authority allows the user to create a user-defined function (UDF) or procedure that is not fenced. CREATE_EXTERNAL_ROUTINE is automatically granted to any user who is granted CREATE_NOT_FENCED_ROUTINE.

- IMPLICIT_SCHEMA

  This authority allows any user to create a schema implicitly by creating an object using a CREATE statement with a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

- LOAD

  This authority allows the user to load data into a table.

- QUIESCE_CONNECT

  This authority allows the user to access the database while it is in a quiescent state.

Privilege

Privileges are authorities assigned to users, groups, or roles, allowing them to perform specific activities on database objects such as creating a table or dropping an index. They strictly define the tasks users can perform on database objects. For example, a user can have the privilege to read data from a table but not update the existing data. Privileges can be granted to individual users, groups, roles, or PUBLIC. PUBLIC is a special group that consists of all users, including future users. Privileges are stored in the database catalogs.

Figure 1-3 illustrates the hierarchy of DB2 authorities and privileges. This ranges from privileges on tables to privileges on schemas and stored procedures. Note that DBADM and SECADM perform different duties and have completed different authorities and privileges.

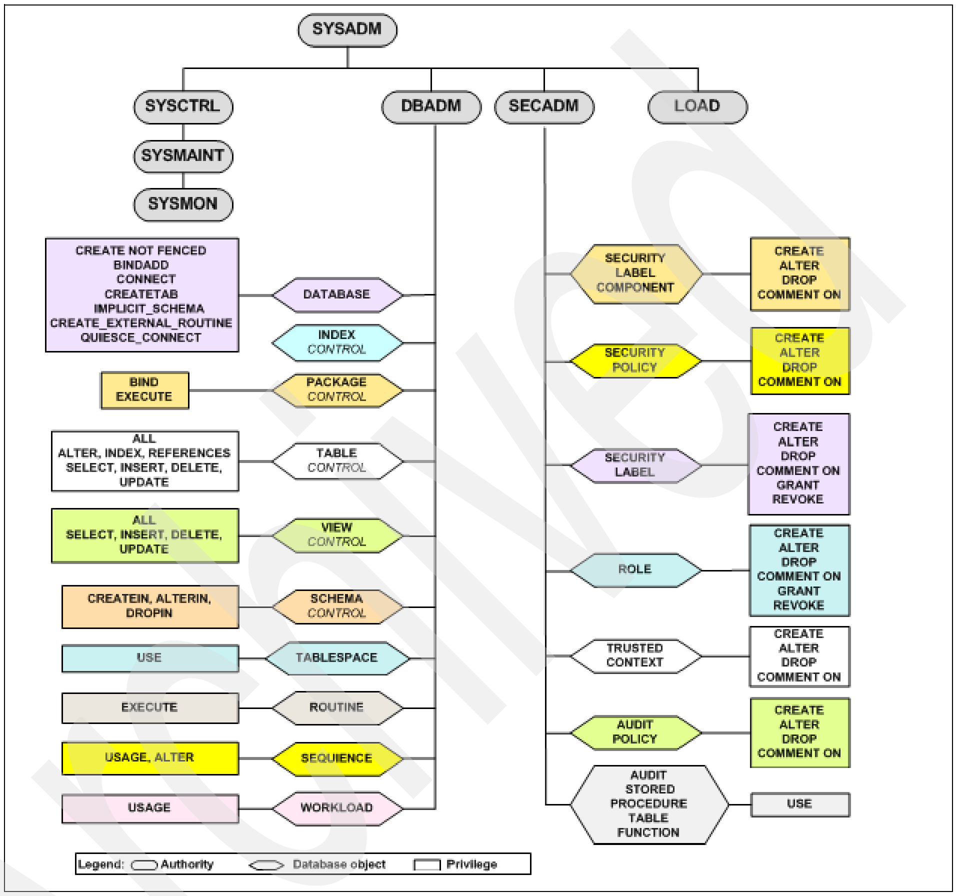

SYSADM
SYSCTRL
DBADM
SECADM
LOAD
SYSMAINT
SYSMON
CREATE NOT FENCED BINDADD CONNECT CREATETAB IMPLICIT_SCHEMA CREATE_EXTERNAL_ROUTINE QUIESCE_CONNECT
DATABASE
INDEX CONTROL
BIND EXECUTE
PACKAGE CONTROL
ALL ALTER, INDEX, REFERENCES SELECT, INSERT, DELETE, UPDATE
TABLE CONTROL
ALL SELECT, INSERT, DELETE, UPDATE
VIEW CONTROL
CREATEIN, ALTERIN, DROPIN
SCHEMA CONTROL
USE
TABLESPACE
EXECUTE
ROUTINE
USAGE, ALTER
SEQUENCE
USAGE
WORKLOAD
SECURITY LABEL COMPONENT
CREATE ALTER DROP COMMENT ON
SECURITY POLICY
CREATE ALTER DROP COMMENT ON
SECURITY LABEL
CREATE ALTER DROP COMMENT ON GRANT REVOKE
ROLE
CREATE ALTER DROP COMMENT ON GRANT REVOKE
TRUSTED CONTEXT
CREATE ALTER DROP COMMENT ON
AUDIT POLICY
CREATE ALTER DROP COMMENT ON
AUDIT STORED PROCEDURE TABLE FUNCTION
USE
Legend: Authority Database object Privilege

Figure 1-3 DB2 authorities and privileges

1.2.3 DB2 security plug-ins

The strategy for authenticating a user within a DB2 database system has been to rely on strong external security facilities such as the operating system, a Kerberos security system, or an LDAP to provide authentication mechanisms. DB2 interacts with these configured facilities to provide both user and group membership information. Beginning with DB2 8.2, authentication is done using security plug-ins. Security plug-ins are dynamically loadable libraries that DB2 invokes to perform user authentication and group membership lookups.

There are three categories of security plug-ins available within a DB2 server:

- A group plug-in performs group membership lookup for a given user on both the client and the database server.
- A client authentication plug-in to manage authentication on a DB2 client. It is also used to perform instance-level local authorization on the database server when instance-level commands such as **`db2start`**, **`db2stop`**, and **`update dbm cfg`** are executed.
- A server authentication plug-in to manage authentication on a DB2 server. This is used to check whether a user ID is known to the plug-in.

DB2 depends on the value specified for the AUTHENTICATION database manager configuration parameter to determine how and where authentication takes place. Currently, the DB2 database system supports two mechanisms for plug-in authentication:

- User ID and password authentication plug-in. This authentication method is used when any of the following authentication types is specified for the AUTHENTICATION parameter:
  - CLIENT
  - SERVER
  - SERVER_ENCRYPT
  - DATA_ENCRYPT
  - DATA_ENCRYPT_CMP
- Generic Security Service Application Program Interface (GSS-API) authentication plug-in Version 2 (IETF RFC2743) and Generic Security Service API Version 2: C-Bindings (IETF RFC2744). This authentication method is used when any of the following authentication types are used:
  - KERBEROS
  - GSSPLUGIN
  - KRB_SERVER_ENCRYPT
  - GSS_SERVER_ENNCRYPT

Each of the plug-ins can be used independently or in conjunction with one or more of the other plug-ins. For example, you might only use a server authentication plug-in and assume the DB2 defaults for client and group authentication. Alternatively, you might have only a group or client authentication plug-in. The only case where both a client and server plug-in are required is for GSS-API authentication plug-ins.

Figure 1-4 provides a general view of DB2 server security plug-in infrastructure.

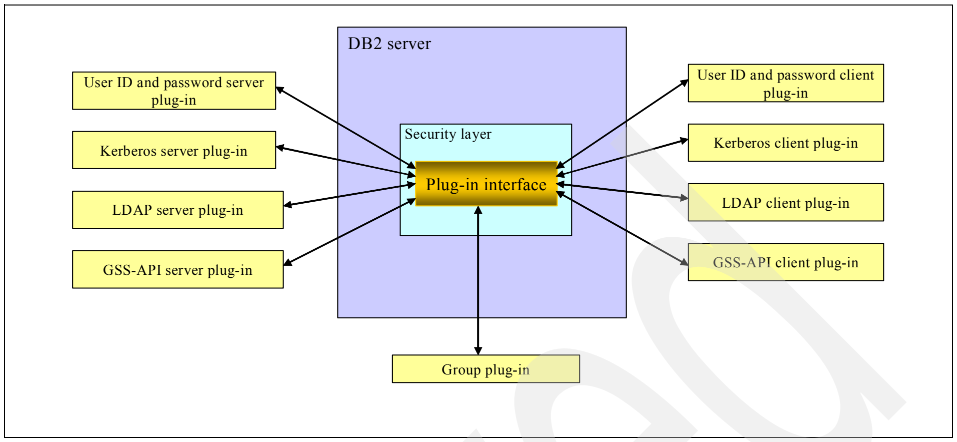


Figure 1-4 DB2 security plug-ins infrastructure

Starting with DB2 8.2, DB2 provides a set of default security plug-ins for group membership, user ID and password authentication, and Kerberos authentication. Under DB2 9.5, the list has been extended to include LDAP authentication plug-in modules, allowing you to authenticate users and groups stored in an LDAP directory, hence, removing the requirement to define users and groups on the operating system. DB2 LDAP security plug-ins are available for server-side authentication, client-side authentication, and group membership. This is supported on AIX®, Linux on IA32, Linux on x64, Linux on zSeries®, Solaris™, and Windows operating systems. The DB2 LDAP security plug-in modules support the following LDAP servers:

- IBM Tivoli® Directory Server (ITDS) Version 5.2, 6.0 and later
- Microsoft® Active Directory® (MSAD) Version 2000, 2003, and later
- Sun™ Java™ System Directory Server Enterprise Edition Version 5.2 and later
- Novell eDirectory Version 8.7 and later
- IBM Lotus® Domino® LDAP Server Version 7.0 and later.
- z/OS® Integrated Security Services LDAP Server Version V1R6 and later

DB2 default behavior is to use the user ID and password authentication plug-in modules, which implements the operating system authentication mechanism. Table 1-1 lists the IBM DB2 plug-ins.

Table 1-1 IBM DB2 authentication plug-in names

Plug-in name	Description
IBMOSauthclient	Client-side operating system based authentication plug-in
IBMLDAPauthclient	Client-side LDAP-based authentication plug-in
IBMOSauthserver	Server-side operating system based authentication plug-in
IBMLDAPauthserver	Server-side LDAP-based authentication plug-in
IBMOSgroups	Operating system-based group lookup plug-in
IBMLDAPgroups	LDAP-based group lookup plug-in
IBMkrb5	Keberos authentication plug-in (same file exists for both server and client side)

Table 1-2, Table 1-3, and Table 1-4 on page 15 show where the IBM DB2 security plug-in modules are located on various platforms.

Table 1-2 For 32-bit Linux and UNIX systems plug-in type location

Plug-in type	Location
Server	/sqllib/security32/plugin/IBM/server
Client	/sqllib/security32/plugin/IBM/client
Groups	/sqllib/security32/plugin/IBM/group

Table 1-3 For 64-bit Linux and UNIX systems plug-in type location

Plug-in type	Location
Server	/sqllib/security64/plugin/IBM/server
Client	/sqllib/security64/plugin/IBM/client
Groups	/sqllib/security64/plugin/IBM/group

Table 1-4 For Windows operating systems 32-bit and 64-bit plug-in type location

Plug-in type	Location
Server	%DB2PATH%\security\plugin\IBM\instance-name\server
Client	%DB2PATH%\security\plugin\IBM\instance-name\client
Groups	%DB2PATH%\security\plugin\IBM\instance-name\group

For the Windows operating system, IBM also provides default security plug-in modules to support two-part user IDs and the mapping of two-part user IDs to two-part authorization IDs. For example, a user ID composed of a domain and user ID such as HEYDRA\db2inst1, where HEYDRA is a domain name or a namespace, and db2inst1 is the user name. If the user account db2inst1 is defined on different locations such as local accounts and domain accounts, the HEYDRA\db2inst1 is a valid authorization ID for a plug-in that handles two-part authorization IDs. By default, both one-part user IDs and two-part user IDs map to one-part authorization IDs.

The plug-in files have the same name as the regular one, except they have the suffix *TwoPart* associated with them. These plug-in modules implement the same authentication method as the one-part plug-in files. Table 1-5 lists the two-part user ID plug-ins. These plug-ins are located under the same directories of the one-part ID plug-ins.

Table 1-5 IBM DB2 default security plug-in names of two-part user IDs

Authentication type	Two-part user ID plug-in name
Client	IBMOSauthclientTwoPart
Server	IBMOSauthserverTwoPart
Kerberos	IBMkrb5TwoPart

DB2 supports the LDAP authentication method through security plug-in. Earlier under DB2 8.2, users who wanted to implement LDAP authentication were asked to write their own LDAP security plug-ins or rely on a third-party vendors to develop customized security plug-ins for them. There were no LDAP default plug-in files provided by IBM or sample programs to show you how to write such modules. Shortly after DB2 8.2 was released, IBM provided LDAP security plug-ins upon request from DB2 technical support representatives. As demands for LDAP increase, DB2 9 provides support for LDAP authentication-based and group lookup through LDAP security plug-in modules. IBM default LDAP security plug-in modules, samples, and documentations are also provided.

If you decide not to use the default plug-ins, you can create your own loadable plug-ins or use a third-party loadable plug-in library to perform user authentication and group management mechanisms. To help you write your own security plug-ins, the sqllib/samples/security/plugins directory (on Lunix and UNIX) or %DB2PATH%\samples\security\plugin (on Windows platforms) contains sample source files for various types of plug-ins.

You can tailor and customize your security plug-ins to implement an alternative authentication method outside of the methods currently provided by DB2. After you write and test your plug-ins you need to install them. Installing requires you to place those loadable library files in the proper directory locations, then update specific database manager configuration parameters.

Security plug-ins present you with an alternative method to create your own customized security libraries to fit your business needs and organization, therefore providing you with more control over your environment.

For additional information about deploying customized security plug-in, refer to IBM White paper:

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0610see/index.html

1.2.4 LBAC

The DB2 LBAC solution is a flexible implementation of mandatory access control (MAC) at both the row level and the column level. Row-level and column-level protection can be used either separately or combined. LBAC complements the traditional DB2 discretionary access control (DAC). When a protected table is accessed, DB2 enforces two levels of access control: DAC, which is at the table level, to make sure the authorized ID holds the required privilege to perform the requested operation on the table; and LBAC at the row level, column level, or both. LBAC is configurable and can be tailored to match your particular security environment. All configuration is performed by a security administrator. This is a user that has been granted the SECADM authority by the system administrator (SYSADM) authority.

A security administrator configures the LBAC system by creating security policies. A security policy describes the criteria that will be used to decide who has access to what data. Only one security policy can be used to protect any one table, but different tables can be protected by different security policies.

After creating a security policy, the security administrator can create objects called security labels that are part of that policy. Exactly what makes up a security label is determined by the security policy and can be configured to represent the criteria that your organization uses to decide who should have access to particular data items. If you decide, for instance, that you want to look at a person’s position in the company and what projects they are part of to decide what data they should see, then you can configure your security labels so that each label can include that information.

LBAC is flexible enough to allow you to configure different security label criteria to a very simple system where each label represents different levels of trust. After a security label has been created, it can be associated with individual columns and rows in a table to protect the data held there. Data that is protected by a security label is called *protected data*. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user’s security label is compared to the security label protecting the data. DB2 then determines whether the user is allowed access to data at the column level, row level, or both, or denies access to the data.

A security administrator can also grant exemptions to users. An exemption allows you to access protected data that your security labels might otherwise prevent you from accessing. Together your security labels and exemptions are called your LBAC credentials. LBAC credentials are stored in the database catalogs.

One key advantage of using LBAC to protect sensitive data is that no authority, including SYSADM, DBADM, or SECADM, has any inherent privileges to access your data.

For more detailed information about LBAC, refer to Chapter 5, “Label-based access control” on page 83.

1.2.5 DB2 database roles

One of the of the new security features in DB2 9.5 that contributes to providing a secure and resilient environment for your data is database roles. A database *role* is a database object that groups together one or more privileges or database authorities. It can be assigned to users, groups, PUBLIC, or other roles using a GRANT statement. For example, you can define a developer role and allow this role to insert, update, and delete data on a set of tables.

A role allows you to control database access at a level of abstraction that is close to the structure of your organization. For example, you can create roles in the database that map directly to those in the organization. Users are granted membership in the roles based on their job responsibilities. As the user's job responsibilities change within the organization, user membership in roles can be easily granted and revoked.

Roles also simplify the assignment of privileges. Instead of granting the same set of privileges to each individual user in a particular job function, the security administrator can grant this set of privileges to a role representing that job function, and then grants that role to the users in that job function. Roles can be updated without updating the privileges for every user on an individual basis.

The primary difference between a role and a group is that a group is managed by software outside DB2 that prohibits DB2 from determining when membership in a group is changed. As a role is managed inside the database, DB2 can determine when an authorization changes and act accordingly. This provides another advantage of a role that the privileges and authorities granted to roles are always used when you create views, triggers, materialized query tables (MQTs), static SQL, and SQL routines, whereas privileges and authorities granted to groups are not used.

In essence, by granting access permissions to roles only and by allowing users to acquire a role or be granted a role, the administration and management of privileges in the database is greatly simplified.

For more detailed information refer to Chapter 3, “Roles” on page 47.

1.2.6 Trusted contexts and trusted connections

The DB2 9.5 Trusted Contexts feature has been designed specifically to address security concerns that comply with government regulations such as the Sarbanes-Oxely Act (SOX) and the Health Insurance Portability and Accountability Act (HIPAA).

A *trusted context* is a database object that describes a trust relationship between the database and an external entity such as a middle-tier application server. Additionally, trusted contexts introduce the concept of context-sensitive privileges. Context-sensitive privileges allow an organization to gain more control of when a privilege becomes available to a user. For example, an organization may want the human resources (HR) manager to acquire the ability to access the payroll table only when he is connected to the database from within the company offices, but not when he is connected from home.

Trusted contexts provide a way of building fast and more secure three-tier applications. It addresses many security concerns, which was lacking in the three-tier application model, such as loss of user’s identity, user accountability, and granting users unnecessary privileges to access certain information. Under DB2 trusted contexts, the user's identity can be sent to DB2 within a trusted connection using a switch command, and that identity can then be used for audit and authorization purposes.

When a user establishes a database connection, the DB2 database system checks whether the connection attributes match the definition of a trusted context object in the database. When a match occurs, the database connection is said to be *trusted*. The trust relationship is based upon the following attributes:

- System authorization ID: represents the user who establishes a database connection
- IP address (or domain name): represents the host from which a database connection is established
- Data stream encryption: represents the encryption setting (if any) for the data communication between the database server and the database client

The security administrator, SECADM, can create a trusted context object in the database that defines a trust relationship between the database and the middle tier. The middle tier can then establish an explicit trusted connection to the database, which gives the middle tier the ability to switch the current user ID on the connection to a different user ID, with or without authentication. In addition to solving the user identity assertion problem, trusted contexts offer another advantage: the ability to control when a privilege is made available to a database user. The lack of control over when privileges are available to a user can weaken overall security. For example, privileges might be used for purposes other than those originally intended. Now the security administrator can assign one or more privileges to a database role and assign that role to a trusted context object. Only trusted database connections (explicit or implicit) that match the definition of that trusted context can take advantage of the privileges associated with that role.

For more detailed information about trusted context and connections refer to Chapter 4, “Trusted contexts and connections” on page 67.

1.2.7 Data encryption

Encryption is a requirement for many of the regulatory compliances. For example, California Senate Bill No.1386 states that any agency, person, or business in California, USA, or that conducts business in the state and has computerized data that includes personal information is required to disclose any breach in security to every resident of California whose information has been compromised. Personal information that triggers the notification requirement of the bill includes an individual’s first name or first initial and last name combined with one or more of the following portions of data when either the name or the portion of data is not encrypted:

- Social security number
- Driver’s license number or california identification card number

- Account number, credit or debit card number, in combination with any required security code, access code, or password that would permit access to an individual’s financial account

For a business operated in California to comply with this bill, data encryption is required.

Most customers consider having a firewall and database security good enough protection, but what if a hacker gets past those barriers? What if the intrusion is from an authorized user? What if a mobile computer or backup tape is stolen? Encryption ensures that the data is unreadable.

Since Version 7.2, DB2 has provided built-in facilities for encrypting and decrypting user IDs and passwords to store and safeguard your data. With the ever-changing regulations to secure this data, under DB2 8.2, two new authentication types, DATA_ENCRYPT and DATA_ENCRYPT_CMP, were added to improve the security of user data transmitted over the network. To keep up with the new regulations and the demands for better and improved security facilities to protect sensitive data, DB2 9.5 supports the use of Secure Socket Layer (SSL). Using this communication protocol, client applications that use IBM Data Sever Drivers such as JDBC™ Type 4 Driver and SQLJ can connect to DB2 database servers using SSL sockets. This security mechanism meets the Federal Information Processing Standards (FIPS) developed by the United States Federal government.

For more detailed information refer to Chapter 7, “Data encryption” on page 151.

1.2.8 Auditing

The audit facility (db2audit) performance and administration have been extended for DB2 9.5 to include fine-grained configuration, new audit categories, separate instance and database logs, and new ways to customize the audit configuration. Because you now have control over exactly which database objects are audited, you no longer need to audit events that occur for database objects that you are not interested in. Consequently, the performance of auditing and its performance impact on other database operations has been greatly improved. Sole responsibility for managing audits at the database level now lies with the security administrator.

The following audit facility enhancements are included in DB2 9.5:

- You can use new database objects called audit policies to control audit configuration within a database.
- Individual databases can have their own audit configurations, as can particular objects within a database, such as tables, or even users, groups,

and roles. In addition to providing easier access to the information that you need, this enhancement also improves performance, because less data needs to be written to disk.

- Auditing SQL statements is easier and produces less output.

  The new audit category, EXECUTE, allows you to audit just the SQL statement that is being run. Previously, you needed to audit the CONTEXT event to capture this detail.

- Audit logs exist for each database.

  There is now one audit log for the instance and one audit log for each database. This feature simplifies audit reviews.

- The audit log now has a customized path.

  Control over the audit log path allows you to place audit logs on a large, high-speed disk, with the option of having separate disks for each node in a database partitioning (DPF) installation. This feature also allows you to archive the audit log offline, without having to extract data from it until necessary.

- You can archive audit logs.

  Archiving the audit log moves the current audit log to an archive directory, while the server begins writing to a new, active audit log. When you extract data from an audit log to a database table, it is from an archived log, not the active audit log. This prevents performance degradation caused by locking of the active audit logs.

  The security administrator is solely in control of configuring an audit for a database, and also has sufficient access to manipulate the audit log, issue the ARCHIVE command, and extract a log file into a table.

- You can audit new information in each category.

  The CURRENT CLIENT special registers allow values for a client user ID, accounting string, workstation name, and application name to be set within applications so that these values will be recorded in the audit data. The local and global transaction IDs can be recorded in audit data. This facilitates correlation between the audit log and the transaction log.

Note: The system administrator with SYSADM no longer has the right to manage the audit on the database level. You need SECADM for that. SYSADM authority is required for instance level auditing and executing the db2audit tool: start, stop, configure, extract, flush, and describe.

1.2.9 DB2 security tools

In addition to the security features provided under DB2 9.5, IBM provides additional tools to improve data security, data storage, and resilient auditing capabilities to help you protect your data. The tools are part of the IBM suite of regulatory compliance tools available for multiple platforms. These tools for DB2 on Linux, UNIX, and Windows simplify the management of complex and routine database tasks, help maximize DBA efficiency, and reduce total cost of ownership. Most importantly, they help your security structure achieve compliance with government regulations and privacy acts without affecting overall performance. With these tools you can provide audit reports that have information such as usage of sensitive data, monitor privileges granted to users to access sensitive data, and you can encrypt sensitive data without the need to change the applications. These tools are:

- IBM Database Encryption Expert

  The responsibility of protecting your company's sensitive data continues to grow, as well as the achievement to comply with different legislation and rules. In today's business world, protecting both production and non-production data is becoming a key requirement for organizations. This prompted many of these organizations to investigate methods of how to store data securely using different encryption tools and software.

  IBM Database Encryption Expert enables you to leverage the power of Storage Area Networks (SANs) safely while complying with privacy and security regulations in place or being enacted worldwide. The tool uses encryption algorithms approved by the U.S. National Institute of Science and Technology. The key advantage of this tool is in its encryption and decryption capability. It can be customized at row and column levels on the respective databases.

- DB2 Audit Management Expert

  Security rules and regulations, such as Sarbanes-Oxely, have placed a heavy burden on IT departments to fulfill the auditor's view of data. In today's business world, auditors are required to track, analyze, and report on the status of legal and regulatory compliance efforts. Until now, there has been no easy way for them to access the information they need, forcing them to request it from the IT department. Manually providing the data that auditors request takes valuable IT staff time, which could be devoted to more strategic activities. Now IBM provides a powerful tool, DB2 Audit Management, which is equipped with all the functions that auditors and database administrators need to get the information they require. The DB2 Audit Management Expert has the following advantages:

  – Provides centralized auditing tools to bring together information from many different sources into a correlated, coherent view.

  - Enables auditors to discover responsible persons and answer relevant questions about violations (that is, who, what, where, and when).
  - Empowers non technical users to define collection criteria and automatically launches traces to collect data.
  - Allows auditors to automatically generate their own reports and export the data into other applications such as Excel®.
- IBM Optim

  IBM Optim Test Data Management and Data Privacy Solutions provide comprehensive capabilities for improving testing efficiencies and de-identifying application data that can be used effectively across non-production environments. Optim's data masking technology preserves the integrity of the data and produces consistent and accurate results that reflect the application logic. Masked data can be propagated accurately across multiple non-production environments to generate valid test results

2

SYSADM, DBADM, SECADM, and OS authorities

In this chapter we focus on three main DB2 authority levels, SYSADM, DBADM, and SECADM, as well as the user rights required for DB2 processes on both Linux and UNIX platforms and Windows.

We discuss the following topics:

- Primary DB2 security authorities
- Operating systems authorities and DB2

2.1 Primary DB2 security authorities

DB2 database manager level authorities define the grouping of privileges and control over maintenance, execution of database manager utilities, and data access. These authorities are associated with group membership, and the group names that are associated with the authority levels are stored in the database manager configuration file for a specific instance. However, database level authorities are a set of privileges associated with a specific database and database objects within the DB2 instance. These authorities are stored in the database system catalog tables.

DB2 defines different hierarchal levels of authorities, SYSADM, SYSCTRL, SYSMAINT, SYSMON, SECADM, and DBADM, each with the ability to perform a subset of administrative operations such as creating of a database, creating database backups, and retrieving data. The first four authorization levels operate on the instance level and have system-wide scope authority. Each of these authorities is associated with a specific DB2 instance level parameter that controls which users receive that authority.

Both the DBADM and SECADM authorities are associated with a specific database managed by the DB2 database system. This chapter focuses on three main levels of authorities, SYSADM, DBADM, and SECADM. By outlining the different responsibilities associated with each authority, we show how each authority affects users accessing DB2 database system resources, DB2 database objects, and DB2 data.

2.1.1 SYSADM authority

System administrator (SYSADM) is the highest level of authority in DB2 that can be granted to a group. SYSADM authority is assigned to a group name designated by the SYSADM_GROUP database manager configuration parameter. Membership in that group is controlled by an external security facility installed on the operating system of the database server. The database manager interacts with this facility through a security plug-in.

Members of the group with SYSADM authority can perform the following tasks:

- Migrate databases.
- Modify the database manager configuration and database configuration files.
- Safeguard the data by performing database and log file backups.
- Perform restoration of databases and database objects such as table spaces.

- Grant and revoke other authorities and privileges to and from users, groups, or roles, for example, SYSCTRL, SYSMAINT, SYSMON, DBADM, and SECADM.
- Issue database manager and database commands such as db2start, db2stop, and db2imigr.
- Full control of instance and database objects such as buffer pools and table spaces.
- Run the db2audit utility and manage audit on the instance level.

The system administration authority is established by assigning a user group defined in the external security facility to the associated instance-level parameter SYSADM_GROUP. Every user in the group receives the SYSADM authority. The value of the parameter can be changed either from the command line or the Control Center. For example, to change the value of SYSADM_GROUP to a group db2admins, which all the DBAs are members of, execute the commands shown in Example 2-1 from the command line. For the change to take effect the DB2 instance must be restarted. Once set, all members in the group acquire SYSADM authority.

Example 2-1 Updating SYSADM_GROUP instance parameter from command line

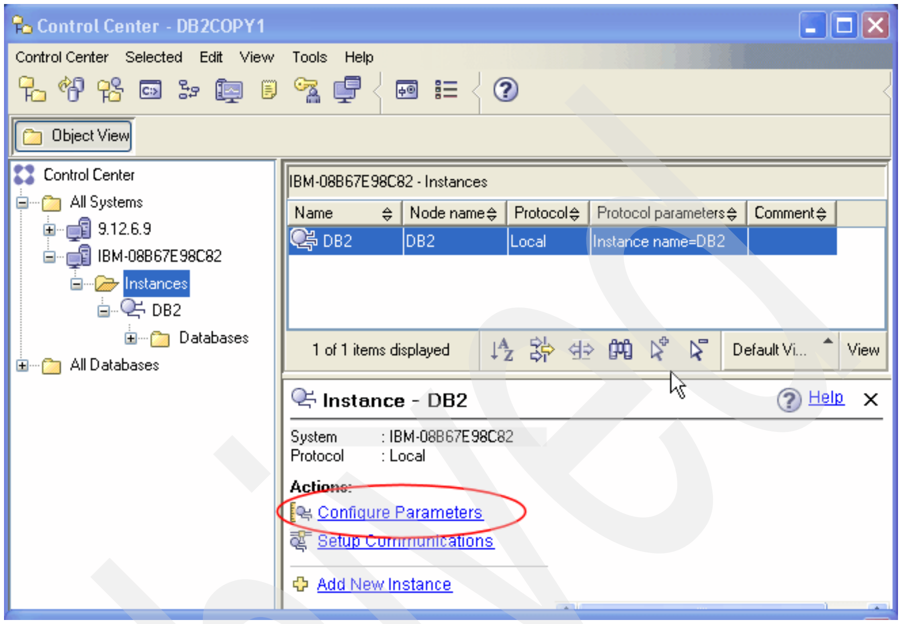
```
db2 => update dbm cfg using SYSADM_GROUP db2admins
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.

db2 => db2start
DB20000I  The DB2START command completed successfully.
```

To change the value of the instance-level authority parameter from the Control Center on a Windows platform, perform these steps:

1. Click **Start** → **All programs** → **IBM DB2, DB2COPY1 (Default)** → **Command Line Tools** → **General Administrator Tools** → **Control Center.**
2. In Control Center, expand the **All Systems** folder.
3. Expand the target system and **Instances** folder and select the instance that you want to change this value for, for example, DB2.

4. In the Instance notebook panel, click the action item **Configure Parameters**. See Figure 2-1.



IBM-08B67E98C82 - Instances

Name	Node name	Protocol	Protocol parameters	Comment
DB2	DB2	Local	Instance name=DB2	

Instance - DB2

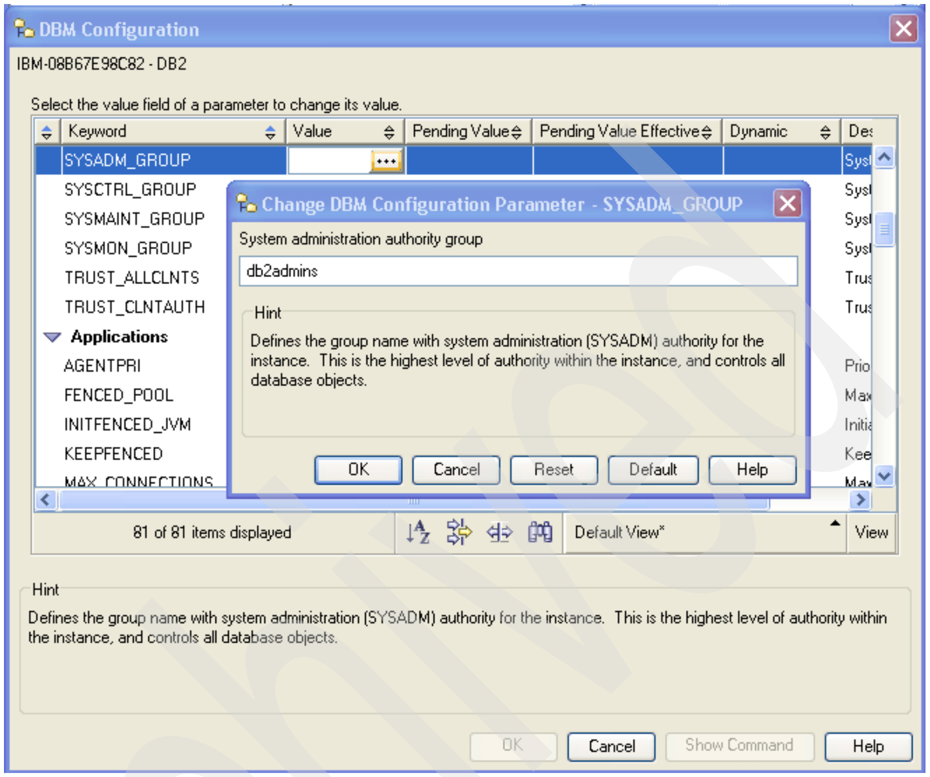
System : IBM-08B67E98C82
Protocol : Local

Actions:

Figure 2-1 Configuration Parameters action item

5. Scroll through the list of parameters until you find the SYSADM_GROUP instance level authority.

6. Click the button beside the parameter to change its value. See Figure 2-2.


DBM Configuration

IBM-08B67E98C82 - DB2

Select the value field of a parameter to change its value.

Keyword	Value	Pending Value	Pending Value Effective	Dynamic	Des
SYSADM_GROUP					Sys
SYSCTRL_GROUP					Sys
SYSMAINT_GROUP					Sys
SYSMON_GROUP					Sys
TRUST_ALLCLNTS					Trus
TRUST_CLNTAUTH					Trus
Applications					
AGENTPRI					Prio
FENCED_POOL					Max
INITFENCED_JVM					Initi
KEEPFENCED					Kee
MAX_CONNECTIONS					Max

81 of 81 items displayed

Default View*

Change DBM Configuration Parameter - SYSADM_GROUP

System administration authority group

db2admins

Hint

Defines the group name with system administration (SYSADM) authority for the instance. This is the highest level of authority within the instance, and controls all database objects.

Hint

Defines the group name with system administration (SYSADM) authority for the instance. This is the highest level of authority within the instance, and controls all database objects.

Figure 2-2 Updating SYSADM_GROUP parameter in Control Center

By default, the SYSADM_GROUP instance parameter is not set. On Linux and UNIX platforms, this means that the SYSADM authority defaults to the primary group name of the instance owner. For Windows, all members of the local Administrators group receive the SYSADM authority. For this reason, we highly recommend that you explicitly change the value of this parameter to a specific group name to prevent unauthorized access. Although users in a group specified by SYSADM_GROUP acquire SYSADM instance-level authority, they do not hold SECADM authority.

Granting SYSADM authority

Users with SYSADM authority must use the GRANT statement to grant other database authorities on a specific database to users, groups, or database roles. Figure 2-3 describes the syntax for the GRANT statement.

```
           .-,----------------------------.
           V                              |
>>-GRANT----+-BINDADD-------------------+-+--ON DATABASE-------->
            +-CONNECT-------------------+
            +-CREATETAB-----------------+
            +-CREATE_EXTERNAL_ROUTINE---+
            +-CREATE_NOT_FENCED_ROUTINE-+
            +-IMPLICIT_SCHEMA-----------+
            +-DBADM---------------------+
            +-LOAD----------------------+
            +-QUIESCE_CONNECT-----------+
            '-SECADM--------------------'

        .-,---------------------------------.
        V                                   |
>--TO----+-+-------+--authorization-name-+-+-------------------><
         | +-USER--+                     |
         | +-GROUP-+                     |
         | '-ROLE--'                     |
         '-PUBLIC------------------------'
```

Figure 2-3 Syntax for GRANT statement

While an authorized user ID with SYSADM instance-level authority can grant other database authorities, including SECADM, to individual users on the system, he cannot grant himself this authority. The command fails with an SQL0554N error, as shown in Example 2-2.

Example 2-2 SQL error message

```
db2 => GRANT secadm ON DATABASE TO USER db2admin
DB21034E  The command was processed as an SQL statement because it was
not avalid Command Line Processor command.  During SQL processing it
returned: SQL0554N  An authorization ID cannot grant a privilege or
authority to itself.SQLSTATE=42502
```

Viewing SYSADM authorities

Prior to DB2 9.5, to view the SYSADM authority information for a specific authorized ID, you would use the GET AUTHORIZATIONS command. The command reports the authorities of the current user from values found in the database manager configuration file and the catalog view SYSCAT.DBAUTH.

The command displays direct and Indirect authorities granted to the current user. Direct authorities are those explicitly granted to the authorized ID, while indirect author ties are authorities implicitly granted through a group membership. Example 2-3 shows the different administrative authorities for the db2admin user ID when executing the GET AUTHORIZATIONS command.

Example 2-3 Administrative authorities for db2admin

```
db2 => GET AUTHORIZATIONS

 Administrative Authorizations for Current User

 Direct SYSADM authority                    = NO
 Direct SYSCTRL authority                   = NO
 Direct SYSMAINT authority                  = NO
 Direct DBADM authority                     = YES
 Direct CREATETAB authority                 = YES
 Direct BINDADD authority                   = YES
 Direct CONNECT authority                   = YES
 Direct CREATE_NOT_FENC authority           = YES
 Direct IMPLICIT_SCHEMA authority           = YES
 Direct LOAD authority                      = YES
 Direct QUIESCE_CONNECT authority           = YES
 Direct CREATE_EXTERNAL_ROUTINE authority   = YES
 Direct SYSMON authority                    = NO

 Indirect SYSADM authority                  = YES
 Indirect SYSCTRL authority                 = NO
 Indirect SYSMAINT authority                = NO
 Indirect DBADM authority                   = NO
 Indirect CREATETAB authority               = NO
 Indirect BINDADD authority                 = NO
 Indirect CONNECT authority                 = YES
 Indirect CREATE_NOT_FENC authority         = NO
 Indirect IMPLICIT_SCHEMA authority         = NO
 Indirect LOAD authority                    = NO
 Indirect QUIESCE_CONNECT authority         = NO
 Indirect CREATE_EXTERNAL_ROUTINE authority = NO
 Indirect SYSMON authority                  = NO
```

Starting with DB2 9.5, the GET AUTHORIZATIONS command has been deprecated. Instead, a new table function SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID has been provided to retrieve this information. The table function returns all authorities held by an authorization ID, either found in the database manager configuration file or granted to an authorization ID directly or indirectly through a group or a role.

The syntax of SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID TABLE is as follows:

```
>>-AUTH_LIST_AUTHORITIES_FOR_AUTHID--(--authid--,--authidtype--)-><
```

This function accepts two input arguments:

- *authid* - This parameter has the VARCHAR(128) type and specifies the authorization ID being queried. The authorization ID can be a user, a group, or a role. If the *authid* is an empty string, or set to NULL, an empty result table is returned.
- *authidtype* - This parameter has the VARCHAR(1) type and specifies the authorization ID type being queried. The possible values are G for group, U for user, and R for role. If the authid type is an empty string, NULL, or does not exist, an empty result table is returned.

Example 2-4 demonstrates how to use the AUTH_LIST_AUTHORITIES_FOR_AUTHID table function in a query to display the different administrative and database authorities, including SYSADM authority, for user db2admin.

Example 2-4 Viewing administrative and database authorities

```
db2 => SELECT char(authority, 26) authority, d_user, d_group, d_public,
role_user, role_group, role_public, d_role FROM TABLE
(AUTH_LIST_AUTHORITIES_FOR_AUTHID ('DB2ADMIN', 'U')) as T

AUTHORITY                  D_USER D_GROUP D_PUBLIC ROLE_USER ROLE_GROUP ROLE_PUBLIC D_ROLE
-------------------------- ------ ------- -------- --------- ---------- ----------- ------
SYSADM                     *      Y       *        *         *          *           *
DBADM                      N      N       N        N         N          N           *
CREATETAB                  Y      N       N        N         N          N           *
BINDADD                    Y      N       N        N         N          N           *
CONNECT                    Y      N       Y        N         N          N           *
CREATE_NOT_FENCED_ROUTINE  Y      N       N        N         N          N           *
SYSCTRL                    *      N       *        *         *          *           *
SYSMAINT                   *      N       *        *         *          *           *
IMPLICIT_SCHEMA            Y      N       N        N         N          N           *
LOAD                       Y      N       N        N         N          N           *
CREATE_EXTERNAL_ROUTINE    Y      N       N        N         N          N           *
QUIESCE_CONNECT            Y      N       N        N         N          N           *
SECADM                     N      N       N        N         N          N           *
SYSMON                     *      N       *        *         *          *           *

  14 record(s) selected.
```

Revoking SYSADM authority

To revoke SYSADM authority from a group, another user with SYSADM authority level needs to modify the SYSADM_GROUP database manager configuration parameter to a different group name. Another option is to remove the user ID from the group defined in the external security facility. To accomplish this, the user must have super-user privileges such as root access on Linux and UNIX or administrator privileges on a Windows platform.

Limiting data access

If for any reason that you want to restrict the person with SYSADM authority to access data in a table, you can use DB2 LBAC feature. LBAC allows you to protect data in row or column levels using security labels. Only a user with security administration authority (SECADM) can grant the access of security labels to others. Without the privilege of the security labels, SYSADM cannot access the protected data. For more details on LBAC refer to Chapter 5, “Label-based access control” on page 83.

2.1.2 DBADM authority

Database administration authority (DBADM) is the second highest level of authority provided by DB2. It provides user, group, and role administration authorities over a single database. It allows the user to perform certain operations on that database such as:

- Create, activate, and drop event monitors.
- Grant and revoke database authorities and other individual privileges such as INSERT, SELECT, UPDATE, and DELETE on database objects.
- Grant and revoke CONTROL privileges on referenced tables, views, packages, and other database objects.
- Create a log file.
- Query the state of a tablespace.
- Query system catalog tables.
- Update log history files.
- Quiesce a tablespace.
- Reorganize database tables.
- Collect catalog statistics using the RUNSTATS utility.

Granting DBADM authorities

Only a user with SYSADM authority can grant and revoke DBADM authority to users, groups, or database roles. Example 2-5 shows the syntax to grant this authority to those entities.

Example 2-5 Granting DBADM authority

```
GRANT DBADM ON DATABASE TO USER db2inst1

GRANT DBADM ON DATABASE TO GROUP db2admins

GRANT DBADM ON DATABAE TO role db2admrole
```

When a database is created, DBADM authority as well as certain database authorities and privileges are implicitly granted to the database creator. These same authorities and privileges are implicitly granted by the database manager when the DBADM authority level is explicitly granted to a user. Table 2-1 describes the authorities and privileges a user will receive when granted this authority.

Table 2-1 Authorities and privileges granted to a user with DBADM authority

Authorities and privilege	Description
BINDADDAUTH	Allows users to create packages in the database
CONNECTAUTH	Allows users to connect to the database
CREATETABAUTH	Allows users to create tables in the database
DBADMAUTH	Allows users to grant and revoke privileges within the database
EXTERNALROUTINEAUTH	Allows users to create a procedure for use by applications and other users of the database
IMPLSCHEMAAUTH	Allows users to create schemas in the database
LOADAUTH	Allows users to load data into tables
NOTFENCAUTH	Allows a user to create a user-defined function (UDF) or procedure
QUIESCECONNECTAUTH	Allows users to access the database while it is in quiesce state
BIND and EXECUTE privileges	Allows users to BIND and EXECUTE packages created by NULLID schema

Authorities and privilege	Description
EXECUTE WITH GRANT privileges	Allows users to invoke stored procedures and functions created with SYSPROC schema and grant permissions to other users
SELECT and UPDATE privileges	Allows the user to view information in system catalog views created by SYSIBM, SYSCAT, and SYSIBMADM schemes
UPDATE privilege	Allows the user to view and update statistical information created by SYSSTAT schema
CONTROL privilege	Allows users to create and access objects, and to grant and revoke privileges to or from other users on those objects
INSERT, SELECT, UPDATE DELETE privileges	Allows users to load, view, update, and delete information from the database

Users holding DBADM authority can grant these authorities and privileges on the database to other users and can revoke any privilege from any user regardless of who granted them. While users with DBADM authority hold some of the same abilities as other authorities, they do not hold any of the abilities of the SECADM authority. The abilities provided by the SECADM authority are not provided by any other authority.

Viewing DBADM authorities

Different from the DB2 instance level authorities, the DB2 database authorities are stored in the database catalog tables. Example 2-6 provides a sample query used to retrieve all authorization names that have been granted DBADM authority and a sample output showing a group, a user, and a role holding this authority.

Example 2-6 Retrieving DBADM authorities from the database

```
db2 => SELECT DISTINCT char(grantee, 18) grantee, char(grantee, 18)
grantee, char(granteetype, 18) granteetype FROM syscat.dbauth
WHERE dbadmauth = ‘Y’

GRANTOR            GRANTEE            GRANTEETYPE
------------------ ------------------ ------------------
DB2ADMIN           DB2ADM             G
DB2ADMIN           DB2ADMROLE         R
DB2ADMIN           DB2INST1           U
```

The possible values displayed under the GRANTEETYPE column are:

- G = group
- R = role
- U = user

You also can use the AUTH_LIST_AUTHORITIES_FOR_AUTHID table function to obtain all of authorized user names holding the DBADM authority on the database. Example 2-7 provides a sample query and output showing that DBADM authority was granted directly to user ID db2inst1.

Example 2-7 Viewing DBADM authorities using AUTH_LIST_AUTHORITIES_FOR_AUTHID table function

```
db2 => SELECT char(authority, 26) authority, d_user, d_group, d_public
FROM TABLE (AUTH_LIST_AUTHORITIES_FOR_AUTHID ('DB2INST1', 'U')) AS t
WHERE authority ='DBADM'

AUTHORITY                    D_USER D_GROUP D_PUBLIC
------------------------------ ------ ------- --------
DBADM                        Y      N       N

  1 record(s) selected.
```

Revoking DBADM authorities

To revoke the DBADM authority from a user, a group, or a role, a user with SYSADM authority can use the REVOKE statement. Example 2-8 provides examples on how to revoke DBADM authority from an authorized ID, a group, and a database role.

Example 2-8 Syntax for revoking DBADM database authority f

```
REVOKE DBADM ON DATABASE FROM USER db2inst1

REVOKE DBADM ON DATABASE FROM GROUP db2admins

REVOKE DBADM ON DATABASE FROM ROLE db2admrole
```

Figure 2-4 shows the syntax for the REVOKE database authorities statement.

```
           .-,-----------------------------.
           V                               |
>>-REVOKE----+-BINDADD-------------------+-+--ON DATABASE------->
             +-CONNECT-------------------+
             +-CREATETAB-----------------+
             +-CREATE_EXTERNAL_ROUTINE---+
             +-CREATE_NOT_FENCED_ROUTINE-+
             +-IMPLICIT_SCHEMA-----------+
             +-DBADM---------------------+
             +-LOAD----------------------+
             +-QUIESCE_CONNECT-----------+
             '-SECADM--------------------'

         .-,---------------------------------.
         V                                   |  .-BY ALL-.
>--FROM----+-+-------+--authorization-name-+-+--+--------+-----><
           | +-USER--+                     |
           | +-GROUP-+                     |
           | '-ROLE--'                     |
           '-PUBLIC------------------------'
```

Figure 2-4 Syntax for REVOKE database authorities statement.

When a user with SYSADM authority creates a database, he is automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSADM group and you want to prevent that user from accessing that database as a DBADM, you must explicitly revoke the user's DBADM authority.

Revoking DBADM authority requires SYSADM or DBADM authority. It is important to understand that revoking DBADM authority does not automatically revoke any privileges that were held by the authorization name on objects in the database, nor does it revoke any of the other database authorities that were implicitly granted when DBADM authority was originally granted. For example, suppose you initially granted DBADM authority to the `db2inst1` authorized ID, and at a later time you decided to revoke that authority. After executing the revoke statement, the authorized ID would no longer hold the DBADM database authority. However, the user would still retain the implicit database authorities (BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, LOAD, and QUIESCE_CONNECT) granted to him when db2inst1 was originally granted the DBADM authority. You need to explicitly revoke these authorities from db2inst1.

Considerations

DB2 uses a special group called PUBLIC, which certain database authorities and privileges can be granted to and revoked from any user who successfully authenticates with the DB2 instance. However, the PUBLIC group cannot be granted the DBADM authority since the group name is not defined by the external security facility.

Table 2-2 shows the database authorities implicitly granted to PUBLIC. We recommend that you revoke these authorities from PUBLIC to prevent unauthorized access. You also can create a database with a restrictive option. DB2 does not implicitly grant the listed database authorities to PUBLIC if the restrictive option is specified.

Table 2-2 List of database authorities granted to PUBLIC after database creation

Database authorities and privileges	**Description**
CONNECT	Allows the user to connect to the database
CREATETAB	Allows the user to create tables in the database
BINDADD	Allows the user to create and bind packages to the database
IMPLICIT_SCHEMA	Allows the user to create objects with a schema that does not exist
SELECT	Allows access to all system catalog tables and views
UPDATE	Updates on SYSSTAT tables
EXECUTE with GRANT	EXECUTE with GRANT privilege on all procedures in schema and all table functions and procedures in SYSROC schema
SQLJ EXECUTE with GRANT	EXECUTE with GRANT privilege to on all functions and procedures in SYSFUN schema
EXECUTE	EXECUTE privilege on all table functions and other procedures in schema SYSIBM
BIND and EXECUTE	On all packages owned by NULLID schema
CREATEIN and CREATE IN	Privilege to create schemas on SQLJ and NULLID
USE	Allows user to create tables in any user space

2.1.3 SECADM authority

Beginning with DB2 9.1, DB2 has been extended to include several security capabilities such as label-based access control (LBAC), support for LDAP authentication security plug-ins, Secure Sockets Layer (SSL) protocol support, and a new security administrator authority level (SECADM). You can benefit from these features to provide greater control over sensitive data and improved reporting capabilities. In this section, we focus on the security administrator authority (SECADM) and the benefits provided by this authority.

The SECADM authority applies at the database level and is the only authority that can perform specific database security related actions. The SECADM authority has no inherent privilege to access data stored in tables or any other additional inherent privileges. SECADM authority can only be granted and revoked by a user with SYSADM authority to individual users. With SECADM authority, the user can perform the following actions in the database:

- Create, alter, comment on, and drop:
  - Audit policies
  - Security label components
  - Security policies
  - Trusted contexts
- Create, comment on, and drop:
  - Roles
  - Security labels
- Grant and revoke:
  - Roles
  - Exemptions
  - Security labels
  - SETSESSIONUSER privileges
- Use of the audit system stored procedures and table function: SYSPROC.AUDIT_ARCHIVE, SYSPROC.AUDIT_LIST_LOGS, and SYSPROC.AUDIT_DELIM_EXTRACT. These can only be invoked by the security administrator.
- Use the AUDIT statement to associate an audit policy with a particular database or database object at the server.
- Transfer ownership of objects that they do not own using the TRANSFER OWNERSHIP statement.

In order to create the security objects, the user receiving SECADM authority must have CONNECT authority on the database. By default, DB2 considers the authorized user ID receiving this authority as a valid member of the PUBLIC group, therefore explicitly granting him the database authorities and privileges described in Table 2-2 on page 38. In this case, the user ID receiving the SECADM authority is able to create his own database objects such as tables, views, and indexes. For this reason, before you grant SECADM authority to a user, revoke all the database authorities and privileges from PUBLIC, except the CONNECT authority, to prevent unauthorized access to the database.

Granting SECADM authority

Using the GRANT statement described in Figure 2-3 on page 30, the SYSADM can explicitly designate an individual user to receive the SECADM authority. Example 2-9 shows how to grant SECADM authority to user db2sec.

Example 2-9 Granting SECADM authority to a user

```
db2 => GRANT SECADM ON DATABASE TO USER db2sec
DB20000I  The SQL command completed successfully.
```

Viewing SECADM authority

Similar to DBADM database authority, the SECADM authority level is stored in the database catalog and can be viewed through SYSCAT.DBAUTH. To view the user authorization names that hold this authority level, use the sample SQL statement illustrated in Example 2-10.

Example 2-10 Retrieving SECADM authority from the database

```
db2 => SELECT DISTINCT char(grantee, 18) grantee, char (grantor, 18)
grantor, char(granteetype, 18) granteetype FROM syscat.dbauth WHERE
securityadmauth = ‘Y’

GRANTOR            GRANTEE            GRANTEETYPE
------------------ ------------------ ------------------
DB2ADMIN           DB2SEC             U

  1 record(s) selected.
```

Another method to retrieve this information is to use the new table function AUTH_LIST_AUTHORITIES_FOR_AUTHID, as shown in Example 2-11.

Example 2-11 Viewing SECADM authority using AUTH_LIST_AUTHORITIES_FOR _AUTHID table function

```
db2 => SELECT char(authority, 30) authority, d_user, d_group, d_public
FROM TABLE (AUTH_LIST_AUTHORITIES_FOR_AUTHID ('db2sec', 'U')) AS T
WHERE authority = 'SECADM'

AUTHORITY                      D_USER D_GROUP D_PUBLIC
------------------------------ ------ ------- --------
SECADM                         Y      N       N

  1 record(s) selected.
```

Revoking SECADM authorities

Only a user with SYSADM authority can revoke SECADM authority from a user. For example, suppose have granted SECADM to user db2sec, and later you decided to revoke this authority from the same user, using the REVOKE database authority statement described in Figure 2-4 on page 37, execute the command as shown in Example 2-12

Example 2-12 Revoking SECADM authority

```
db2 => REVOKE secadm ON DATABASE FROM db2sec
DB20000I  The SQL command completed successfully.
```

After revoking SECADM from a user, any security objects such as roles or security policies created by the security administrator remain in the database and cannot be dropped even with SYSADM authority.

2.2 Operating system authorities and DB2

DB2 uses the security facility of an operating system or a product in the database server to authenticate users. Before DB2 can successfully authenticate a user ID to communicate with the database system, DB2 must successfully interact with the operating system to access the external security facility configured on the machine. DB2 communicates with the security facility installed on the machine by calling certain APIs. To invoke these APIs, a user account with the proper user privileges is required on the operating system by DB2. The user privileges are established during the DB2 installation process.

For Linux and UNIX operating systems, you can use either root or non-root authority to perform the installation. The DB2 install program assigns proper permissions, ownership, and group to all DB2 files and libraries installed to allow proper access by the DB2 instance created on the system and run without interruptions. These files and libraries are copied to the instance owner home directory with the correct permissions, ownership, and group name during DB2 instance creation. Altering the permissions, ownership, or group causes unpredictable results, and in some cases failure to access shared memory resources or failure to read the database configuration file. Some of these problems are resolved by executing the **db2iupdt** command.

For Windows-based operating systems, user privileges are determined by the user security settings configured on Windows. These user privileges are influenced by the user account performing the installation. The DB2 install can be performed using any of the following:

- A local user account to perform the installation

  The user must belong to the local administrators group or be logged in as the local administrator on the machine on which DB2 is installed.

- A domain user account

  The user account must belong to the domain administrator group on the domain where DB2 instance accounts will be created.

- An alternative non-administrator user account

  This user can perform DB2 installation if the Windows elevated privileges settings are configured to allow a non-administrator user account to perform an installation.

- A local system user account

  To use the local system user account to perform the installation, the user right “Access this computer from the network” has to be assigned to the user account.

DB2 users on Windows platforms must posses certain operating system privileges to perform different tasks on the database system such as creating a database, performing database backups and restore, and creating table spaces. These privileges are inherited implicitly through group membership. For the Windows operating system, during the DB2 installation process an extended Windows security feature is implicitly enabled by default. As a result, the DB2 installer creates two new default group names, DB2ADMNS and DB2USERS. Optionally, you can create your own group names. If the group names you choose already exist on your system, the privileges for these groups will be altered. Both groups, DB2ADMNS and DB2USERS, are granted specific privileges. These privileges are not associated with DB2 authority level such as SYSADM, SYSMAINT, and SYSCTRL, but they are related to the operating system level authorities to allow members in these groups access to system resources such as the file system and directories, services, shared memory resources, and registry keys.

One of the key advantages of using the DB2ADMNS group is that you can assign one or all of the DB2 authority levels to it instead of using the local administrators group. For instance, if you want to assign SYSADM authority to a group, you should set SYSADM_GROUP instance parameter to the DB2ADMNS group name. Members of the DB2ADMNS group have access and control of all DB2 objects including shared memory and semaphore resource, services, registry keys, and file system. Both groups must be of the same type, either local groups or domain groups.

The following Windows user privileges are required by DB2 for DB2ADMNS or any group name that you designate as the DB2 administrator group during installation:

- Create a token object - required for certain token manipulation operations and used in authentication and authorization.
- Replace a process level token - allows user to create a process as another user.
- Increase quotas - allows the user to create a process as another user.
- Act as part of the operating system - allows the user to execute the LogonUser API for authentication purposes.
- Generate security audits - allows the user to manipulate the audit and security log.
- Take ownership of files or other objects - allows users to modify object Access Control Lists (ACLs).
- Increase scheduling priority - modifies the process working set.
- Back up files and directories - profile/registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, and RegLoadKey(Ex).
- Restore files and directories - profile/registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, and RegLoadKey(Ex)).
- Debug programs - token manipulation (required for certain token manipulation operations and used in authentication and authorization)
- Manage auditing and security log - generates auditing log entries.
- Log on as a service to run DB2 as a service.

- Access this computer from the network - allows network credentials (allows the DB2 database manager to use the LOGON32_LOGON_NETWORK option to authenticate, which has performance implications).
- Impersonate a client after authentication - required for Windows to allow use of certain APIs to impersonate DB2 clients: Impersonate LoggedOnUser, ImpersonateSelf, RevertToSelf, and so on.
- Lock pages in memory - allows large page support.
- Create global objects - terminal server support, which is required on Windows.

On the other hand, the Windows user privileges for the DB2USERS group are limited compared to the DB2AMNS. The Windows user rights and privileges required by DB2 for DB2USERS or any group name you designate as the DB2 users group during installation are:

- Access this computer from the network - allows network credentials (allows the DB2 database manager to use the LOGON32_LOGON_NETWORK option to authenticate, which has performance implications).
- Create global objects - terminal server support, which is required on Windows.

If you choose to disable the extended security feature during the installation, the two groups DB2AMNS and DB2USERS are not created, and members of the local administrators group are granted the following user rights:

- Act as part of the operating system.
- Debug programs.
- Create token object.
- Increase quotas.
- Lock pages in memory.
- Log on as a service.
- Replace a process level token.

To view the security settings for local user accounts on your Windows system, click **Start** → **Control Panel** → **Administrative Tools** → **Local Security Policy** → **User Rights Assignment**.

If you later decide to enable the extended security feature, use the **db2extsec** command. The command updates the permissions on DB2 objects such as files and directories, network shares, shared memory and semaphore resource, registry keys, and services on updated DB2 database system installations. To execute the command, you must have SYSADM authority, and it does not require a database connection.

Figure 2-5 shows the syntax for the **db2extsec** command.

```
>>-db2extsec--+----------------------+-------------------------->
              '-+-/u-----+--usergroup-'
                '-/users-'

>--+-------------------------+--+------------------------+------->
   '-+-/a------+--admingroup-'  '-/oldusers--oldusergrp-'
     '-/admins-'

>--+------------------------+--+------------------+------------->
   '-/oldadmins--oldadmngrp-'  '-/file--inputfile-'

>--+----------+--+------------+--+-----------+------------------><
   '-/verbose-'  '-+-/r-----+-'  '-+-/h----+-'
                   '-/reset-'      +-/help-+
                                   '-?-----'
```

Figure 2-5 Syntax for db2extsec command

To enable extended security to use group names other than the default group names, for instance, a local DB2 user's group name mydb2users and DB2 administrator group name mydb2admns, execute the command as shown in Example 2-13.

Example 2-13 Enabling extended security with local user groups

```
db2extsec /u mydb2users /a mydb2admns
```

Example 2-14 shows how to enable extended security with domain group names mydomain\mydb2users and mydomain\mydb2admns.

Example 2-14 Enabling extended security with domain groups

```
db2extsec /u mydomain\mydb2users /a mydomain\mydb2admns
```

Example 2-15 shows how to disable or undo the changes previously set by the **db2extsec** command. Use the /r or /reset. If this option is specified, other options are ignored.

Example 2-15 Resting

```
C:\SQLLIB\BIN>db2extsec /r
DB2EXTSEC: processing completed successfully.
```

When the extended security is enabled on your system, a global DB2 instance registry variable, DB2_EXTSECURITY, is set to a value of YES. The registry variable accepts two values either, YES and NO or ON and OFF. Setting the registry variable to NO does not enforce unauthorized access to DB2 objects or on the DB2 file system. To check whether the registry variable is set, run the `db2set -all` command. Setting and unsetting this variable requires you to restart the DB2 instance to the new value to take effect.

Without acquiring the proper user privileges or rights, DB2 processes and functionality fail, causing applications and tasks to generate fail with unexpected error messages. It is extremely important to coordinate and communicate with your UNIX or Windows system administrator to discuss the proper user privileges and rights required by the DB2 instance owner account in order for your database and business application to work without interruptions and minimize down time.

3

Roles

DB2 9.5 provides new options for tighter security and allows for more granularity and flexibility in administrating the data access. One such option is the database entity called a *role.*

Database roles can be thought of as groups that are defined and managed by DB2. A database role is essentially a database object that may group together one or more privileges or database authorities, and may be granted to users, groups, PUBLIC, trusted context, or other roles. Roles simplify the administration and management of privileges by offering capabilities similar to groups but without the same restrictions.

In this chapter we discuss the concept of roles using examples. We demonstrate how to take advantage of this new DB2 feature in combination with other essential e-business technologies such as Web services, Web application server, and DB2 database server.

3.1 Definition of a role

A role is a database object to which one or more DB2 privileges, authorities, or other roles can be granted or revoked. A role does not have an owner, and it can only be created or dropped by the security administrator (SECADM).

By associating a role with a user, the user inherits all the privileges held by the role, in addition to privileges already held by the user.

The key advantage of database roles is that they simplify the administration and management of privileges in a database. For instance:

- Security administrators can control access to their databases at a level of abstraction that is close to the structure of their organizations. For example, if the company has 12 branches and everyone within each branch as a set of identical privileges then the SECADM would set 12 roles and then grant membership to users based on their location.
- Users are granted membership in the roles based on their job responsibilities. As the user's job responsibilities change, which may be frequent in a large organization, user membership in roles can be easily granted and revoked. For example, if a user moves from the New York branch to the Boston branch then the SECADM simply revokes his access to the role for New York and grants access to the role for the Boston branch.
- The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role representing that job function and then grants that role to the users in that job function. For example, individual jobs can often require many different privileges for a user. However, if the privileges are granted to a role then it is simple to grant or revoke the privileges without having to maintain large scripts for each job. if the SECADM needs to alter the privileges for a role, he alters it in one place without having to replicate to process for all users.
- Roles can be updated without updating the privileges for every user on an individual basis. For example, if the SECADM needs to alter the privileges for the branch in New York, he can alter the definition for the role without having to replicate the process for all users.

All DB2 privileges and authorities that can be granted within a database, with the exception of SECADM, can be granted to a role. By granting privileges and authorities to roles only and making users members in roles, the administration and management of privileges in the database is greatly simplified.

3.2 Scenario - basic setup

In this chapter we introduce some sample code to describe the behavior of roles. The samples are based on database objects that are added to the SAMPLE database. You can download the setup scripts and data from IBM Redbooks Web site. Refer to Appendix B, “Additional material” on page 263, for download instructions.

The examples are based on a simple company called LUW Enterprises, Inc. Figure 3-1 shows the company’s organization structure. It is a small startup company and the structure lists the departments’ members. Note that Adam and Mary work in two different departments, as indicated by the arrows, and Yang also performs the role of pensions administrator.

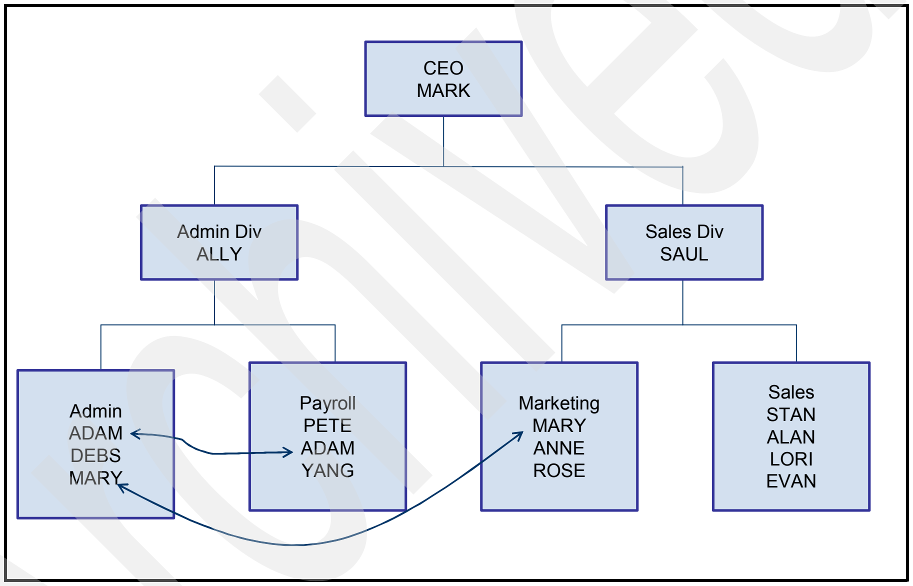


Figure 3-1 LUW Enterprises company structure

Follow these steps to set up the database:

1. Create users.

   Create the following user IDs in your system:

   DB2INST1 DB2SEC

MARK	ALLY	ADAM	DEBS
PETE	YANG	SAUL	MARY
ANNE	ROSE	STAN	ALAN
LORI	EVAN		

2. Create directory.

   Unzip the file Roles.zip. The setup scripts should be run in the given sequence to ensure smooth running.

   Before running any scripts you need to substitute the word *password* with the relevant password that you created with the user.

3. Run AddUsers.sql to grant connect access:

   ```
   db2 -tvf AddUsers.sql
   ```

   This script grants connect access to users added in step 1 on page 49, as well as adding the database administrator (DBADM) to DB2INST1 and the security administrator (SECADM) to DB2SEC.

4. Run LUWEBASE.sql to create a base table for the examples:

   ```
   db2 - tvf LUWEBASE.sql
   ```

5. Run LUWELOAD.sql to import data into the base tables:

   ```
   db2 -tvf LUWELOAD.sql
   ```

6. Run LUWEPLUS.sql to create views and other database objects:

   ```
   db2 -tvf LUWEPLUS.sql
   ```

Now you should have the required setup to be able to follow the examples in this book.

3.3 Planning

The planning and design considerations for setting up roles for new and existing databases should be approached with the same care as the design of the application.

For new databases the design of the uses of roles should be integral to the design of the database and the applications that use the database. When planning the use of roles, you are modeling the security access to a business system and the rules or relationships within that system. In many cases, if the database is well designed and applications are well defined, then the definition of roles should follow. However, you must consider the use of roles, together with all aspects of security and design.

For existing databases, it is possible for groups and roles to complement each other, or you can migrate from a *group* structure or a complex series of individual privileges and authorities to a structure to take the full advantage of the roles. First you need to establish the current design for the security access. If this is based on a group structure then the migration should be a smooth transition of replicating the group structure with roles. Then you would grant the roles to users, as you had the groups. However, if you have a complex design with individual users with different privileges, then you need to design a role structure from new.

For the examples in this book we chose a simple role design that duplicates the company structure. In order to demonstrate some of the strengths of this feature we then included some anomalies.

Designing a role structure is a much more complex task than is indicated here, and there are many items that need to be considered, such as trusted contexts, security and authorization, and so forth. You can find some of this information in the DB2 Information Center and in the many DB2 manuals that are available.

3.4 Setup and configuration of roles

The authority to manage membership in a role is vested in SECADM. The SECADM uses standard DML commands to manage the roles.

All DML commands for use with roles can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared only if DYNAMICRULES run behavior is in effect for the package.

The CREATE ROLE statement defines a role at the current server. The role must have a unique identifying name.

The DROP ROLE statement deletes a role at the current server. The role must already exist at the current server and the role cannot be deleted if currently in use as a connection attribute (SESSION_USER) or as a trusted context.

The GRANT and REVOKE ROLE statements are used to grant and revoke roles to users, groups, or to other roles.

The COMMENT ON statement can be used to provide a description of the ROLE in the system catalogs.

All the system catalog views that have details about users or groups have been updated to include the new roles. There are two new views in DB2 9.5 that show the details of roles:

- SYSCAT.ROLES

  This view has one row for each role defined and contains identifiers for the role plus the creation date and associated audit policy information.

- SYSCAT.ROLEAUTH

  This view has one row for each user, group, or role that has been granted authority over a role and whether they have grant authority on the role.

To simplify the administration of roles, there is a system function called AUTH_LIST_ROLES_FOR_AUTHID provided to enable the administrator to see which roles have been granted to a user, group, or role. This function will be described and demonstrated in “Query access to a role” on page 62.

3.4.1 Create roles

Once you have the roles planned, you can create roles in the database using the CREATE ROLE statement:

```
CREATE ROLE role-name
```

As you can see, the CREATE ROLE statement is very simple and it merely creates a database object that can be use to define authorities over other database objects like tables, views, triggers, and trusted contexts. Users can be assigned to a role to enable them to access data within the database.

In our example, we create a hierarchy of roles that identify the structure of the LUW Enterprises company, as illustrated in Figure 3-2. We create the basic roles of:

- Sales: for the sales department
- Mktg: for the marketing department
- Admin: for the administration department
- Payroll: for the payroll department
- Pension: for Yang who is the pensions administrator
- SalesMgr: To include the roles sales and Mktg
- AdminMgr: to include the roles admin and payroll
- CEO: to include the roles AdminMgr and SalesMgr

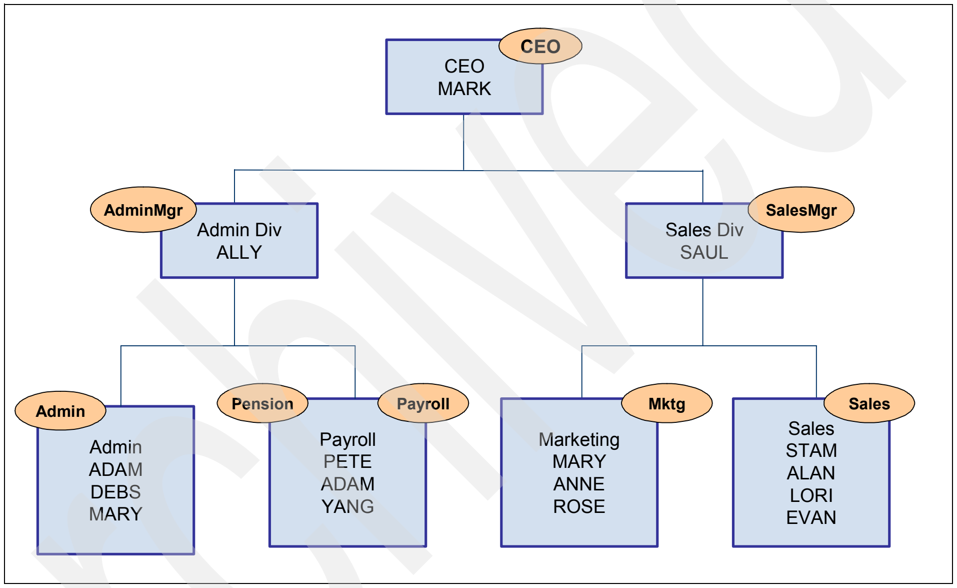


Figure 3-2 Roles structure

Example 3-1 shows the SQL statement to create these roles. This example is in script file CreateRoles.sql and it can be run from a DB2 command window:

```
db2 - tvf CreateRoles.sql
```

Example 3-1 Create roles

```
------------------------------------------------
-- DDL Statement for Role Sales
------------------------------------------------
CREATE ROLE Sales;

------------------------------------------------
```

```
-- DDL Statement for Role Mktg
------------------------------------------------
CREATE ROLE Mktg;

------------------------------------------------
-- DDL Statement for Role Admin
------------------------------------------------
CREATE ROLE Admin;

------------------------------------------------
-- DDL Statement for Role Payroll
------------------------------------------------
CREATE ROLE Payroll;

------------------------------------------------
-- DDL Statement for Role Pension
------------------------------------------------
CREATE ROLE Pension;

------------------------------------------------
-- DDL Statement for Role SalesMgr
------------------------------------------------
CREATE ROLE SalesMgr;

------------------------------------------------
-- DDL Statement for Role AdminMgr
------------------------------------------------
CREATE ROLE AdminMgr;

------------------------------------------------
-- DDL Statement for Role CEO
------------------------------------------------
CREATE ROLE CEO;

COMMIT WORK;

CONNECT RESET;

TERMINATE;
```

3.4.2 Assign privileges

The main advantage of roles is that they simplify administration and keep the security control of the database within the database. Prior to DB2 9.5, the database administrator would have had to assign privileges and authorities to each user as she joins or leaves individual departments or the entire company or rely upon the groups managed by the underlying operating systems and its security features.

You can grant DB2 privileges and authorities to a role, with one exception — SECADM is the only authority that must be granted to an Auth ID.

We already created roles to match the LUW Enterprise structure. Now we grant privileges to these roles and assign individuals to the roles. We assign the privileges to the roles, as shown in Table 3-1.

Table 3-1 Privileges for roles

Role	Privilege	Table
Sales	SELECT	LE_ORG LE_SALES
	UPDATE, INSERT	LE_SALES
Mktg	SELECT	LE_ORG LE_SALES
Admin	SELECT	LE_DEPARTMENT LE_EMPLOYEE LE_ORG
	UPDATE	LE_EMPLOYEE LE_DEPARTMENT
Payroll	SELECT	LE_DEPARTMENT LE_EMPLOYEE LE_EMP_PHOTO LE_EMP_RESUME LE_PENSIONS

	UPDATE
	INSERT
	DELETE
		LE_EMPLOYEE

Pension	SELECT
	UPDATE
	INSERT
	DELETE
		LE_PENSIONS

Example 3-2 shows the SQL statement to grant privileges to roles. This example is in script file AssignRoles.sql and it can be run from a DB2 command window:

```
db2 - tvf AssignRoles.sql
```

Example 3-2 Assign privileges

```
------------------------------------------------
-- DDL Statement for Role Sales
------------------------------------------------

GRANT SELECT ON TABLE db2inst1.le_org TO ROLE Sales;
GRANT SELECT, UPDATE, INSERT ON TABLE db2inst1.le_sales TO ROLE Sales;

------------------------------------------------
-- DDL Statement for Role Mktg
------------------------------------------------

GRANT SELECT ON TABLE db2inst1.le_org TO ROLE Mktg;
GRANT SELECT ON TABLE db2inst1.le_sales TO ROLE Mktg;

------------------------------------------------
-- DDL Statement for Role Admin
------------------------------------------------

GRANT SELECT ON TABLE db2inst1.le_department TO ROLE Admin;
GRANT SELECT ON TABLE db2inst1.le_employee TO ROLE Admin;
GRANT SELECT ON TABLE db2inst1.le_org TO ROLE Admin;
GRANT UPDATE ON TABLE db2inst1.le_department TO ROLE Admin;
GRANT UPDATE ON TABLE db2inst1.le_employee TO ROLE Admin;

------------------------------------------------
-- DDL Statement for Role Payroll
------------------------------------------------

GRANT SELECT ON TABLE db2inst1.le_department TO ROLE Payroll;
GRANT SELECT ON TABLE db2inst1.le_emp_photo TO ROLE Payroll;
GRANT SELECT ON TABLE db2inst1.le_emp_resume TO ROLE Payroll;
GRANT SELECT ON TABLE db2inst1.le_pensions TO ROLE Payroll;
GRANT SELECT ON TABLE db2inst1.le_employee TO ROLE Payroll;
GRANT UPDATE, INSERT, DELETE  ON TABLE db2inst1.le_employee TO ROLE
Payroll;

------------------------------------------------
-- DDL Statement for Role Pension
------------------------------------------------
```

```
GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE db2inst1.le_pensions TO
ROLE Payroll;


COMMIT WORK;

CONNECT RESET;

TERMINATE;
```

3.4.3 Granting membership to roles

The security administrator can use the GRANT ROLE statement to propagate roles. They can be assigned to individuals, groups, other roles, or even to PUBLIC. It can also be used to delegate authority by using the WITH ADMIN OPTION.

The WITH ADMIN OPTION allows the specified authorization ID the authority to grant or revoke the role-name to or from others. It does not allow the specified authorization ID to drop the role or to delegate the WITH ADMIN authority.

To use this statement you must be SECADM or you must have been granted the WITH ADMIN OPTION on the role.

Figure 3-3 shows the syntax of the GRANT ROLE statement.

```
                   .-,---------.
          .-ROLE-.  V           |
>>-GRANT--+------+----role-name-+-------------------------------->

       .-,---------------------------------.
       V                                   |
>--TO----+-+-------+--authorization-name-+-+--------------------->
         | +-USER--+                     |
         | +-GROUP-+                     |
         | '-ROLE--'                     |
         '-PUBLIC------------------------'

>--+-------------------+---------------------------------------><
   '-WITH ADMIN OPTION-'
```

Figure 3-3 GRANT ROLE syntax

We discuss the REVOKE ROLE statement and its consequences in 3.4.4, “Maintenance and administration of roles” on page 59.

You can design roles to reflect the hierarchy of an organization structure and build the security level into this hierarchy. The members in the top level role have all the authorities or privileges of the lower level roles. To design a hierarchy of security, grant the lower roles to the preceding level. For example, using the defined hierarchy for LUW Enterprises, by granting SalesMgr and AdminMgr roles to CEO role, you establish the hierarchy. The member with CEO role will have all the authorities and privileges of SalesMgr and AdminMgr.

In our scenario, we create the membership as shown in Example 3-3.

Example 3-3 Creating role membership

```
------------------------------------------------
-- Grant membership for Role CEO
------------------------------------------------
GRANT ROLE SalesMgr, AdminMgr to ROLE CEO;

GRANT ROLE CEO TO USER Mark;

------------------------------------------------
-- Grant membership for Role AdminMgr
------------------------------------------------
GRANT ROLE Admin, Payroll to ROLE AdminMgr;

GRANT ROLE AdminMgr TO USER Ally;

------------------------------------------------
-- Grant membership for Role SalesMgr
------------------------------------------------
GRANT ROLE Sales, Mktg to ROLE SalesMgr;

GRANT ROLE SalesMgr TO USER Saul;

------------------------------------------------
-- Grant membership for Role Admin
------------------------------------------------
GRANT ROLE Admin TO USER Adam, Debs, Mary;

------------------------------------------------
-- Grant membership for Role Mktg
------------------------------------------------
GRANT ROLE Mktg TO USER Mary, Anne, Rose;
```

```
------------------------------------------------
-- Grant membership for Role Payroll
------------------------------------------------
GRANT ROLE Payroll TO USER Adam, Pete, Yang;

------------------------------------------------
-- Grant membership for Role Pension
------------------------------------------------
GRANT ROLE Pension TO USER Yang;

------------------------------------------------
-- Grant membership to role Sales
------------------------------------------------
GRANT ROLE Sales TO USER Stan, Alan, Lori, Evan;
```

The security administrator can delegate the management of membership in a role to other users by specifying the WITH ADMIN OPTION on the GRANT ROLE SQL statement. For example, the security administrator could grant the project leader in each department the WITH ADMIN OPTION to their respective roles and they would be able to manage users as they join and leave their team. Example 3-4 shows the WITH ADMIN OPTION applied to the project leader for sales.

Example 3-4 WITH ADMIN OPTION example

```
------------------------------------------------
-- DDL Statement for User STAN
------------------------------------------------
GRANT ROLE Sales TO USER Stan WITH ADMIN OPTION;
```

If a new sales representative, Seth, joins the company then Stan can grant access to the role SALES to user Seth. See Example 3-5.

Example 3-5 Delegated grant example

```
CONNECT TO sample USER Stan USING password;

GRANT ROLE Sales TO USER Seth;
```

3.4.4 Maintenance and administration of roles

In this example it is not easy to see the benefits of role administration. However, if you look at a large company with thousands of employees in hundreds of roles, it becomes clearer. For example, if the company has an average of 20 sales representatives per branch, 10 branches per region or state, and 50 regions within the sales structure, suddenly you have 10,000 sales representatives and probably another 1,000 associated staff. If roles are not used, we would have had to grant privileges to each person or user group, which may not be able to reflect the company's organization hierarchy. Assume that we have different roles for each branch, region, and support roles — we still have over 500 roles. However, when a person joins the company, leaves the company, or moves position, it would be a simple grant or revoke of a role. If we were to assume an attrition rate of 5%, that would still be over 500 changes in a year. However, if we delegate the administration to region or branch levels, the administration tasks at these levels becomes easily manageable.

We have looked at the use of the GRANT command with roles. DB2 provides three further options and a function to manage roles:

- REVOKE ROLE
- COMMENT ON ROLE
- DROP ROLE
- AUTH_LIST_ROLES_FOR_AUTHID function

Revoke role

Access to a role can be revoked either by the security administrator or a user with the WITH ADMIN OPTION. The rules for revoking access to a role are the same as any other privilege. For example, Mary has been granted access to both the Mktg and Admin roles. Both roles provide SELECT on tables LE_ORG and LE_SALES. However, the Admin role also provides SELECT on LE_DEPARTMENT and LE_EMPLOYEE. Therefore Mary can run a query to join data from the LE_EMPLOYEE and LE_SALES tables. See Example 3-6

Example 3-6 Query on LE_EMPLOYEE & LE_SALES

```
SELECT e.firstnme, e.lastname, s.sales FROM
   db2inst1.le_employee e, db2inst1.le_sales s
   WHERE e.firstnme=s.sales_person
```

Mary can create a view to perform the join between these two tables. See Example 3-7.

Example 3-7 View on LE_EMPLOYEE and LE_SALES

```
CREATE VIEW total_sales (fname, lname, sales)
   AS SELECT e.firstnme, e.lastname, SUM(s.sales)
      FROM db2inst1.le_employee e, db2inst1.le_sales s
      WHERE e.firstnme=s.sales_person
      GROUP BY e.firstnme, e.lastname
```

If Mary had the access to the role admin revoked, the query would fail and the view would become inoperative.

When the WITH ADMIN option is revoked from a user, any authorities granted by the user are retained. For example, Stan has been granted WITH ADMIN on the role SALES. If he were to grant the role to a new salesperson, Seth, and later the WITH ADMIN option is removed, then Seth retains his access to the role sales. See Example 3-8.

Example 3-8 Revoke admin option

```
REVOKE ADMIN OPTION FOR ROLE sales FROM USER stan
```

Comment on role

The security administrator or someone with grant authority can add a comment on a role to aid the definition of the role. See Example 3-9. This information is held in the SYSCAT.ROLES system view in the column REMARKS.

Example 3-9 Example of comment on role

```
COMMENT ON ROLE pension IS ‘Pensions administrator has full authority
over Pensions Table.’
```

Drop a role

Only the security administrator can drop a role. In order to drop a role, it must not be referenced in a trusted context or in use with a current workload. Example 3-10 shows a DROP role example.

Example 3-10 Drop a role example

```
DROP ROLE pension
```

Query access to a role

DB2 9.5 provides a function, AUTH_LIST_ROLES_FOR_AUTHID, to query the usage of roles. The function has two parameters, AUTHID and TYPE.

Example 3-11 shows how to use the AUTH_LIST_ROLES_FOR_AUTHID function to find the roles assigned to user Ally.

Example 3-11 Query roles held by user Ally

```
DB2 SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE,
            ROLENAME, CREATE_TIME, ADMIN
   FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('ALLY', 'U') ) AS T
```

3.5 Comparisons with roles and groups

The key advantage of using roles verses groups in DB2 is that roles are managed by the database. That is, DB2 knows when membership in a role changes. Therefore DB2 can consider roles for all operations.

This is not the case with groups because they are managed outside the database (for example, controlled by the operating system, Microsoft Active Directory, or an LDAP server). Because of this, DB2 imposes some restrictions on when privileges and authorities assigned to groups are taken into account when performing an authorization check. These restrictions include not taking groups into account when creating the following objects:

- Views
- Triggers
- Materialized Query Tables (MQTs)
- Packages with static SQL
- SQL routines

Because roles are managed by DB2, this restriction does not apply to roles. That is, privileges granted to roles are taken into account when creating the views, triggers, MQT, packages with static SQL, and SQL routines.

Also, roles can be defined as a hierarchy, with built-in inheritance of authorities. Groups are outside of the control of DB2 and therefore group authorities cannot be granted to other groups. To design a hierarchy of security using a group, each level would have to be granted the authorities of the preceding level. For example, using the defined hierarchy for LUW Enterprises, we would need to grant all the authorities to the Admin group, then we would have to grant the same authorities to an AdminMgr group, and then again to CEO group. Alternatively, we would have to assign the respective user to each group defined so that Mark, as CEO, would need to be a member of all the base groups defined. Both methods complicate the administration needed to manage complex hierarchies.

Example of role versus group

Let us assume, given the roles created, that we want user Yang to create a trigger on the LE_PENSIONS table to insert a record on another table to keep an audit trail of the row deleted. Yang is a member of the Pension role, which has SELECT, UPDATE, INSERT, and DELETE on table DB2INST1.LE_PENSIONS. Yang has no create table authority.

For Yang to create this new trigger, we perform the following tasks:

1. Create a table LE_PEN_DELETES under the DBADM user DB2INST1:

   ```
   CREATE TABLE db2inst1.le_pen_deletes (X INT, D_DATE DATE)
   ```

2. Grant INSERT privilege to the role PENSION using the SECADM user DB2SEC:

   ```
   GRANT INSERT ON db2inst1.le_pen_deletes TO ROLE Pension
   ```

3. User Yang can now successfully create the trigger PENS_TRG because he is a member of the PENSION role. See Example 3-12.

   Example 3-12 Creating trigger on PENSIONS table

   ```
   C:\Program Files\IBM\SQLLIB\BIN>db2 CREATE TRIGGER PENS_TRG AFTER
   DELETE ON DB2INST1.LE_PENSIONS FOR EACH STATEMENT MODE DB2SQL INSERT
   INTO DB2INST1.LE_PEN_DELETES VALUES (1,CURRENT DATE)
   DB20000I  The SQL command completed successfully.
   ```

This could not have been replicated using groups, although we could have created the table and granted INSERT authority to a group. The CREATE TRIGGER would have failed, as demonstrated here:

1. Create a group outside DB2.

   We have already created the table LE_PEN_DELETES. To replicate the trigger creation, we create a Pension_gp group outside of DB2 and add user Ally to the group.

2. Grant the authorities to the group under the DBADM user DB2INST1, as shown in Example 3-13.

Example 3-13 Grant group privileges

```
GRANT CONNECT ON DATABASE TO GROUP Pension_gp
GRANT INSERT ON DB2INST1.LE_PEN_DELETES TO GROUP Pension_gp
GRANT DELETE ON DB2INST1.LE_PENSIONS TO GROUP Pension_gp
```

3. Create trigger using user Ally.

   Despite user Ally being a member of group Pension_gp and the group having the required authorities on the tables LE_PEN_DELETES and LE_PENSIONS, the create trigger statement fails (Example 3-14).

Example 3-14 Trigger creation failed

```
C:\Program Files\IBM\SQLLIB\BIN>db2 CREATE TRIGGER PENS_TRG AFTER
DELETE ON DB2INST1.LE_PENSIONS FOR EACH STATEMENT MODE DB2SQL INSERT
INTO DB2INST1.LE_PEN_DELETES VALUES (1,CURRENT DATE)
DB21034E  The command was processed as an SQL statement because it
was not a valid Command Line Processor command.  During SQL
processing it returned:
SQL0551N  "ALLY" does not have the privilege to perform operation
"INSERT" on object "DB2INST1.LE_PEN_DELETES". LINE NUMBER=1.
SQLSTATE=42501
```

3.6 Considerations

It is important to remember that all roles assigned to a user are available from the moment they establish a connection. Although the SET ROLE SQL statement is supported, it does not affect the user in anyway. There is no way to alter, enable, or disable roles assigned to a user once a connection is established.

There are some restrictions on the use of roles, for example:

- A role cannot own a database object.
- A role cannot be granted security administrator (SECADM) authority.
- A role can be granted to a group. However, permissions assigned to the role will not be considered if you create:
  - Packages containing static SQL
  - Views
  - Materialized Query Tables

   - Triggers
   - SQL routines

     Only roles granted to the user, directly or indirectly, are considered when creating these objects.

4

Trusted contexts and connections

Trusted contexts is a new security enhancement introduced in DB2 9.5. A trusted context establishes a trusted relationship between DB2 and an external entity, such as a Web server, application server, or another DB2 server. At connect time, the DB2 Server checks whether the connection matches the definition of a trusted context object in the database. When a match occurs, the database connection is said to be trusted. Using trusted contexts provides greater control while using restricted, sensitive privileges, and allows middle-tier servers or applications to assert the identity of the user to the database server.

In this chapter, we discuss the following topics:

- Definition of trusted contexts
- Setup and configuration
- User ID switching within trusted connections
- Role membership inheritance
- Switching user roles considerations

4.1 Definition of trusted contexts

Trusted contexts introduce the concept of context-sensitive privileges. Context-sensitive privileges allow an organization to have more control over when a privilege becomes available to a user. For example, an organization might want a manager to be able to access the payroll table only when the manager connects to the database from within the company offices, but not when the manager connects from home.

Traditionally, all the interactions with the database server occur through a database connection established by the middle tier, using a combination of a user ID and a credential, that identifies that middle tier to the database server. In other words, the database server uses the database privileges associated with the middle tier's user ID for all authorization checking and auditing that must occur for any database access, including those accesses performed by the middle tier on behalf of a user.

Example 4-1 illustrates the 3-tier application model. While the 3-tier application model has many benefits, having all interactions with the database server (for example, a user request) occur under the middle tier's authorization ID raises several security concerns.

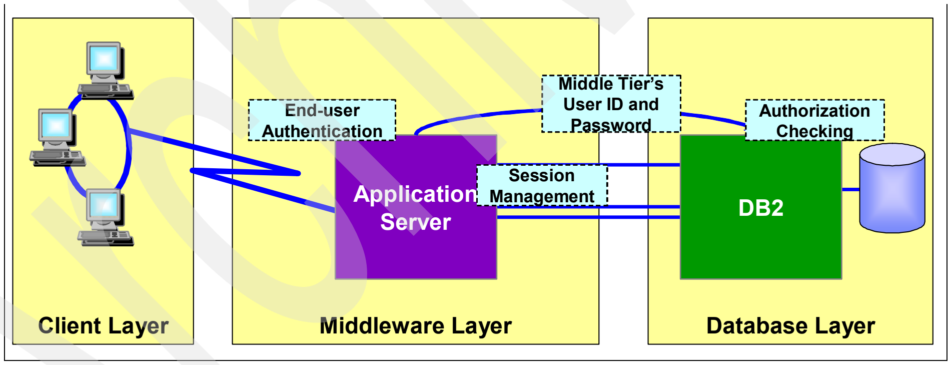


Figure 4-1 3-tier application model

The DB2 trusted contexts capability has been designed specifically to address these security concerns. The security is enhanced because it uses the actual user's identity and database privilege to perform database activities. The performance can be enhanced because there is no need for a new connection if you switch user, and if the switched user does require authentication, then there is no overhead with authenticating the a new user at the database server.

In a DB2 trusted context, the trust relationship is based upon the following set of attributes:

- System authorization ID

  This is the authorization ID representing the user who establishes a database connection.

- IP address (or domain name)

  This represents the IP address from which the database connection is established.

- Data stream encryption

  This is the encryption level (if any) used to encrypt the data communication between the database server and the connecting user.

Figure 4-2 shows the connection using a trusted context in a 3-tier environment.

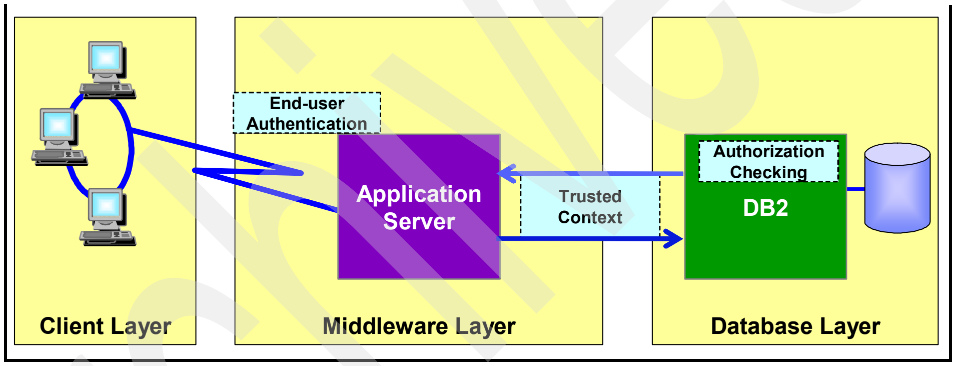


Figure 4-2 Trusted context in 3-tier environment

When a database connection is established, DB2 compares the attributes of that connection against the definitions of each trusted context object in the database. If the connection attributes match the definition of a trusted context object, that connection is referred to as a *trusted connection*. A trusted connection allows the initiator of such a connection to acquire additional capabilities that are not available to them outside the scope of that trusted connection. These additional capabilities vary depending on whether the trusted connection is explicit or implicit.

An *explicit* trusted connection is a trusted connection that is explicitly requested. It allows the initiator of such a trusted connection the ability to:

- Switch the current user ID on the connection to a different user ID with or without authentication.

- Acquire additional privileges that may not be available to them outside the scope of the trusted connection.

An explicit trusted connection can be established either through a DB2 Call Level Interface (CLI) application or through a DB2 Java Database Connectivity (JDBC) application. In other words, an application change is required to take advantage of explicit trusted connections.

An *implicit* trusted connection is a trusted connection that is not explicitly requested. It allows only the ability to acquire additional privileges. There are no changes required to an existing application in order to take advantage of implicit trusted connections.

The definition of a trusted context object must specify the following attributes:

- A system authorization ID representing the connection authorization ID that must be used by an incoming connection in order to be trusted.
- A list of IP addresses or domain names representing the IP addresses from which an incoming connection must originate in order to be trusted.
- A data stream encryption value representing the level of encryption that must be used by an incoming connection in order to be trusted. The data stream encryption value must be one of:
  - NONE: No specific level of encryption is required.
  - LOW: The authentication type on the database server must be DATA_ENCRYPT in order for an incoming connection to be trusted.
  - HIGH: Secure Socket Layer (SSL) encryption must be used for data communication between the DB2 client and the DB2 server in order for an incoming connection to be trusted.

4.2 Setup and configuration

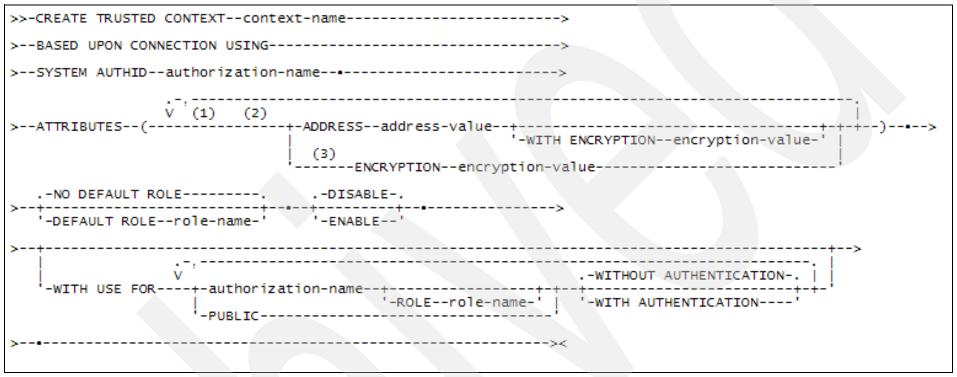
In DB2 9.5, you can create a trusted context object in the database that defines a trust relationship between the database and the middle-tier application server. The middle-tier application server can then establish an explicit trusted connection to the database, which gives the middle-tier application the ability to switch the current user ID on the connection to a different user ID, with or without authentication. Trust contexts provide a mechanism whereby the actual user's identity and database privileges are used for database requests performed by the middle-tier application on behalf of that user. In addition, they allow you to control when a privilege is made available to a database user. Note that only one trusted context can specified for an authid.

To demonstrate this process, we use the roles defined in 3.2, “Scenario - basic setup” on page 49, for the examples in this section.

4.2.1 Trusted context statement

You can create a trusted context object using the CREATE TRUSTED CONTEXT statement. Figure 4-3 shows the SQL statement syntax.

Note that only the database security administrator (SECADM) can create, alter, or drop a trusted context object.


```
>>-CREATE TRUSTED CONTEXT--context-name------------------------->

>--BASED UPON CONNECTION USING---------------------------------->

>--SYSTEM AUTHID--authorization-name--•------------------------->

                  .-,------------------------------------------------------------------------.
                  V   (1)    (2)                                                               |
>--ATTRIBUTES--(----------------+-ADDRESS--address-value--+------------------------------------+-+-+--)--•-->
                                |                          '-WITH ENCRYPTION--encryption-value-'   |
                                |  (3)                                                             |
                                '-------ENCRYPTION--encryption-value-------------------------------'

   .-NO DEFAULT ROLE---------.     .-DISABLE-.
>--+-------------------------+--•--+---------+--•--------------->
   '-DEFAULT ROLE--role-name-'     '-ENABLE--'

>--+-------------------------------------------------------------------------------------+-->
   |                  .-,---------------------------------------------------------------. |
   |                  V                                         .-WITHOUT AUTHENTICATION-. | |
   '-WITH USE FOR----+-authorization-name--+------------------+-+--+------------------------+-+-'
                     |                      '-ROLE--role-name-' |  '-WITH AUTHENTICATION----'
                     '-PUBLIC-----------------------------------'

>--•-------------------------------------------------------------><
```

Figure 4-3 CREATE TRUSTED CONTEXT statement syntax

The trusted context object appServerctx defined in Example 4-1 states that a database connection made by Pete from either host1.Admin.LUWent.com or host2.Payroll.LUWent.com and using secure socket layer (SSL) will be considered a trusted connection.

Example 4-1 Creating trusted context

```
CREATE TRUSTED CONTEXT appServerctx
BASED UPON CONNECTION USING SYSTEM AUTHID Pete
ATTRIBUTES (
      ADDRESS 'host1.Admin.LUWent.com',
      ADDRESS 'host2.Payroll.LUWent.com'
      ENCRYPTION ‘HIGH’)
ENABLE
```

4.2.2 Using trusted contexts

The DB2 security model, prior to DB2 9.5, was such that the privileges and authorities held by an authorization ID were universally applicable irrespective of the context of the action. For example, if an authorization ID is granted the SELECT privilege on the payroll database table, that authorization ID could exercise that privilege regardless of how it gains access to the database. The lack of control on when a privilege is available to a user can weaken overall security because the privilege may be used for purposes other than it was originally intended. For example, a company could desire, for better security, to make the SELECT privilege on the payroll table only available to an authorization ID only when it is connected from a computer located inside the company offices.

To provide control on when a privilege or a database authority becomes available to an authorization ID, the security administrator can grant that privilege or database authority to a role and assign that role to a trusted context. The privileges and database authorities granted to that role become available to an authorization ID only when that authorization establishes a trusted connection based upon that trusted context.

Assigning a role to a trusted context

Here we demonstrate how a user can obtain a privilege on a table through a trusted context.

Example 4-2 shows that Ally is in the ADMINMGR role and does not have access to the LE_SALES table, which belongs to the SALES and MKTG roles only.

Example 4-2 Query roles for Ally

```
DB2 SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE,
           ROLENAME, CREATE_TIME, ADMIN
   FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('ALLY', 'U') ) AS T

GRANTOR GRANTORTYPE GRANTEE  GRANTEETYPE ROLENAME CREATE_TIME                ADMIN
------- ----------- -------  ----------- -------- -------------------------- -----
DB2SEC  U           ADMINMGR R           PAYROLL  2007-10-09-01.06.15.000002 N
DB2SEC  U           ALLY     U           ADMINMGR 2007-10-09-01.06.15.078002 N
DB2SEC  U           ADMINMGR R           ADMIN    2007-10-12-00.16.05.671002 N
```

Therefore, if Ally tries to query the LE_SALES table, she receives an error message, as shown in Example 4-3.

Example 4-3 Select error on table LE_SALES

```
SQL0551N  "ALLY" does not have the privilege to perform operation
"SELECT" on object "DB2INST1.LE_SALES".  SQLSTATE=42501
```

Example 4-4 shows the SQL statement used to grant the SELECT privilege on the LE_SALES table to the role Mktg.

Example 4-4 Assigning a privilege to a role

```
GRANT select ON TABLE db2inst1.le_sales TO ROLE Mktg
```

Example 4-5 shows the SQL statement for assigning the Mktg role to the trusted context MktgTrustedContext. This trusted context definition states that when the administration manager (Ally) establishes a trusted connection based upon this trusted context, the set of privileges and authorities will include the role Mktg. Thus, the Administration manager also acquires the ability to select from the LE_SALES table through that role. This additional ability is not available to the administration manager when outside the scope of that trusted connection.

Example 4-5 Assigning a role to a trusted context

```
CREATE TRUSTED CONTEXT MktgTrustedContext
BASED UPON CONNECTION USING SYSTEM AUTHID ally
ATTRIBUTES (
      ADDRESS 'host1.Admin.LUW.com',
      ENCRYPTION 'HIGH')
DEFAULT ROLE Mktg
ENABLE
```

Note that the DB2 server must be able to access the Domain Name Server (DNS). The DNS is used to verify that the host address used in the statement is in a valid format. If the DNS is unable to validate an address, you receive an error message SQL0644N, as shown in Example 4-6.

Example 4-6 Address resolution error in trusted context

```
CREATE TRUSTED CONTEXT appServerctx BASED UPON CONNECTION USING AUTHID
Ally ATTRIBUTES ( ADDRESS 'host2.Admin.LUWent.com' , ADDRESS
'host2.Payroll.LUWent.com' , ENCRYPTION 'high' ) ENABLE
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During the SQL processing
it returned:
SQL0644N  Invalid value specified for key word "ADDRESS" in statement
"TRUSTED CONTEXT", SQLSTATE=42615
```

Now Ally would be able to query the LE_SALES table from within the trusted connection but not from any other connection.

4.2.3 Administering

Only the security administrator can manage trusted context objects, database roles, and audit policies. The security administrator is able to create, alter, drop, or comment on any of these objects.

You can use the ALTER statement to change the trusted context privileges given to a user or to disable a trusted context. The trusted context APPSERVER exists and is enabled. We can issue an ALTER TRUSTED CONTEXT statement, as shown in Example 4-7, to allow Ally to switch the authentication ID to Stan within the trusted context MktgTrustedContext without authentication. Switching the authentication ID is discussed in 4.3, "User ID switching within trusted connections" on page 75.

Example 4-7 Alter a trusted context to add a user

```
ALTER TRUSTED CONTEXT MktgTrustedContext
ADD USE FOR Stan WITHOUT AUTHENTICATION
```

You can disable the trusted context using the statement shown in Example 4-8, or add a comment using the COMMENT ON statement.

Example 4-8 Disabling a trusted context

```
ALTER TRUSTED CONTEXT MktgTrustedContext
DISABLE
```

You can also drop a trusted context using a DROP statement, as shown in Example 4-9.

Example 4-9 Dropping a trusted context

```
DROP TRUSTED CONTEXT MktgTrustedContext
```

The system catalogs have been changed in DB2 9.5 with two new catalog views and one view altered to store the trusted context details:

- SYSCAT.CONTEXTS

  Each row represents a trusted context.

- SYSCAT.CONTEXTATTRIBUTES

  Each row represents a trusted context attribute and there can be multiple rows for a trusted context.

- SYSCAT.SURROGATEAUTHIDS

  This is an altered view. When the TRUSTEDID represents a trusted context object, the row represents an authorization ID that can be used by a trusted connection. There can be multiple rows for a trusted context.

4.3 User ID switching within trusted connections

DB2 allows an established trusted connection to be used under a different authorization ID. To allow this, the trusted context must be defined with use for the specific ID. If PUBLIC is specified, it allows the trusted connection to be used by any authorization ID. When an explicit trusted connection is established, the initiator of such a trusted connection acquires the ability to switch the authorization ID of the connection to a different authorization ID. The ability to switch is possible only through a DB2 CLI application or a DB2 JDBC application.

For each trusted context object, the security administrator has the option to specify the list of IDs to whom switching is allowed, and whether authentication is required to carry out the switch. The security administrator can also specify a default role or an alternative role for individual IDs.

The trusted context definition in Example 4-10 shows that, upon the establishment of an explicit trusted connection based on this trusted context, authorization ID Ally can switch to Evan without the need to provide an authentication token. However, when switching to either Yang or Stan authentication is required.

Example 4-10 Trusted context with list of IDs

```
CREATE TRUSTED CONTEXT AdminUserCtxt
   BASED UPON CONNECTION USING SYSTEM AUTHID Ally
   ATTRIBUTES (ADDRESS   '192.168.99.188')
   WITH USE FOR Yang WITH AUTHENTICATION,
                Stan WITH AUTHENTICATION,
                Evan WITHOUT AUTHENTICATION,
   DEFAULT ROLE Admin
   ENABLE
```

When the ID Ally, on the trusted connection, is switched to new ID Yang, all traces of the connection environment under Ally are removed. In other words, the switching of IDs results in an environment that is identical to a new connection environment. For example, if Ally had any temporary tables or WITH HOLD cursors open, these objects are lost. The switch must be made on a transaction boundary or any open transaction will be rolled back and the connection will terminate with an error. It is good practice to complete all logical units of work or transactions with an explicit COMMIT WORK statement. This establishes a transaction boundary and ensures that any work in progress is completed before switching IDs.

User ID switching within a trusted context can be very useful in applications where you may need a user to perform operations that are outside of their normal duties. In the trusted context AdminUserCtxt, Ally would normally perform operations under the default role of Admin, which has only SELECT on tables LE_DEPARTMENT, LE_EMPLOYEE, and LE_ORG, LE_SALES. However, it might be necessary for her to update the LE_PENSIONS table as well as the LE_EMPLOYEE table. For example, a person in the SALES team, Evan, is promoted from junior to senior sales representative and he gets an increase in pay and pension benefits. This means that the LE_EMPLOYEE and LE_PENSIONS tables need to be updated. Normally, only Yang has update access to the LE_PENSIONS table. Therefore, by switching user IDs under the trusted context, Ally is able to assume the authorities and privileges of Yang to complete the transaction.

Example 4-11 demonstrates how to switch users in a CLI script. We also provide sample programs in C++ and Java in Appendix A, "Sample applications and scripts" on page 229.

Example 4-11 CLI program - switching user in trusted context

```
# setup the script
Option ECHO ON
Option CALLERROR ON

# setup the environment
SQLAllocEnv 1
SQLAllocConnect 1 1

# enable trusted connections using SQL_ATTR_USE_TRUSTED_CONTEXT (2561)
SQLSetConnectAttr 1, 2561, 1

# connect to a remote database
SQLConnect 1 SAMPLER SQL_NTS Ally SQL_NTS AllyPasswd SQL_NTS

# perform some work
SQLAllocStmt 1 1
SQLExecDirect 1 "SELECT * FROM DB2INST1.LE_EMPLOYEE WHERE
EMPNO='012214'" SQL_NTS
FetchAll 1
SQLFreeStmt 1 SQL_CLOSE
```

```
SQLExecDirect 1 "UPDATE DB2INST1.LE_EMPLOYEE SET JOB='SeniorSR' WHERE
EMPNO='012214'" SQL_NTS
SQLTransact 1 1 SQL_COMMIT

# switch the userid on the trusted connection using
SQL_ATTR_TRUSTED_CONTEXT_USERID (2562)
SQLSetConnectAttr 1, 2562, "Yang"
# when context switching requires a password, use
SQL_ATTR_TRUSTED_CONTEXT_PASSWORD (2563)
#SQLSetConnectAttr 1, 2563, "YangPasswd"

# perform some work using the new userid
SQLAllocStmt 1 2
SQLExecDirect 2 "SELECT * FROM DB2INST1.LE_EMPLOYEE WHERE
empno='012214'" SQL_NTS
FetchAll 2
SQLFreeStmt 2 SQL_CLOSE
SQLExecDirect 2 "UPDATE DB2INST1.LE_PENSIONS SET BASE=BASE + 2500 WHERE
EMPNO='012214'" SQL_NTS
SQLTransact 1 1 SQL_COMMIT

# clean up the environment
SQLDisconnect 1
SQLFreeConnect 1
SQLFreeEnv 1
```

4.4 Role membership inheritance

The security administrator can specify trusted contexts with both a default role and user-specific roles. The default role is inherited by all users of the trusted connection and explicitly specified users can automatically inherit additional privileges relating to the trusted context object.

Example 4-12 shows that the security administrator creates a trusted context object AdminUserCtxt for the manager of the administration department to enable the manager (Ally) switch authorization IDs.

Example 4-12 Creating a trusted context that assigns an alternative role

```
CREATE TRUSTED CONTEXT AdminUserCtxt
   BASED UPON CONNECTION USING SYSTEM AUTHID Ally
   ATTRIBUTES (ADDRESS   '192.168.99.188')
   WITH USE FOR Yang WITH AUTHENTICATION,
              Stan WITH AUTHENTICATION,
              Evan WITHOUT AUTHENTICATION ROLE Sales,
   DEFAULT ROLE AdminMgr
   ENABLE
```

When Ally establishes a trusted connection, she will have her normal privileges, as the default role is AdminMgr. These same privileges are also inherited by Yang or Stan when the current ID on the trusted connection is switched to their IDs, that is, if the user ID of the connection is switched to Stan or Evan at some point, then they would also inherit the trusted context default role, AdminMgr.

The security administrator may choose to have Evan inherit a different role from the trusted context default role. When the ID on the trusted connection is switched to Evan, they do not inherit the default role, they inherit the role Sales, as specified by the security administrator. When a user-specific role is specified in the definition of a trusted context, it always overrides the default trusted context role.

Table 4-1 shows the roles held by Ally when switching her ID in the trusted context.

Table 4-1 Trusted context roles inheritance

User ID	**Roles inherited**
Ally	Admin, Payroll
Yang	Admin, Payroll, Pension
Stan	Admin, Payroll, Sales
Evan	Sales

4.5 Switching user roles considerations

To switch a user on a trusted connection, the request must be made explicitly, using one of the APIs in Table 4-2 within the application. Table 4-2 lists the application APIs for switching a user on a trusted connection request.

Table 4-2 APIs for switching user on a trusted connection request

Application	API
CLI/ODBC	SQLSetConnectAttr
JAVA	getDB2Connection, reuseDB2Connection
XA	SQLSetConnectAttr

The algorithm DB2 uses to authorize the switching of the current user ID on a connection can be described as follows:

- If the switch request is:
  - Not made from an explicit trusted connection, the connection is shut down and an error message is returned (SQLSTATE 08001, SQLCODE -30082 with reason code 41).
  - Not made on a transaction boundary, the transaction is rolled back, the connection is put into an unconnected state, and an error message is returned (SQLSTATE 58009, SQLCODE -30020).
  - Made from within a stored procedure, an error message is returned (SQLCODE -30090, reason code 29), indicating that this is an illegal operation in this environment.
  - Made on an instance attach (rather than a database connection), the attachment is shut down and an error message is returned (SQLCODE -30005).
  - Made with an authorization ID that is not allowed on the trusted connection, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
  - Made with an authorization ID that is allowed on the trusted connection WITH AUTHENTICATION, but the appropriate authentication token is not provided, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
  - Made with a user ID allowed on the trusted connection, but that user ID does not hold the CONNECT privilege on the database, the connection is put in an unconnected state and an error message is returned (SQLSTATE 08004, SQLCODE -1060).

- If the trusted context object associated with the trusted connection:
  - Is disabled, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
  - Is dropped, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
  - Changes the system authorization ID, and a switch request for that trusted connection is made, error (SQLSTATE 42517, SQLCODE -20361) is returned, and the connection is put in an unconnected state.
- If the trusted context system authorization ID appears in the WITH USE FOR clause, DB2 honors the authentication setting for the system authorization ID on switch user request to switch back to the system authorization ID.
- If the trusted context system authorization ID does not appear in the WITH USE FOR clause, then a switch user request to switch back to the system authorization ID is always allowed even without authentication.

4.6 Problem determination

The system catalog views SYSCAT.CONTEXTATTRIBUTES and SYSCAT.CONTEXTS contain all the definition information about trusted contexts in a database. To establish whether a trusted context is defined and enabled for a particular AuthID, see the sample query in Example 4-13.

Example 4-13 Query context

```
SELECT CONTEXTNAME, SYSTEMAUTHID, ENABLED FROM SYSIBM.SYSCONTEXTS
   WHERE SYSTEMAUTHID='Stan'
```

To help determine why a particular connection was not considered a trusted connection, the db2pd tool has been extended so that it also returns the following information when the -application option is specified:

- The IP address from which the connection originated
- The data stream encryption level used by the connection (NONE, LOW, or HIGH)
- The system authorization ID of the connection

Using the catalog views and the **db2pd** command, the security administrator can determine why a particular connection was not considered. For example, the trusted context is shown as existing and enabled in the catalog. However, the program failed authentication with a switch.

Example 4-14 shows the command usage and a sample output. In the example, the password for the switch ID was missing and therefore the switched failed. This is shown by the RollbackActive status in the db2pd output. This means that the connection was disconnected and the transaction has to be rolled back.

Example 4-14 db2pd command

```
C:\ITSO 7D25>db2pd -db sample -applications

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:02

Applications:
Address    AppHandl [nod-index] NumAgents  CoorEDUID  Status                 C-
AnchID C-StmtUID  L-AnchID L-StmtUID  Appid
                       WorkloadID  WorkloadOccID
0x7AF4A8B0 64       [000-00064] 1          3904       ConnectCompleted       0
       0          0        0          *LOCAL.DB2.071024104405
                       0           0
0x7AF47A50 63       [000-00063] 1          3900       ConnectCompleted       0
       0          0        0          *LOCAL.DB2.071024104404
                       0           0
0x7AF44BF0 62       [000-00062] 1          3896       ConnectCompleted       0
       0          0        0          *LOCAL.DB2.071024104403
                       0           0
0x7A8BBFE0 61       [000-00061] 1          3892       ConnectCompleted       0
       0          0        0          *LOCAL.DB2.071024104402
                       0           0
0x7AECCC10 60       [000-00060] 1          1312       RollbackActive         0
       0          231      1          192.168.73.134.5892.07102410440
                       0           0

External Connection Attributes
Address    AppHandl [nod-index] ClientIPAddress                          Encrypt
ionLvl SystemAuthID

0x7AF4A8B0 64       [000-00064] n/a                                      None
       DB2INST1

0x7AF47A50 63       [000-00063] n/a                                      None
       ALLY

0x7AF44BF0 62       [000-00062] n/a                                      None
       ALLY

0x7A8BBFE0 61       [000-00061] n/a                                      None
       ALLY

0x7AECCC10 60       [000-00060] 192.168.73.134                           None
```

If the program is re-run with the appropriate password, Example 4-15 shows that the connection was trusted and conformed to all attributes defined in the database.

Example 4-15 db2pd output showing the connection is trusted

```
Trusted Connection Attributes
Address    AppHandl [nod-index] TrustedContext ConnTrustType
RoleInherited

0x7AF4A8B0 51       [000-00051] ADMINUSERCTXT explicit trusted
connection  n/a
```

Another possible area for failure is the encryption level if there is a mis-match between the encryption of the data stream and the definition of the trusted context. As a general rule of thumb, the encryption of the data stream must match or exceed the definition of the trusted context.

Table 4-3 summarizes when the data stream encryption level used by a connection matches the encryption attribute of a trusted context.

Table 4-3 Data stream encryption level

Data stream encryption level used by the connection	Value of the encryption attribute of the trusted context	Does the data stream encryption match the encryption attribute?
No encryption	NONE	Yes
No encryption	LOW	No
No encryption	HiGH	No
Low encryption (DATA_ENCRYPT)	NONE	Yes
Low encryption (DATA_ENCRYPT)	LOW	Yes
Low encryption (DATA_ENCRYPT)	HiGH	No
High encryption (SSL)	NONE	Yes
High encryption (SSL)	LOW	Yes
High encryption (SSL)	HiGH	Yes

5

Label-based access control

Label-based access control (LBAC) is a security feature introduced in DB2 9. It provides content-based authorization using *security labels*. The LBAC allows you to control read and write access of users to individual rows and columns at the table level. Data that is protected by a security label is called protected data.

One of the key advantages of using the LBAC to protect sensitive data is that no authority, including SYSADM, DBADM, or SECADM, has any privileges (direct or indirect) to access data protected with LBAC. The SECADM configures the LBAC by granting users security labels to access protected data, but the SECADM user cannot access the data.

The LBAC is commonly used in installations where strict access control is required. The LBAC is also an ideal solution for anyone using table views to limit security.

In this chapter, we discuss the following topics:

- Overview of the LBAC
- New LBAC features introduced in DB2 9.5
- Setup and configuration of the LBAC
- Maintenance and administration tasks associated with LBAC

5.1 Overview

LBAC allows you to control access to the data at the column level and row level based on security labels. The key DB2 LBAC concepts are security label components, security policies, and security labels.

Figure 5-1 illustrates the basic concept of LBAC. The security administrator, a user with SECADM authority, performs the LBAC security setup. The security administrator configures the LBAC system by creating security components and policies. A *security policy* describes the criteria that are used to decide who has access to specific data. Under the security policy, the security administrator creates security labels and associates the label with the rows and columns to be protected. The security administrator also associates labels with users. When a protected table is accessed, the LBAC policies compare the data labels with the user's label to determine whether the user has access to the specific row or column. It is a unique aspect of LBAC that users only see the rows that they are allowed to see. Each person gets a consistent view of the data, but it is not the same view.

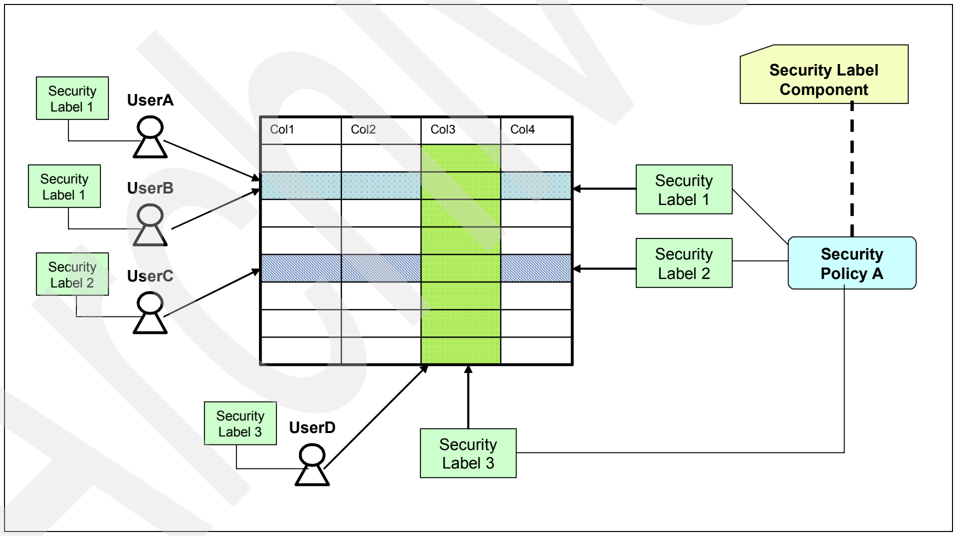


Figure 5-1 LBAC overview

Column-level access control

A column in a table can be secured with a security label. When accessing such a column, the user's security label is compared with the column's security label. If the labels match, access is granted and the column can be accessed. If the labels do not match, the access is denied and an error is returned.

There is no storage overhead when using LBAC to protect a column. The column’s security label is stored in the system catalog, not in the table itself.

Row-level access control

With LBAC, it is possible to assign security labels to the individual data rows. This kind of access control requires a column of the DB2SECURITYLABEL type to be present in the table where security label data is stored. DB2 applies LBAC rules when a user accesses a table protected with LBAC, causing the table content to potentially appear differently for different users. The rules allow the users to see all the rows, some of the rows, or none of the rows, depending on their security label and the security labels protecting the data rows in the table.

- Unauthorized access to a table with LBAC-secured rows results in an empty table. The LBAC does not return any error message.
- Protecting a table at the row level using LBAC requires additional storage. There must be present one row security label column of type DB2SECURITYLABEL. The actual amount of storage needed depends on the type of security label chosen. The total cost per row is (N*8 + 4) bytes, where N is the number of components in the security policy protecting the table.

5.1.1 LBAC enhancements in DB2 9.5

DB2 9.5 introduces support for granting security labels and exemptions to roles and groups. Also, it is possible now to alter security label components and security policies without the need to drop and recreate them.

The following is a summary of LBAC enhancements:

- A new ALTER SECURITY LABEL COMPONENT statement for adding new elements to a security label component.
- A new ALTER SECURITY POLICY statement allows for modifying a security policy.
- The GRANT SECURITY LABEL statement now allows you to grant security labels to roles and groups.
- The REVOKE SECURITY LABEL statement now allows you to revoke security labels from roles and groups.
- The GRANT EXEMPTION statement now allows you to grant exemptions to roles and groups.
- The REVOKE EXEMPTION statement now allows you to revoke exemptions from roles and groups.

5.2 Security label components

A security label component represents any criteria upon which access control is based. Typical examples of such criteria are:

- Level of trust, for example, top secret, secret, confidential, or public
- Department, such as CEO, management, payroll, sales, and so on

The security label component consists of one or more criteria (elements) grouped together in specified relationship:

SETs	Unordered lists of elements
ARRAYs	Ordered list of elements
TREEs	Elements arranged in a tree structure

The SQL statement for security label components includes:

- CREATE SECURITY LABEL COMPONENT - For creating a security label component.
- DROP SECURITY LABEL COMPONENT - For dropping a security label component.
- ALTER SECURITY LABEL COMPONENT - This is a new SQL statement introduced in DB2 9.5 for adding a new element to an existing security label component.

Figure 5-2 shows the syntax of the CREATE SECURITY LABEL COMPONENT statement.

```
>>-CREATE SECURITY LABEL COMPONENT--name--+-| array-clause |-+-><
                                          +-| set-clause |---+
                                          '-| tree-clause |--'

array-clause
             .-,-------.
             V         |
|--ARRAY--[----string--+--]----------------------------|

set-clause
           .-,-------.
           V         |
|--SET--{----string--+--}------------------------------|

tree-clause
|--TREE--(--string---ROOT--+---------------------------------+--)--|
                           | .-----------------------------. |
                           | V                             | |
                             '---,--string---UNDER--string--+-'
```

Figure 5-2 Syntax of the CREATE SECURITY LABEL COMPONENT statement

The components (elements) within the security label component can maintain certain relationships. They can be unordered or ordered as an array or a tree.

5.2.1 SET

The components of type SET are unordered lists of elements. There is no relationship between them. The only comparison that can be made for elements of this component type is whether or not a given element is in the list.

An example of such elements is a list of colors, as shown at Figure 5-3 and Example 5-1, where there is no relationship between the colors.

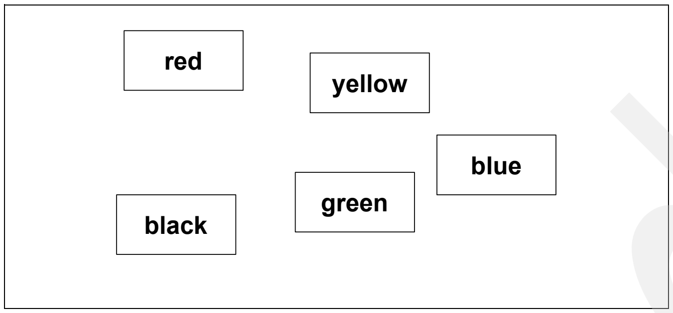


Figure 5-3 Elements suitable for type SET

Example 5-1 Creating security label component of type SET

```
CREATE SECURITY LABEL COMPONENT colors SET {'red', 'yellow', 'blue',
'black', 'green'}
```

5.2.2 ARRAY

The components of type ARRAY are ordered lists of elements. They have a linear relationship, where the first element listed has the highest value and the last element listed has the lowest value.

Figure 5-4 and Example 5-2 show an example of the ARRAY component type, where the position of the elements in the list defines the linear hierarchy of the security levels.

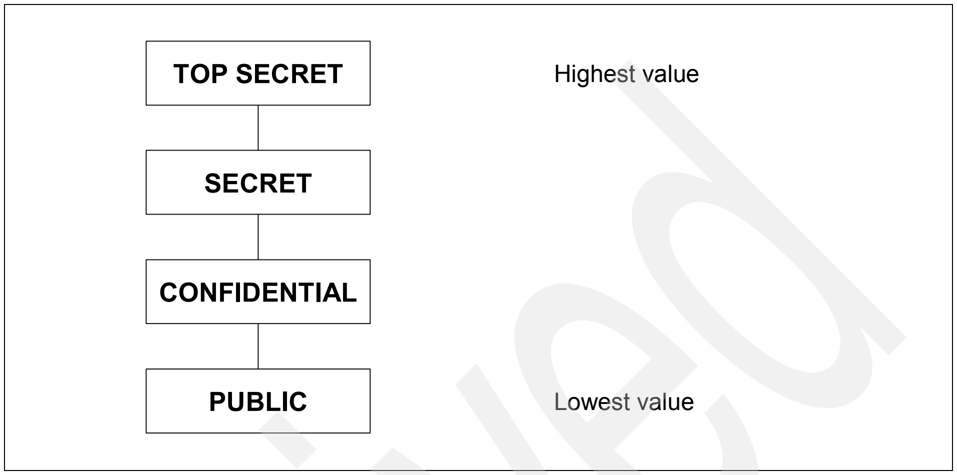


Figure 5-4 Elements suitable for type ARRAY

Example 5-2 Creating security label component of type ARRAY

```
CREATE SECURITY LABEL COMPONENT securityLevel ARRAY ['TOP SECRET',
'SECRET', 'CONFIDENTIAL', 'PUBLIC']
```

5.2.3 TREE

In the TREE type of component, the elements are treated as though they are arranged in a tree structure. This relationship requires one ROOT element. All the other elements need to be defined, as they are *under* another *existing* element. This is a simple and effective way of defining a tree structure. Any tree relationship can be described this way (parent, child, sibling, ancestor, descendant).

Figure 5-5 and Example 5-3 show an example of the TREE component type with the roles hierarchy.

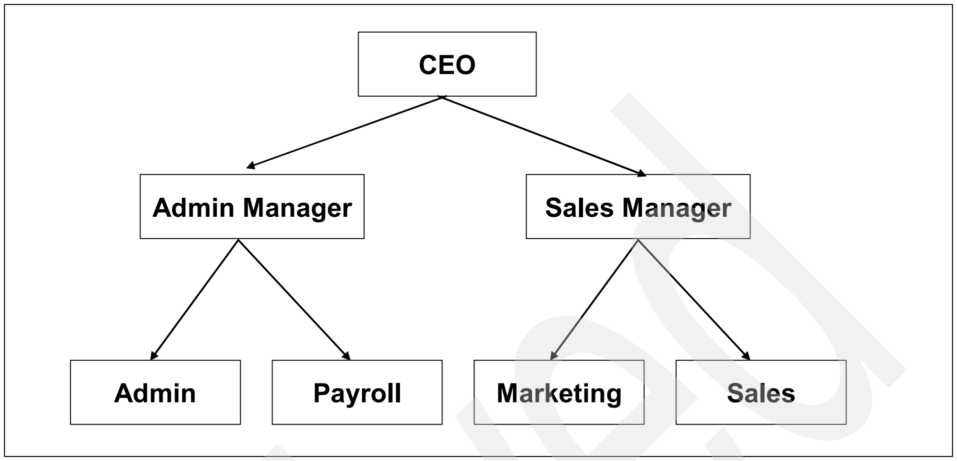


Figure 5-5 Elements suitable for type TREE

Example 5-3 Creating security label component of type TREE

```
CREATE SECURITY LABEL COMPONENT roles TREE
(
  'CEO'           ROOT,
  'Admin Manager' UNDER 'CEO',
  'Sales Manager' UNDER 'CEO',
  'Admin'         UNDER 'Admin Manager',
  'Payroll'       UNDER 'Admin Manager',
  'Marketing'     UNDER 'Sales Manager',
  'Sales'         UNDER 'Sales Manager'
)
```

5.3 Security policies

A security policy describes the criteria that is used to decide *who* has read or write access to individual rows and individual columns of a table. A security policy defines the structure of a security label and also access rules, referred as DB2LBACRULES. These rules are predefined in DB2. There are read access rules and write access rules.

Every protected table must have one and only one security policy associated with it. However, one security policy can be used by more than one table. Rows and columns in that table can only be protected with security labels that are part of that security policy, and all access of protected data follows the rules of that policy.

The SQL statements for security policies are:

- CREATE SECURITY POLICY - for creating a security policy
- DROP SECURITY POLICY - for dropping a security policy.
- ALTER SECURITY POLICY - This is a new SQL statement introduced in DB2 9.5 for modifying a security policy.

Figure 5-6 shows the syntax of the CREATE SECURITY POLICY statement.

```
>>-CREATE SECURITY POLICY--security-policy-name----------------->

              .-,--------------.
              V                |
>--COMPONENTS----component-name-+--WITH DB2LBACRULES------------>

   .-OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL-.
>--+----------------------------------------------+------------><
   '-RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL-'
```

Figure 5-6 Syntax of the CREATE SECURITY POLICY statement

5.3.1 Policies and rules

When creating a security policy, one or more security label components must be specified along with an LBAC rule set.

An LBAC rule set is a predefined set of rules that are used when comparing security labels. When the values of two security labels are compared, one or more of the rules in the rule set will be used to determine whether one value blocks another. At the time of writing, the only supported rule set is DB2LBACRULES. The predefined DB2 LBAC rules (DB2LBACRULES) are divided into two categories:

- Read access rules
  - DB2LBACREADARRAY

    The array component of the user's security label must be greater than or equal to the array component of the object's security label.
  - DB2LBACREADSET

    The set component of the user's security label must include the set component of the object's security label.

  - DB2LBACREADTREE

    The tree component of the user's security label must include at least one of the elements in the tree component of the object's security label (or the ancestor of one such element).

- Write access rules
  - DB2LBACWRITEARRAY

    The array component of the user's security label must be *equal* to the array component of the object's security label.
  - DB2LBACWRITESET

    The set component of the user's security label must include the set component of the object's security label.
  - DB2LBACWRITETREE

    The tree component of the user's security label must include at least one of the elements in the tree component of the object's security label (or the ancestor of one such element).

Example 5-4 creates a security policy securityClearance using security component securityLevel. The policy securityClearance specifies that an INSERT or UPDATE statement with and explicitly specified security label statement fails if the user is not authorized to write the security label.

Example 5-4 Creating security policy

```
CREATE SECURITY POLICY securityClearance COMPONENTS securityLevel WITH
DB2LBACRULES RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL;
```

It is possible to allow a user, a group, or a role to bypass a particular rule in a security policy using a special privilege called *exemption*. An exemption can be granted using the GRANT EXEMPTION ON RULE statement and revoked using the REVOKE EXEMPTION ON RULE statement.

All the DB2 LBAC predefined rules that exemption can be granted or revoked on are:

- DB2LBACREADARRAY
- DB2LBACREADSET
- DB2LBACREADTREE
- DB2LBACWRITEARRAY WRITEDOWN
- DB2LBACWRITEARRAY WRITEUP
- DB2LBACWRITESET
- DB2LBACWRITETREE

Note: The DB2LBACWRITEARRAY rule can be customized using WRITEDOWN or WRITEUP options. By default, the DB2LBACWRITEARRAY rule allows you to write to the objects where their array level is *equal* to the user's security label level. When WRITEDOWN or WRITEUP is granted to a user, it allows the user to write to an array level lower or higher, respectively, than their own array level.

Example 5-5 shows using GRANT EXEMPTION ON RULE to allow Ally and Mark write security labels with equal and *also* lower value.

Example 5-5 Creating security policy and granting a LBAC rule exemption

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN FOR
securityClearance TO USER Ally, Mark;
```

5.4 Security labels

A security label is a database object that describes a certain set of security criteria. Security labels are applied to data in order to protect the data. They are granted to users to allow them to access protected data.

A security label name consists of two parts, a qualifier and the security label name. The qualifier is a security policy name that defines the scope where the security label is defined, for example, securityClearance.secretSecLabel, where securityClearance is the security policy name and secretSecLabel is the actual security label name.

The LBAC relies on security labels to perform access control. Security labels can be assigned to users, data rows, and columns:

- User: a security label that is granted to a database user
- Row: a security label associated with a data row of a database table
- Column: a security label associated with a column of a database table

When assigning a security label to a user, the security administrator has the flexibility to grant that security label for read access only, for write access only, or for both. A user can have at most one security label assigned explicitly at a time. It is not possible to grant multiple security labels to a user. However, a user can acquire more implicitly through groups and roles. You cannot alter a security label. The only way of modifying a security label is to drop and recreate it.

SQL statements that work on security labels are:

- CREATE SECURITY LABEL - for creating a security label
- DROP SECURITY LABEL - for dropping a security label
- GRANT SECURITY LABEL - for granting a security label to a user, a group, or a role
- REVOKE SECURITY LABEL - for revoking a security label from a user, a group, or a role

Figure 5-7 shows the syntax of the CREATE SECURITY LABEL statement.

```
>>-CREATE SECURITY LABEL--security-label-name-------------------->

   .-,----------------------------------------------.
   |                            .-,---------------. |
   V                            V                 | |
>----COMPONENT--component-name----string-constant-+-+----------><
```

Figure 5-7 Syntax of the CREATE SECURITY LABEL statement

Note: The CREATE SECURITY LABEL statement allows you to specify one or more valid elements for a specific component. This is true for SET or TREE components only. An attempt to use more than one element for the ARRAY component type results in an error. To effectively specify more than one ARRAY element, use the DB2LBACWRITEARRAY WRITEDOWN or WRITEUP rule exemptions, as discussed in 5.3.1, “Policies and rules” on page 91.

To grant a security label use the SQL statement GRANT SECURITY LABEL. Figure 5-8 shows the syntax.

```
>>-GRANT SECURITY LABEL--security-label-name-------------------->


       .-,----------------------------.
       V                              |
>--TO----+-------+--authorization-name-+------------------------>
         +-USER--+
         +-GROUP-+
         '-ROLE--'

   .-FOR ALL ACCESS---.
>--+------------------+--------------------------------------><
   +-FOR READ ACCESS--+
   '-FOR WRITE ACCESS-'
```

Figure 5-8 Syntax of the GRANT SECURITY LABEL statement

Example 5-6 shows creating a security label and granting read and write access to a user.

Example 5-6 Creating a security label and granting read/write access to a user

```
CREATE SECURITY LABEL securityClearance.secret COMPONENT securityLevel
'SECRET';

GRANT SECURITY LABEL securityClearance.secret TO USER Ally;
```

5.5 Set up LBAC

To set up the LBAC security we need to follow these steps in *this* order:

1. Create a security label component.
2. Create a security policy.
3. Create a security label.
4. Secure table with LBAC.

Note: The LBAC subsystem can be configured by a security administrator (SECADM) only.

In this section, we use the LE_PENSIONS table to demonstrate the LBAC setup procedure to secure data at the row level, table LE_EMPLOYEE to secure data at the column level, and finally table LE_SALES to secure data at the both row and column levels. Figure 5-9 illustrates the LBAC objects used in this section as well as the tables involved.

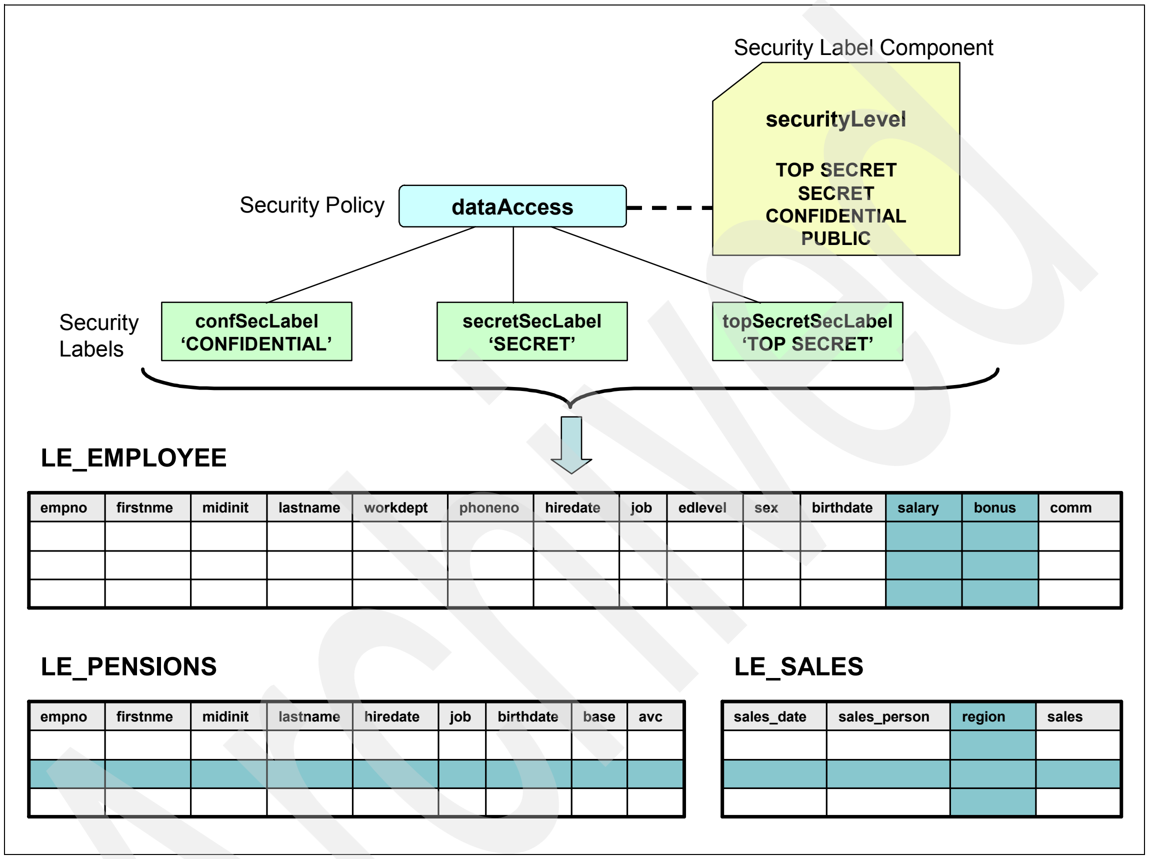

Security Label Component
securityLevel
TOP SECRET
SECRET
CONFIDENTIAL
PUBLIC
Security Policy
dataAccess
Security Labels
confSecLabel 'CONFIDENTIAL'
secretSecLabel 'SECRET'
topSecretSecLabel 'TOP SECRET'
LE_EMPLOYEE
empno firstnme midinit lastname workdept phoneno hiredate job edlevel sex birthdate salary bonus comm
LE_PENSIONS
empno firstnme midinit lastname hiredate job birthdate base avc
LE_SALES
sales_date sales_person region sales

Figure 5-9 LBAC objects for examples

5.5.1 Creating a security label component

The first step in setting up the LBAC security for a table is creating a security label component using the SQL statement CREATE SECURITY LABEL COMPONENT.

Example 5-7 shows creating a security label component of the ARRAY type.

Example 5-7 Creating an array security label component

```
CREATE SECURITY LABEL COMPONENT securityLevel ARRAY ['TOP SECRET',
'SECRET', 'CONFIDENTIAL', 'PUBLIC']
```

5.5.2 Create a security policy

The second step in setting up the LBAC security is creating a security policy using the SQL statement CREATE SECURITY POLICY.

Example 5-8 creates a security policy dataAccess for security label component securityLevel.

Example 5-8 Creating a security policy

```
CREATE SECURITY POLICY dataAccess COMPONENTS securityLevel WITH
DB2LBACRULES RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
```

5.5.3 Create a security label

The third step in the LBAC setup is creating a security label. Security labels can be assigned to users, data rows, and columns. When assigning a security label to a user, the security administrator has the flexibility to grant that security label for read access only, for write access only, or for both.

Example 5-9 shows creating security labels and granting them to the database administrator so that the administrator can add the label to a table.

Example 5-9 Creating and granting security labels

```
CREATE SECURITY LABEL dataAccess.topSecretSecLabel COMPONENT
securityLevel 'TOP SECRET';
GRANT SECURITY LABEL dataAccess.topSecretSecLabel TO USER Mark for ALL
ACCESS;

CREATE SECURITY LABEL dataAccess.secretSecLabel COMPONENT securityLevel
'SECRET';
GRANT SECURITY LABEL dataAccess.secretSecLabel TO USER Ally for ALL
ACCESS;

CREATE SECURITY LABEL dataAccess.confSecLabel COMPONENT securityLevel
'CONFIDENTIAL';
```

```
GRANT SECURITY LABEL dataAccess.confSecLabel TO USER Yang for ALL
ACCESS;
```

5.5.4 Secure table with LBAC

Once a security label is created, it can be used to secure a table. In order to enable any LBAC security on a table, a security policy must be assigned to the table. This can be done using either the CREATE TABLE statement with the SECURITY POLICY clause when creating a new table, or using the ALTER TABLE statement and specifying the ADD SECURITY POLICY clause when modifying an existing table. One table can have only one security policy.

A security policy can be added to a table by a security administrator or any user having CREATETAB authority (for a new table) or ALTER privilege (for an existing table). To add a column or row protection to a table, the privileges held by the authorization ID of the statement must include at least a *security label* from the security policy associated with the table.

Column protection

In order to add data protection to a table at the column level, you must alter a column and specify the SECURED WITH clause. A column can be secured when creating a table or later by altering the table.

New table

Use the CREATE TABLE statement to define a new table, and specify the SECURITY POLICY clause to assign a security policy to the table. Then for each column that needs to be protected by a security label use the SECURED WITH clause. Example 5-10 creates a table with a security policy and secures one column with a security label.

Example 5-10 Creating a table with one secured column

```
CREATE TABLE db2inst1.le_employee
(
  empno CHAR(6) NOT NULL,
  firstnme VARCHAR(12) NOT NULL,
  midinit CHAR(1),
  lastname VARCHAR(15) NOT NULL,
  workdept CHAR(3),
  phoneno CHAR(4),
  hiredate DATE,
  job CHAR(8),
  edlevel SMALLINT NOT NULL,
  sex CHAR(1),
  birthdate DATE,
  salary DECIMAL(9,2) SECURED WITH secretSecLabel,
  bonus DECIMAL(9,2),
  comm DECIMAL(9,2)
)
SECURITY POLICY dataAccess
```

Existing table

An existing table, originally created without a security policy and security labels, can be secured using the ALTER TABLE statement. You need to add a security policy to the table using ADD SECURITY POLICY. Once the table has a security policy assigned, alter a column using the SECURED WITH clause.

Example 5-11 alters an existing table db2inst1.le_employee and additionally secures the column BONUS.

Example 5-11 Securing a column in an existing table

```
ALTER TABLE db2inst1.le_employee
  ALTER COLUMN bonus SECURED WITH secretSecLabel;
```

Reading data from protected columns

To allow a user to read data from the protected column, You need to grant the user the privilege to use the security label, as shown in Example 5-12.

Example 5-12 Grant access to the protected column

```
GRANT SECURITY LABEL dataAccess.secretSecLabel TO USER Ally for ALL
ACCESS;
```

Example 5-13 shows that Ally, who holds the security label, can access the table. The user can access all the columns. No error is reported.

Example 5-13 Reading all the columns by a user holding the security label

```
CONNECT TO sample USER Ally

SELECT * FROM db2inst1.le_employee

...

14 record(s) selected.
```

Example 5-14 shows an error when a user who does not have the required security label tries to access a secured column. Note that even the security administrator DB2SEC cannot read the data.

Example 5-14 Error while reading all the columns for non-authorized user

```
CONNECT TO sample USER db2sec

SELECT * FROM db2inst1.le_employee

SQL20264N  For table "DB2INST1.LE_EMPLOYEE", authorization ID "DB2SEC"
does not have "READ" access to the column "BONUS".  SQLSTATE=42512
```

Example 5-15 shows the same user reading a non-secured column without any error.

Example 5-15 User can read data

```
CONNECT TO sample USER db2sec

SELECT lastname FROM db2inst1.le_employee

...

14 record(s) selected.
```

Inserting data into protected columns

When inserting data to a protected column, the user has to have the INSERT privilege on the table, and the LBAC credentials of the current user are compared with the security label protecting that column. Based on this comparison, access is either blocked or allowed.

Example 5-16 shows an attempt to insert a row when the user does not hold a security label allowing it.

Example 5-16 Attempt to insert data when the user is not allowed to write to column

```
CONNECT TO sample USER Yang

INSERT INTO db2inst1.le_employee VALUES ('011090', 'JOHN', '',
'POWELL', 'CI5', '1007', '08/11/1997', 'SALES', 17, 'M', '08/11/1957',
101230.00, 500.00, 3000.00)
SQL20264N  For table "DB2INST1.LE_EMPLOYEE", authorization ID "YANG"
does not have "WRITE" access to the column "BONUS".  SQLSTATE=42512
```

Example 5-17 shows that the same insert can be done by Ally, who has the credentials.

Example 5-17 Successful insert of the data into LBAC secured column

```
CONNECT TO sample USER Ally

INSERT INTO db2inst1.le_employee VALUES ('011090', 'JOHN', '',
'POWELL', 'CI5', '1007', '08/11/1997', 'SALES', 17, 'M', '08/11/1957',
101230.00, 500.00, 3000.00)
DB20000I  The SQL command completed successfully.
```

Deleting data in protected columns

Deleting data from the LBAC protected tables requires the same credentials as the update. Read access is required because without it, the rows are not visible and thus cannot be deleted. To delete the rows, write access is required.

Note: The LBAC credentials do not prevent users from dropping entire tables or databases that contain protected data. This is controlled by the DROPIN schema privilege outside of the LBAC.

Example 5-18 shows that an attempt to delete a row that includes LBAC protected columns fails because the user Saul has read access only.

Example 5-18 Error while deleting a row because Saul does not have write access

```
CONNECT TO sample USER db2sec;
GRANT SECURITY LABEL dataAccess.secretSecLabel TO USER Saul FOR READ
ACCESS;

CONNECT TO sample USER Saul;

DELETE FROM db2inst1.le_employee WHERE EMPNO='012201';
SQL20264N  For table "DB2INST1.LE_EMPLOYEE", authorization ID "SAUL"
does not have "WRITE" access to the column "SALARY".  SQLSTATE=42512
```

Example 5-19 shows the same delete statement executed by Ally, who has read and write access to the row.

Example 5-19 Successful delete of a row with LBAC column protection

```
CONNECT TO sample USER Ally;

DELETE FROM db2inst1.le_employee WHERE EMPNO='012201';
DB20000I  The SQL command completed successfully.
```

Updating data in protected columns

Updating the LBAC protected data requires that the user has the write access to the data and the UPDATE privilege on the table. Example 5-20 shows that user Mark tries to update this column and gets an error showing that access is denied.

Example 5-20 Update attempt when user cannot access the column

```
CONNECT TO sample USER mark
UPDATE db2inst1.le_employee SET SALARY=122000 WHERE EMPNO='010010'
SQL20264N  For table "DB2INST1.LE_EMPLOYEE", authorization ID "MARK"
does not have "READ" access to the column "SALARY".  SQLSTATE=42512
```

Example 5-21 shows that once Mark is granted read and write access on security label topSecretSecLabel, Mark can perform the update.

Example 5-21 Successful update of the LBAC protected column

```
CONNECT TO sample USER db2sec
GRANT SECURITY LABEL dataAccess.secretSecLabel TO USER Mark for ALL
ACCESS

CONNECT TO sample USER Mark
UPDATE db2inst1.le_employee SET SALARY=122000 WHERE EMPNO='010010'
DB20000I  The SQL command completed successfully.
```

Row protection

In order to add data protection to a table at the row level, you must add a column of type DB2SECURITYLABEL to the table. Each table can have at most one column of this type.

Note: An existing column cannot be altered to become a DB2SECURITYLABEL type. The only way of creating this type of column is adding a new one.

New table

Use the CREATE TABLE statement to define a new table and specify the SECURITY POLICY clause to assign a security policy to the table and then add a DB2SECURITYLABEL column. Example 5-22 creates a table with a security policy and adds the DB2SECURITYLABEL column for row security.

Example 5-22 Creating a table with row security enabled

```
CREATE TABLE db2inst1.le_pensions
(
  empno CHAR(6) NOT NULL,
  firstnme VARCHAR(12) NOT NULL,
  midinit CHAR(1),
  lastname VARCHAR(15) NOT NULL,
  hiredate DATE,
  job CHAR(8),
  birthdate DATE,
  base DECIMAL(9,2),
  avc DECIMAL(9,2),
  rowSec DB2SECURITYLABEL
)
SECURITY POLICY dataAccess
```

Existing table

An existing table can be secured using the ALTER TABLE statement. You need to add a security policy to the table using the ADD SECURITY POLICY clause and then add a column of the DB2SECURITYLABEL type.

Example 5-23 alters an existing table by adding a security policy to the table and secures the table at the row level by adding a DB2SECURITYLABEL column.

Example 5-23 Adding row security to an existing table

```
ALTER TABLE db2inst1.le_pensions
  ADD COLUMN rowSec DB2SECURITYLABEL
```

Note: When you alter a table and add a column of type DB2SECURITYLABEL, DB2 needs to assign a security label to existing rows in that table. Thus, DB2 requires the user doing this type of alter to hold a security label for write access (or for all access) from the security policy associated with the table. DB2 assigns this security label to existing rows in the table. Holding only SECADM is not sufficient.

Reading data in protected rows

Example 5-24 shows accessing a table by user Ally who holds the SECRET security label. All 15 SECRET rows are displayed. There is no CONFIDENTIAL or PUBLIC row present, Ally cannot access the TOP SECRET row.

Example 5-24 Retrieving rows for user Ally holding the SECRET security label

```
CONNECT TO sample USER Ally

SELECT firstnme, lastname, CHAR(SECLABEL_TO_CHAR('DATAACCESS', rowSec),
20) rowSec FROM db2inst1.le_pensions

FIRSTNME     LASTNAME        ROWSEC
------------ --------------- --------------------
MARK         SCUFF           SECRET
ALLY         AJAY            SECRET
SAUL         MASCUS          SECRET
ADAM         ANT             SECRET
DEBS         KING            SECRET
MARY         CONTRARY        SECRET
PETE         SOUTS           SECRET
YANG         DOODLE          SECRET
ANNE         BODDY           SECRET
ROSE         BUSH            SECRET
STAN         DELIVER         SECRET
ALAN         OMAHONY         SECRET
LORI         VERLOAD         SECRET
EVAN         WELSH           SECRET

  14 record(s) selected.
```

Example 5-25 shows the output for a user who does not have the required security label to read the secured data at all. The user receives no records and no error because all the rows are secured by a security label.

Example 5-25 Output of a query for a user without any security label

```
CONNECT TO sample USER db2sec

SELECT firstnme, lastname, CHAR(SECLABEL_TO_CHAR('DATAACCESS', rowSec),
20) rowSec FROM db2inst1.le_pensions

FIRSTNME     LASTNAME        ROWSEC
------------ --------------- --------------------

  0 record(s) selected.
```

Inserting data in protected rows

When inserting a new row into a table with protected rows, you do not have to provide a value for the column that is of type DB2SECURITYLABEL. If the value is not provided, the column is automatically populated with the *default* security label, which is the security label of the current user for *write* access. There is no possibility to explicitly specify a different default security label for the column. By using built-in functions like SECLABEL or SECLABEL_BY_NAME, the security label inserted in a column of type DB2SECURITYLABEL can be changed at the insert or update time.

Example 5-26 shows an attempt to insert a row with an explicitly specified security label when this user is not authorized to use it.

Example 5-26 Attempt to insert a row where given security label does not allow it

```
CONNECT TO sample USER Ally

INSERT INTO db2inst1.le_pensions VALUES ('018020', 'PAUL', '',
'THOMPSON', '09/29/2002', 'MANAGER', '02/13/1952', 521000.00, 11847.00,
SECLABEL_BY_NAME('DATAACCESS', 'TOPSECRETSECLABEL'))
SQL20402N  Authorization ID "ALLY" does not have the LBAC credentials
to perform the "INSERT" operation on table "DB2INST1.LE_PENSIONS".
SQLSTATE=42519
```

In order to write a different security label to a row, the user inserting the data must have been granted an exemption on a LBAC rule. In Example 5-27, the security administrator grants a write up exemption to Ally, allowing her to insert a row with a higher security label than the user currently has.

Example 5-27 Granting a write exemption to succeed with the insert

```
CONNECT TO sample USER db2sec

GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP FOR dataAccess TO
USER Ally

CONNECT TO sample USER Ally

INSERT INTO db2inst1.le_pensions VALUES ('018020', 'PAUL', '',
'THOMPSON', '09/29/2002', 'MANAGER', '02/13/1952', 521000.00, 11847.00,
SECLABEL_BY_NAME('DATAACCESS', 'TOPSECRETSECLABEL'))

DB20000I  The SQL command completed successfully.
```

Example 5-28 shows the output for user Mark who holds TOP SECRET label and is allowed to read everything. The output contains all 14 SECRET rows plus one TOP SECRET row.

Example 5-28 All rows listed as user Mark holds the highest security label

```
CONNECT TO sample USER Mark

SELECT firstnme, lastname, CHAR(SECLABEL_TO_CHAR('DATAACCESS', rowSec),
20) rowSec FROM db2inst1.le_pensions

FIRSTNME     LASTNAME        ROWSEC
------------ --------------- --------------------
MARK         SCUFF           SECRET
ALLY         AJAY            SECRET
SAUL         MASCUS          SECRET
ADAM         ANT             SECRET
DEBS         KING            SECRET
MARY         CONTRARY        SECRET
PETE         SOUTS           SECRET
YANG         DOODLE          SECRET
ANNE         BODDY           SECRET
ROSE         BUSH            SECRET
STAN         DELIVER         SECRET
ALAN         OMAHONY         SECRET
```

```
LORI         VERLOAD         SECRET
EVAN         WELSH           SECRET
PAUL         THOMPSON        TOP SECRET

  15 record(s) selected.
```

Updating data in protected rows

In the case of updating a protected row, the LBAC credentials must allow both read and update access to the row. If the user's LBAC credentials do not allow the user to read a row, then it is as though that row does not exist and there is no way to update that row.

Example 5-29 grants SELECT and UPDATE privileges on the DB2INST1.LE_PENSIONS table, and the security label SECRET for read only to user Saul. This way user Saul is able to see rows protected by the SECRET security label.

Example 5-29 Granting SECRET security label to Saul for read only, select then works

```
CONNECT TO sample USER db2adm;
GRANT SELECT, DELETE, UPDATE ON db2inst1.le_pensions TO Saul;

CONNECT TO sample USER db2sec;
GRANT SECURITY LABEL dataAccess.secretSecLabel TO USER Saul FOR READ
ACCESS;

CONNECT TO sample USER Saul;
SELECT empno, firstnme, midinit, lastname FROM db2inst1.le_pensions
WHERE empno='012201';

EMPNO  FIRSTNME     MIDINIT LASTNAME
------ ------------ ------- ---------------
012201 ANNE         E       BODDY

  1 record(s) selected.
```

Example 5-30 shows an error message when Saul tries to update a row secured by the SECRET security label. This fails because Saul has read only access to the row.

Example 5-30 Attempt for unauthorized update of a row

```
CONNECT TO sample USER Saul
UPDATE db2inst1.le_pensions SET MIDINIT='F' WHERE EMPNO='012201'
SQL20402N  Authorization ID "SAUL" does not have the LBAC credentials
to perform the "UPDATE" operation on table "DB2INST1.LE_PENSIONS".
SQLSTATE=42519
```

Example 5-31 shows the same update done by Ally, who has read and write access to the rows protected by the SECRET security label.

Example 5-31 Row update by Ally who has read and write access

```
CONNECT TO sample USER Ally;

UPDATE db2inst1.le_pensions SET MIDINIT='F' WHERE EMPNO='012201';
DB20000I  The SQL command completed successfully.

SELECT empno, firstnme, midinit, lastname FROM db2inst1.le_pensions
WHERE empno='012201';

EMPNO  FIRSTNME     MIDINIT LASTNAME
------ ------------ ------- ---------------
012201 ANNE         F       BODDY

  1 record(s) selected.
```

Deleting data in protected rows

The security labels do not allow the user to read rows when a user does not have LBAC credentials for read. The rows appear as though they do not exist. This behavior implies that it is not possible to delete rows that cannot be read. In order to delete a row that is readable, the user's LBAC credentials must allow the user to write to the row. To delete any row in a table that has protected columns, the user deleting the row must have LBAC credentials that allow her to write to *all* protected columns in the table.

Example 5-32 shows that user Saul, who has only read access, tried to delete the data and the delete action failed.

Example 5-32 Error while deleting a row protected by a security label

```
CONNECT TO sample USER Saul;

DELETE FROM db2inst1.le_pensions WHERE EMPNO='011111';
SQL20402N  Authorization ID "SAUL" does not have the LBAC credentials
to perform the "DELETE" operation on table "DB2INST1.LE_PENSIONS".
SQLSTATE=42519
```

Example 5-33 shows successful delete statement executed by Ally, who has read and write access to the row.

Example 5-33 Successful delete of a row with LBAC row protection

```
CONNECT TO sample USER Ally
DELETE FROM db2inst1.le_pensions WHERE EMPNO='011111'
DB20000I  The SQL command completed successfully.
```

Protecting both column and row

You can add the LBAC row as well as column protection to a table at the same time. Such a secured table must have been assigned a security policy, have one DB2SECURITYLABEL column defined, and then one or more columns secured by a security label.

New table

Example 5-34 shows creating a new table with both row and column security enabled.

Example 5-34 Creating a new table with both rows and columns secured

```
CREATE TABLE db2inst1.le_sales
(
  sales_date DATE,
  sales_person VARCHAR(15),
  region VARCHAR(15) SECURED WITH confSecLabel,
  sales INTEGER,
  rowSec DB2SECURITYLABEL
)
SECURITY POLICY dataAccess
```

Existing table

Example 5-35 shows altering an existing table to add security at the row level and the column level.

Example 5-35 Securing an existing table at the row and column levels

```
ALTER TABLE db2inst1.le_sales
  ADD SECURITY POLICY dataAccess
  ALTER COLUMN region SECURED WITH confSecLabel
ADD COLUMN rowSec DB2SECURITYLABEL
```

Accessing data in protected rows and columns

In order to access data in a table that is secured at the row and column levels, a user needs to have sufficient LBAC credentials that allow him to access both protected rows and columns, as described in "Column protection" on page 98 and "Row protection" on page 102.

Removing LBAC protection

The data secured by the LBAC can be made unprotected by removing the security labels from the rows or columns or simply by removing the security policy itself.

Note: The security policy can be removed only by the security administrator (SECADM). Because the removal is done using the ALTER statement, the SECADM must hold the *alter* privilege on the table.

Removing row protection

The data at the row level is protected by a security label that is stored in a column of the DB2SECURITYLABEL type. You *cannot* alter or drop a column of type DB2SECURITYLABEL from a table when this table is still associated with a security policy. An attempt to drop such a column results in error SQL0196N, as shown in Example 5-36.

Example 5-36 Error while trying to drop a DB2SECURITYLABEL column

```
ALTER TABLE db2inst1.le_sales DROP COLUMN rowSec

SQL0196N  Column "ROWSEC" in "LE_SALES" cannot be dropped. Reason code
= "3".
```

The recommended steps for this problem from SQL0196N RC=3 are to drop and recreate the table without creating the column of type DB2SECURITYLABEL. This approach would involve exporting and importing data if the table is not empty. An easier way to do the same but without the need to export and import data is to utilize the DROP SECURITY POLICY behavior as described in "Removing table protection" on page 111.

To drop the DB2SECURITYLABEL column, we recommend these steps:

1. Remove the security policy from the table.
2. Drop the DB2SECURITYLABEL column.
3. Add the security policy back to the table.
4. Secure all the columns as they were originally.

Example 5-37 shows the steps for removing column *rowSec* of type DB2SECURITYLABEL from the table DB2INST1.LE_SALES created in Example 5-34 on page 109.

Example 5-37 Removing a DB2SECURITYLABEL column from an existing table

```
CONNECT TO sample USER db2sec;

ALTER TABLE db2inst1.le_sales DROP SECURITY POLICY;

ALTER TABLE db2inst1.le_sales DROP COLUMN rowSec;

ALTER TABLE db2inst1.le_sales ADD SECURITY POLICY dataAccess;

CONNECT TO sample USER Yang

ALTER TABLE db2inst1.le_sales
  ALTER COLUMN region SECURED WITH confSecLabel;
```

Removing column protection

To remove column protection, simply alter the table definition and remove the column security using the DROP COLUMN SECURITY clause, as shown in Example 5-38.

Example 5-38 Removing LBAC column protection

```
ALTER TABLE db2inst1.le_pensions3 ALTER COLUMN base DROP COLUMN
SECURITY
```

Removing table protection

The data protection can be removed from a table by altering its definition and specifying the DROP SECURITY POLICY clause. Removing the security policy makes the table completely unprotected. It removes all the LBAC protection from the table (row and column protection).

If the table has a column with data type DB2SECURITYLABEL, the data type is changed to VARCHAR (128) FOR BIT DATA. If the table has one or more protected columns, those columns become unprotected.

Example 5-39 shows removing LBAC security from a table.

Example 5-39 Removing LBAC security from a table

```
ALTER TABLE db2inst1.le_sales DROP SECURITY POLICY
```

5.5.5 Maintenance and administration

In this section we demonstrate how to determine whether a column, a row, and a table are LBAC protected. We also discuss the built-in functions for managing LBAC security labels.

Determining if columns, rows, and tables are LBAC protected

There may be situations when you want to see whether a table, row, or column is protected, and if so, what security label is used to protect the data.

Columns

The information about which column in a table is LBAC protected is stored in the system catalog as metadata. It cannot be displayed using the DESCRIBE TABLE command.

To obtain this information, query the SYSCAT.COLUMNS catalog view as demonstrated in Example 5-40. All the columns with *enabled* security have a non-null security label in the SECLABELNAME column.

Example 5-40 Query the system catalog to show what columns are LBAC protected

```
CONNECT TO sample USER Mark;

SELECT COLNAME, SECLABELNAME FROM SYSCAT.COLUMNS WHERE
TABSCHEMA='DB2INST1' AND TABNAME='LE_EMPLOYEE';

COLNAME    SECLABELNAME
---------- --------------
BIRTHDATE  -
BONUS      SECRETSECLABEL
COMM       -
EDLEVEL    -
EMPNO      -
FIRSTNME   -
HIREDATE   -
```

```
JOB          -
LASTNAME     -
MIDINIT      -
PHONENO      -
SALARY       SECRETSECLABEL
SEX          -
WORKDEPT     -

  14 record(s) selected.
```

Rows

Row-level LBAC security requires a column of the DB2SECURITYLABEL to be present in a table. This information can be obtained using the DESCRIBE TABLE command, as shown in Example 5-41.

Example 5-41 Obtaining description of a table to find the row LBAC security

```
DESCRIBE TABLE db2inst1.le_pensions

Column name Data type Data type name   Column Scale Nulls
            schema                     Length
----------- --------- ---------------- ------ ----- -----
EMPNO       SYSIBM    CHARACTER             6     0 No
FIRSTNME    SYSIBM    VARCHAR              12     0 No
MIDINIT     SYSIBM    CHARACTER             1     0 Yes
LASTNAME    SYSIBM    VARCHAR              15     0 No
HIREDATE    SYSIBM    DATE                  4     0 Yes
JOB         SYSIBM    CHARACTER             8     0 Yes
BIRTHDATE   SYSIBM    DATE                  4     0 Yes
BASE        SYSIBM    DECIMAL               9     2 Yes
AVC         SYSIBM    DECIMAL               9     2 Yes
ROWSEC      SYSPROC   DB2SECURITYLABEL      0     0 No

  10 record(s) selected.
```

Table

There is a possibility that a security policy is assigned to a table but no row or column is actually secured. For this case, or just to see which security policy is active for a table, query the system catalog using the statement from Example 5-42.

Example 5-42 Obtaining security policy assigned to a table

```
SELECT A.SECPOLICYNAME FROM SYSCAT.SECURITYPOLICIES A, SYSCAT.TABLES B
WHERE A.SECPOLICYID=B.SECPOLICYID AND B.TABSCHEMA='DB2INST1' AND
B.TABNAME='LE_PENSIONS'

SECPOLICYNAME
------------------------------------------------
DATAACCESS

  1 record(s) selected.
```

Built-in functions for managing LBAC security labels

DB2 comes with built-in functions SECLABEL, SECLABEL_BY_NAME, and SECLABEL_TO_CHAR for managing the LBAC security labels. These functions are useful if we want to insert non-default security labels to the rows.

SECLABEL

This function builds a security label by specifying a security policy and values for each of the components in the label. The returned value has a data type of DB2SECURITYLABEL. Figure 5-10 shows the syntax of the SECLABEL function.

```
>>-SECLABEL--(--security-policy-name--,--security-label-string--)-><
```

Figure 5-10 Syntax of the SECLABEL function

Example 5-43 shows use of the SECLABEL function.

Example 5-43 Creating a security label using SECLABEL function

```
INSERT INTO db2inst1.le_pensions VALUES ('018020', 'PAUL', '',
'THOMPSON', '09/29/2002', 'MANAGER', '02/13/1952', 521000.00, 11847.00,
SECLABEL('DATAACCESS', 'TOP SECRET'))
```

SECLABEL_BY_NAME

This function builds a security label by specifying a security policy and existing security label. The returned value has a data type of DB2SECURITYLABEL. This function must be used when inserting an existing security label into a column. Figure 5-11 shows the syntax of the SECLABEL_BY_NAME function.

```
>>-SECLABEL_BY_NAME--(--security-policy-name--,--security-label-name--)-><
```

Figure 5-11 Syntax of the SECLABEL_BY_NAME function.

Example 5-44 shows using the SECLABEL_BY_NAME function.

Example 5-44 Creating a security label using the SECLABEL_BY_NAME function

```
INSERT INTO db2inst1.le_pensions VALUES ('018020', 'PAUL', '',
'THOMPSON', '09/29/2002', 'MANAGER', '02/13/1952', 521000.00, 11847.00,
SECLABEL_BY_NAME('DATAACCESS', 'TOPSECRETSECLABEL'))
```

SECLABEL_TO_CHAR

This function returns a string representation of values that make up a security label. This is useful when we want to display user-friendly output of the DB2SECURITYLABEL columns.

Figure 5-12 shows the syntax of the SECLABEL_TO_CHAR function.

```
>>-SECLABEL_TO_CHAR-(--security-policy-name--,--security-label--)-><
```

Figure 5-12 Syntax of the SECLABEL_TO_CHAR function

Example 5-45 shows using the SECLABEL_TO_CHAR function.

Example 5-45 Using the SECLABEL_TO_CHAR function to display security labels

```
CONNECT TO sample USER Mark

SELECT firstnme, lastname, CHAR(SECLABEL_TO_CHAR('DATAACCESS', rowSec),
20) rowSec FROM db2inst1.le_pensions

FIRSTNME     LASTNAME        ROWSEC
------------ --------------- --------------------
MARK         SCUFF           TOP SECRET
ALLY         AJAY            SECRET
SAUL         MASCUS          SECRET
ADAM         ANT             SECRET
DEBS         KING            SECRET
MARY         CONTRARY        SECRET
PAUL         THOMPSON        TOP SECRET
YANG         DOODLE          SECRET
```

```
ROSE        BUSH           TOP SECRET
STAN        DELIVER        SECRET
ALAN        OMAHONY        SECRET
LORI        VERLOAD        SECRET
EVAN        WELSH          SECRET
JOHN        POWELL         SECRET

  14 record(s) selected.
```

6

Auditing

Auditing is one of the aspects of information security. It monitors data access and provides information needed for subsequent analysis. Auditing can help discover unwanted, unknown, and unacceptable access to the data as well as keep history records of the activities on the database system.

DB2 provides an audit facility that generates, and allows maintenance of, an audit trail for a series of predefined database events. It can audit at both the instance and the individual database level, independently recording all instance and database level activities.

In this chapter we discuss the following topics:

- New db2audit features introduced in DB2 9.5
- Audit policies and audit statement
- Setup and configuration of the DB2 audit facility
- Working with the audit data
- Maintenance and administration tasks associated with audit

6.1 db2audit

The DB2 provides an audit facility to assist in the detection of unknown or unanticipated access to data. The DB2 audit facility generates and permits the maintenance of an audit trail for a series of predefined database events. The records generated from this facility are kept in audit log files. The analysis of these records can reveal usage patterns that would identify system misuse. Once identified, actions can be taken to reduce or eliminate such system misuse.

6.1.1 Concept

db2audit is the DB2 audit facility administrator tool used to configure audit at the *instance* level, as well as control when such audit information is collected. The tool also can be used to archive both *instance* and *database* audit logs and to extract from archived logs of either type.

db2audit can be used only by the system administrators (users having the SYSADM authority) to control the following actions:

- Start recording auditable events within the DB2 instance. This does not include database-level activities.
- Stop recording auditable events within the DB2 instance.
- Configure the behavior of the audit facility at the instance level only.
- Select the categories of the auditable events to be recorded at the instance level only.
- Request a description of the current audit configuration for the instance.
- Flush any pending audit records from the instance and write them to the audit log.
- Archive audit records from the current audit log for either the instance or a database under the instance.
- Extract audit records from an archived audit log by formatting and copying them to a flat file or an ASCII delimited file. Extraction is done in preparation for analysis of log records.

6.1.2 Changes for DB2 9.5

The changes introduced in the DB2 9.5 audit facility enable the use of DB2 auditing on production database systems by increasing usability, decreasing the performance impact, and enhancing the information captured to match what is required to demonstrate security compliance.

Instance and database level auditing

The audit facility in DB2 9.5 separates the instance and database audit logs. Auditing for each database now can be configured separately along with the instance itself.

Audit log archives

In DB2 9.5, the audit facility introduces *audit log archives* designed to be used for archiving active audit logs to a new location for future extraction. When extracting data from the logs, the **extract** command now only accepts an archived log, not the active log. This results in better overall system performance because an instance does not hang until the extract operation is completed, as in previous versions.

Audit data location

The default location of the audit log files is the security/auditdata subdirectory of the current instance. We discuss more audit files location and naming conventions in 6.4.1, "db2audit files" on page 126.

Audit categories

DB2 9.5 enables you to configure the auditing at a finer level, allowing you to specify exactly what to audit. In previous versions, it was only possible to enable or disable audit for all categories at once. Now in DB2 9.5, any audit category can be enabled or disabled separately.

Changes in the db2audit commands

db2audit commands have been enhanced as follows:

- **prune** command

  The **prune** command has been removed. Its functionality has been replaced by the **archive** command. The **archive** command moves the active audit log to a new location, effectively pruning all non-active records from the active log. All records that are currently being written to the audit log will be completed before the log is archived to ensure that full records are not split apart. All records that are created while the archive is in progress will be written to the current audit log, not the archived log, once the archive has finished.

- **extract** command

  The **extract** command now works only with archived log files. It is no longer possible to extract from the active log. It also provides an option to choose what audit categories to extract and whether to extract success or failure events (or both).

- **`configure`** command

  The **`configure`** command can be used to specify success, failure, or both events for any of the audit categories separately. Additionally, the settings are cumulative. Only the categories specified are modified. All the others remain unchanged. The command also introduces DATAPATH and ARCHIVEPATH keywords for setting the actual location of the audit active and archived logs.

- **`describe`** command

  The output of the **`describe`** command has changed to support the new syntax of the **`configure`** command. It shows status SUCCESS, FAILURE, NONE, or BOTH for each audit category. In addition, it shows the current audit data and archive paths.

- **`start`** and **`stop`** commands

  These two commands affect only instance-level auditing. To control auditing at the database level use the AUDIT SQL statement discussed in 6.3, “Audit statement” on page 123.

6.2 Audit policies

An audit policy is a database object that holds an instance of the audit configuration. The policy determines what categories are to be audited. It can then be applied to other database objects to determine how the use of those objects is to be audited. DB2 9.5 audit policies provide a finer granularity of control on what needs to be audited compared to previous DB2 versions, where only instance-level auditing was available.

The advantages of using the audit policies includes:

- Smaller audit trails
- Smaller performance overhead

The audit policy can be defined by the CREATE AUDIT POLICY statement, modified by ALTER AUDIT POLICY statement, and deleted by the DROP AUDIT POLICY statement.

CREATE AUDIT POLICY statement

Figure 6-1 shows the syntax of the CREATE AUDIT PLOICY statement.

```
>>-CREATE AUDIT POLICY--policy-name--.--CATEGORIES-------------->

   .-,-----------------------------------------------------.
   V  (1)                                                  |
>----------+-ALL-----------------------+--STATUS--+-BOTH----+-+-->
           +-AUDIT---------------------+          +-FAILURE-+
           +-CHECKING------------------+          +-NONE----+
           +-CONTEXT-------------------+          '-SUCCESS-'
           |          .-WITHOUT DATA-. |
           +-EXECUTE--+--------------+-+
           |          '-WITH DATA----' |
           +-OBJMAINT------------------+
           +-SECMAINT------------------+
           +-SYSADMIN------------------+
           '-VALIDATE------------------'

>--?--ERROR TYPE--+-NORMAL-+--.--------------------------------><
                  '-AUDIT--'
```

Figure 6-1 Syntax of CREATE AUDIT POLICY statement

Example 6-1 shows creating, altering, and dropping audit policies.

Example 6-1 Creating, altering, and dropping audit policies

```
CREATE AUDIT POLICY auditdb CATEGORIES ALL STATUS BOTH ERROR TYPE
NORMAL;

CREATE AUDIT POLICY admins CATEGORIES SYSADMIN STATUS BOTH ERROR TYPE
NORMAL;

CREATE AUDIT POLICY powerusers CATEGORIES OBJMAINT STATUS BOTH,
SECMAINT STATUS BOTH ERROR TYPE AUDIT;

CREATE AUDIT POLICY context CATEGORIES CONTEXT STATUS BOTH ERROR TYPE
NORMAL;

ALTER AUDIT POLICY context CATEGORIES CONTEXT STATUS FAILURE ERROR TYPE
NORMAL;

ALTER AUDIT POLICY auditdb CATEGORIES EXECUTE WITH DATA STATUS BOTH
ERROR TYPE NORMAL;

DROP AUDIT POLICY context;
```

6.2.1 Audit categories

Audit categories are event types that can be independently audited. These categories can be specified when creating audit policies for database-level auditing or enabling auditing at the instance level using the `db2audit` command.

Starting from DB2 9.5, all the database event categories now contain local and global transaction IDs. In addition to that, the following special registers are also captured:

- CLIENT_USERID
- CLIENT_ACCTNG
- CLIENT_WORKSTNNAME
- CLIENT_APPLNAME

Note: By default, these special registers are set to an empty string on the client. This means that if the application does not set it explicitly then the audit record contains an empty string (in ASCII delimited file format) or it will not be listed at all (in text file format).

The audit categories are:

- AUDIT

  Generates records when audit settings are changed or when the audit log is accessed.

- CHECKING

  Generates records during authorization checking of attempts to access or manipulate database objects or functions.

- CONTEXT

  Generates records to show the operation context when a database operation is performed.

- OBJMAINT

  Generates records when data objects are created or dropped.

- SECMAINT

  Generates records when object privileges, database privileges, or DBADM authority is granted or revoked. Records are also generated when the database manager security configuration parameters SYSADM_GROUP, SYSCTRL_GROUP, or SYSMAINT_GROUP are modified.

- SYSADMIN

  Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.

- VALIDATE

  Generates records when users are authenticated or when system security information related to a user is retrieved.

- EXECUTE

  Generates records to show the execution of SQL statements and, optionally, shows the input data values provided to the statements.

  Execute is a new *database*-level-only audit category introduced in DB2 9.5. It replaces the CONTEXT category for SQL statement execution. This category provides sufficient information to replay statements to understand their effect.

  The EXECUTE category can provide:

  – Statement text
  – Data type, length, value of input host variables, and parameter markers (does not include LOBS, LONG, XML, and structured types)
  – Compilation environment
  – Counts indicating number of rows read, returned, and modified

  Note: When creating a policy, it is possible to choose all audit categories by specifying CREATE AUDIT POLICY CATEGORIES ALL ... . That sets all the audit categories to the same status, and the EXECUTE category is set to WITHOUT DATA by default.

6.3 Audit statement

The AUDIT SQL statement is used to configure and control the audit requirements for an individual *database*.

The security administrator (who holds SECADM authority at the database level) can use audit policies in conjunction with the AUDIT SQL statement to configure and control the audit requirements at the database level. The policies can be used to configure the audit system to gather information only about the data and objects that are needed.

The AUDIT statement determines the audit policy that is to be used for a particular database or database object at the current server. Whenever the object is in use, it is audited according to that policy.

- Many objects can be audited according to a single audit policy.
- Only one audit policy can be associated with any one object.

- Audit policies are combined in inclusive manner when multiple policies are in effect. For example, if there are two groups, the first audited using a policy with audit category OBJMAINT and the second audited using audit category SECMAINT, a user belonging to both groups is audited using combined policies, which means that both event types OBJMAINT and SECMAINT are recorded for this user.

The audit statements must be immediately followed by a COMMIT or ROLLBACK statement. They cannot be issued within distributed extended architecture (XA) transactions.

6.3.1 Auditable objects

An audit policy can be applied to the following database objects:

- Databases

  All auditable events that occur within the database are audited according to the audit policy. Example 6-2 defines the auditable policy to be used for database level audit.

  Example 6-2 Auditing the entire database

  ```
  AUDIT DATABASE USING POLICY auditdb
  ```

- Tables

  All data manipulation language (DML) and XQuery access to the tables is audited. The supported table types are:

  – Untyped tables
  – Materialized Query Tables (MQT)
  – Nicknames

  Example 6-3 shows how to associate an audit policy with a table.

  Example 6-3 Auditing a table

  ```
  AUDIT TABLE le_employee USING POLICY auditdb
  ```

  An audit policy cannot be associated with a view, a catalog table, or a declared temporary table. Views that access a table that has an associated audit policy are audited according to the underlying table's policy.

  Only EXECUTE category audit events with or without data are generated when the table is accessed even if the policy indicates that other categories should be audited.

- Trusted contexts

  All auditable events that happen within a trusted connection defined by the particular trusted context are audited according to the audit policy. Example 6-4 associates audit policy admins with trusted context appserver.

  Example 6-4 Auditing a context

  ```
  AUDIT TRUSTED CONTEXT appserver USING POLICY admins
  ```

- Users, groups, or roles

  All auditable events that are initiated by the specified user are audited according to the audit policy.

  All auditable events that are initiated by users that are a member of the group or role are audited according to the audit policy. Indirect role membership, such as through other roles or groups, is also included. Example 6-5 associates an audit policy admins with role admin.

  Example 6-5 Auditing a role

  ```
  AUDIT ROLE admin USING POLICY admins
  ```

- Authorities

  All auditable events that are initiated by a user that holds authority SYSADM, SECADM, DBADM, SYSCTRL, SYSMAINT, or SYSMON, even if that authority is unnecessary for the event, are audited according to the audit policy. If an audit policy is associated with DBADM authority, any user with SYSADM authority is also audited according to this policy, because he is considered to have DBADM authority.

  Example 6-6 defines an audit policy powerusers to audit DB2 authorities.

  Example 6-6 Auditing a authorities

  ```
  AUDIT SYSADM, DBADM, SECADM, GROUP DBAS USING POLICY powerusers
  ```

For a specific object, there can only be one audit policy in effect. For example, there cannot be multiple audit policies associated with the same table at the same time.

6.4 Setup and configuration

The DB2 audit facility can be configured at the instance level and database level. The system administrator (SYSADM) controls instance auditing using the db2audit tool. The security administrator (SECADM) controls database auditing using the AUDIT SQL statement.

6.4.1 db2audit files

There are two types of DB2 audit log files, active logs and archived logs. db2audit stores its configuration in the db2audit.cfg file located in the security subdirectory of the current instance. By default, the collected audit data is stored in several log files located in the security/auditdata subdirectory of the current instance.

Note: Starting from DB2 9.5, the location of the instance directories on Windows is no longer under the SQLLIB directory. The new location is in Documents and Settings\All Users\Application Data\IBM\DB2\<db2_copy_name> on the Windows XP and Windows 2003 operating systems and ProgramData\IBM\DB2\<db2_copy_name> on the Windows Vista® operating system.

Active logs

The active log file names follow these naming conventions:

- The instance audit log:

  `db2audit.instance.log.node_number`

- The database audit log:

  `db2audit.db.dbname.log.node_number`

In a non-DPF environment, node_number is always 0.

Example 6-7shows examples of the db2audit log file names generated by the db2audit for the SAMPLE database.

Example 6-7 db2audit log file names

```
Directory of C:\Documents and Settings\All Users\Application
Data\IBM\DB2\VIPER 2\DB2\security\auditdata

09/26/2007  06:10 PM    <DIR>          .
09/26/2007  06:10 PM    <DIR>          ..
09/26/2007  03:49 PM          11,202 db2audit.db.SAMPLE.log.0
```

```
09/26/2007  06:10 PM          8,594 db2audit.instance.log.0
09/26/2007  06:09 PM         35,548
db2audit.instance.log.0.20070926180937
               3 File(s)        33,344 bytes
```

Changing log file location

To change the location of the active log file, use the following command:

```
db2audit configure datapath <log_path>
```

The actual data path directory either can be a fixed path or can contain database partition expressions, which are necessary in a cluster environment for High Availability (HA) failover. A database partition expression must start with the blank character (space) followed by the $N prefix and optionally can contain arithmetic operators + (plus) or % (modulus) and integer numbers.

Table 6-1 lists all possible combinations of the arithmetic operators and numbers in the database partition expressions (the examples assume the current node number is 23).

Table 6-1 All supported database partition expression formats

Syntax	Example	Result
[blank]$N	" $N"	"23"
[blank]$N+[number]	" $N+1000"	"1023"
[blank]$N%[number]	" $N%4"	"3"
[blank]$N+[number]%[number]	" $N+19%8	"2"
[blank]$N%[number]+[number]	" $N%3+20"	"22"

The data path directory can contain none, one, or more than one database partition expression. The following example shows using two database partititon expressions in one path at the same time:

```
db2audit configure datapath "D:\auditLogs $N+1 _ $N+1"
```

If we assume that the current database partition is 0, then the path where the active logs will be created is:

```
D:\auditLogs1_1”.
```

Archive logs

The archive audit logs, by default, are also located in the security/auditdata subdirectory. The archive logs follow the same file naming convention as the active logs, but they have appended the time stamp of *when* the **archive** command was run:

- The instance audit log:

  `db2audit.instance.log.node_number.timestamp`

- The database audit log:

  `db2audit.db.dbname.log.node_number.timestamp`

The timestamp format is *YYYYMMDDHHMMSS*, where *YYYY* is the year, *MM* is the month, *DD* is the day, *HH* is the hour, *MM* is the minute, and *SS* is the seconds. The time is the local time.

To change the location of the archive log files, use the following command:

```
db2audit configure archivepath <archive_path>
```

Example 6-8 shows setting the active and archive log paths.

Example 6-8 Changing the log file paths

```
D:\SQLLIB\BIN>db2audit configure datapath D:\auditLogs

AUD0000I  Operation succeeded.

D:\SQLLIB\BIN>db2audit configure archivepath D:\auditLogs\archive

AUD0000I  Operation succeeded.

D:\SQLLIB\BIN>db2audit describe
DB2 AUDIT SETTINGS:

Audit active: "TRUE "
Log audit events: "BOTH"
Log checking events: "BOTH"
Log object maintenance events: "BOTH"
Log security maintenance events: "BOTH"
Log system administrator events: "BOTH"
Log validate events: "BOTH"
Log context events: "BOTH"
Return SQLCA on audit error: "FALSE "
Audit Data Path: "D:\auditLogs\"
Audit Archive Path: "D:\auditLogs\archive\"

AUD0000I  Operation succeeded.
```

Note: When changing the audit log paths, both active and archive, it is required that the newly assigned directories already exist. The db2audit does not create a directory when it does not exist. An attempt to configure non-existing directory results in error AUD0003N:

```
AUD0003N I/O error on accessing "D:\nonExistingDirectory", make sure
the directory/file exists and has the right permission.
```

6.4.2 Instance

Auditing at the instance level is controlled by the db2audit tool. You can use the **`db2audit configure`** command to specify the scope of the instance level auditing, status and errors required in the audit data, and audit files location.

The scope of the auditing specifies what categories are audited. These are the same categories as described in 6.2.1, “Audit categories” on page 122:

- AUDIT
- CHECKING
- CONTEXT
- OBJMAINT
- SECMAINT
- SYSADMIN
- VALIDATE

Note: At the instance level (using db2audit tool), you cannot specify the EXECUTE category because it is for the SQL statements only.

The status specifies whether only successful or failing events, or both successful and failing events, should be logged. The status has the following options:

both	Successful and failing events will be audited.
none	No events for this category will be audited.
failure	Only failing events will be audited.
success	Only successful events will be audited.

Only the categories specified on the configure statement are modified. All other categories have their status preserved.

db2audit configure command

Figure 6-2 shows the syntax of the **`db2audit configure`** command.

```
|>- db2audit -+- configure -+- reset -----------------+----------+
                            '-( Audit Configuration )-'          |

Audit Configuration:
>--------+-----------------------------------------------------+->
         |         .-----,-----------------------------------. |
         |         V                                         | |
         +- scope ---+-- all -------+-- status -+- both ----+-+-+
                     +-- audit -----+           +- none ----+
                     +-- checking --+           +- failure -+
                     +-- context  --+           '- success -'
                     +-- objmaint --+
                     +-- secmaint --+
                     +-- sysadmin --+
                     '-- validate --'

>--+------------------------+--------------------------------->
   '- errortype -+- audit --+--'
                 '- normal -'

>--+-------------------------------+---------------------------->
   '- datapath--<audit-data-path>---'

>--+---------------------------------------+-------------------->
   '- archivepath--<audit-archive-path>-----'
```

Figure 6-2 db2audit configure

Example 6-9 shows how the **`db2audit configure`** command can be used for configuring the instance-level auditing.

Example 6-9 Configuring the instance-level auditing

```
db2audit configure scope objmaint status success errortype normal;

db2audit configure scope secmaint status both, sysadmin status both;
```

6.4.3 Database

Auditing at the database level is controlled by the AUDIT SQL statement. In order to issue the statement, a connection to a database needs to be established first. Any subsequent AUDIT statement then configures auditing for this database only.

AUDIT DATABASE statement

Figure 6-3 shows the syntax of the AUDIT DATABASE statement.

```
>>-AUDIT------------DATABASE------------------------------------>

>--+-+-USING---+--POLICY--policy-name-+------------------------><
   | '-REPLACE-'                      |
   '-REMOVE POLICY--------------------'
```

Figure 6-3 AUDIT DATABASE statement

6.4.4 Table

Auditing at the table level is controlled by the AUDIT SQL statement. A valid connection to the database, where such a table resides, must be established in order to issue the statement.

AUDIT TABLE statement

Figure 6-4 shows the syntax of AUDIT TABLE statement.

```
>>-AUDIT------------TABLE--table-name--------------------------->

>--+-+-USING---+--POLICY--policy-name-+------------------------><
   | '-REPLACE-'                      |
   '-REMOVE POLICY--------------------'
```

Figure 6-4 AUDIT TABLE statement syntax

6.4.5 User, role, and group

Auditing at the user, role, or group level is controlled by the AUDIT SQL statement as well. It requires a valid database connection in order to issue the AUDIT statement.

AUDIT USER/ROLE/GROUP statement

Figure 6-5 shows the syntax of the AUDIT USER/ROLE/GROUP statement.

```
>>-AUDIT------------+-USER--+--authorization-name--------------->
                    +-GROUP-+
                    '-ROLE--'

>--+-+-USING---+--POLICY--policy-name-+------------------------><
   | '-REPLACE-'                      |
   '-REMOVE POLICY--------------------'
```

Figure 6-5 AUDIT USER/ROLE/GROUP statement syntax

6.5 Audit data

The audit data is stored in log files. There are active and archive log files. The active log files contain all the records about operations currently in progress or records that have not been archived. When there is a need to back up active logs or work with audit data, an archive must be created.

6.5.1 Archive

The **`archive`** command of the db2audit tool allows you to move audit log records from the active log into the archive (backup) log. The archive logs are stored either in the default location, in the location specified by the **`db2audit configure`** command, or in the directory explicitly specified by the **`archive`** command.

db2audit archive command

Figure 6-6 shows the syntax of the **`db2audit archive`** command.

```
|>- db2audit -+- archive ---( Audit Archive )--------------------+-><|

Audit Archive:
>--+----------------------------+-+--------------------------------+->
   '- database -<database name>-' '- node -+-----------------------+
                                           '-<current node number>-'

>--+----------------------------+---------------------------------->
   '- to --<audit-archive-path>--'
```

Figure 6-6 Syntax of the archive db2audit command

Example 6-10 shows examples of using the **db2audit archive** command to archive the instance log and SAMPLE database log.

Example 6-10 Archiving the instance and database logs

```
D:\>db2audit archive

Node     AUD      Archived or Interim Log File
         Message
-------- -------- ---------------------------------------------------
       0 AUD0000I db2aduit.instance.log.0.20071001161758

AUD0000I  Operation succeeded.

D:\>db2audit archive database sample node 0 to d:\

Node     AUD      Archived or Interim Log File
         Message
-------- -------- ---------------------------------------------------
       0 AUD0000I db2audit.SAMPLE.log.0.20071001161830

AUD0000I  Operation succeeded.
```

When running the **db2audit archive** command against an instance or a database and there are no audit records to archive, the **db2audit audit** command returns AUD0000I Operation succeeded, even though no archive file is created. See Example 6-11.

Example 6-11 Running the archive command against database with no audit records

```
D:\>db2audit archive database sample node 0 to d:\

Node     AUD      Archived or Interim Log File
         Message
-------- -------- ---------------------------------------------------
       0 AUD0000I

AUD0000I  Operation succeeded.
```

6.5.2 Extract

The audit data stored in the audit log files can be extracted using the extract parameter of the **db2audit** command. You can only run the **db2audit extract** command against the archived log files, not the active logs. The data extracted from the archived log can be stored in a flat, easy-to-read text file or as delimited

ASCII format suitable for loading into DB2 tables. In all cases, the extract follows the rule that if the output file already exists, it is not replaced, but the new content is appended to the existing file.

The list of input audit log files that will be extracted is specified by the option *files*. This may be a single file or a list of files. The file names will be combined with the path to get the fully qualified file names if they are not already fully qualified. In such case, the path is either the default archive path specified in the audit configuration file or the explicit path specified by the *path* option. The list may include standard shell wild cards to specify multiple files.

db2audit extract command

Figure 6-7 shows the syntax of the **db2audit extract** command.

```
|>- db2audit -+- extract -( Audit Extraction )-------------------+

Audit Extraction:
>--+-----------------------------------------------------------+-->
   +-file--<output file>----------------------------------------+
   '-delasc--+--------------------------+-+-----------------+
             '-delimiter-<load delimiter>-' '-to-<delasc path>-'

>--+-------------------------------------------------------+---------->
   +-status--+-success-+-------------------------------+
   |         '-failure-'                               |
   |                                                   |
   |           .--,---------------------------------.  |
   |           V                                    |  |
   '-category--+-audit----+--+-------------------+-+--'
               +-checking-+  '-status-+-failure-+-'
               +-context--+           '-success-'
               +-execute--+
               +-objmaint-+
               +-secmaint-+
               +-sysadmin-+
               '-validate-'

>--from--+-----------------------+--files--<input log files>-----|
         '--path-<archive path>---'
```

Figure 6-7 db2audit extract

Text format of the extracted data

By default, when there is no delasc or file option specified, the db2audit writes the data to the db2audit.out file in the archive path specified in the audit configuration file. The location of the produced file and output file name can be overridden using the *file* option of the **db2audit extract** command. If this file option only specifies a file name and does not contain a path, the file is outputted to the current working directory.

The next three examples show all ways of exporting logs in text format.

Example 6-12 shows how to extract logs as text into the default output file.

Example 6-12 Extracting logs as text into default db2audit.out file

```
D:\>db2audit extract from files D:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.

D:\>db2audit describe
DB2 AUDIT SETTINGS:

Audit active: "TRUE "
Log audit events: "BOTH"
Log checking events: "BOTH"
Log object maintenance events: "BOTH"
Log security maintenance events: "BOTH"
Log system administrator events: "BOTH"
Log validate events: "BOTH"
Log context events: "BOTH"
Return SQLCA on audit error: "FALSE "
Audit Data Path: "D:\auditLogs $N+1 _ $N+1 \"
Audit Archive Path: "D:\auditLogs\archive\"

AUD0000I  Operation succeeded.

D:\>dir D:\auditLogs\archive

Directory of D:\auditLogs\archive

10/01/2007  06:59 PM             4,560 db2audit.out
               1 File(s)          4,560 bytes
```

Example 6-13 shows how to extract logs as text into the db2audit log file in the current directory.

Example 6-13 Extracting logs as text into db2audit.log_files file in the current directory

```
D:\>db2audit extract file db2audit.log_files from files
D:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.
```

```
D:\>dir *.log_files

 Directory of D:\

10/01/2007  07:07 PM             4,560 db2audit.log_files
               1 File(s)          4,560 bytes
```

Example 6-14 shows how to extract logs as text into a file in a particular directory.

Example 6-14 Extracting logs as text into db2audit.log_files file in D:\tmp directory

```
D:\>db2audit extract file D:\tmp\db2audit.log_files from files
d:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.

D:\>dir D:\tmp

 Directory of D:\tmp

10/01/2007  07:11 PM             4,560 db2audit.log_files
               1 File(s)          4,560 bytes
```

Example 6-15 shows one record for the EXECUTE audit category from an exported audit log in text format.

Example 6-15 Exported log in text format showing EXECUTE audit record

```
timestamp=2007-09-28-17.11.37.831000;
  category=EXECUTE;
  audit event=STATEMENT;
  event correlator=30;
  event status=0;
  database=SAMPLE;
  userid=db2admin;
  authid=DB2ADMIN;
  session authid=DB2ADMIN;
  origin node=0;
  coordinator node=0;
  application id=*LOCAL.DB2.070927223246;
  application name=db2bp.exe;
  package schema=DB2ADMIN;
  package name=SQLC2G13;
  package section=201;
  local transaction id=0x2013000000000000;
  global transaction id=0x0000000000000000000000000000000000000000;
  uow id=23;
  activity id=1;
  statement invocation id=0;
  statement nesting level=0;
  activity type=READ_DML;
  statement text=select * from department;
  statement isolation level=CS;
  Compilation Environment Description
    isolation: CS
    query optimization: 5
    min dec div 3: NO
    degree: 1
    SQL rules: DB2
    refresh age: +00000000000000.000000
    resolution timestamp: 2007-10-01-16.34.16.000000
    federated asynchrony: 0
    schema: DB2ADMIN
    maintained table type: SYSTEM;
  rows modified=0;
  rows returned=14;
```

Example 6-16 shows a CONTEXT record from the exported audit log file where trusted context switching is logged.

Example 6-16 Exported log in text format showing CONTEXT audit record

```
timestamp=2007-10-11-11.42.20.882000;
  category=CONTEXT;
  audit event=SWITCH_USER;
  event correlator=6;
  database=SAMPLE;
  userid=Yang;
  authid=YANG;
  origin node=0;
  coordinator node=0;
  application id=127.0.0.1.46607.071011184217;
  application name=db2cli.exe;
  trusted context name=PAYROLLTRUSTEDCONTEXT;
  connection trust type=EXPLICIT_TRUSTED_CONNECTION;
  role inherited=PAYROLL;
```

ASCII delimited format of the exported data

The extracted audit records can be placed in a delimited ASCII format suitable for loading into DB2 tables. To do this, use the *delasc* option of the **export** command. The output is placed in separate files, one for each audit category. In addition, the file auditlobs is also created to hold any LOBs that are included in the audit data.

The ASCII delimited format file names are:

- audit.del
- checking.del
- objmaint.del
- secmaint.del
- sysadmin.del
- validate.del
- context.del
- execute.del
- auditlobs

The auditlobs file is created if the context or execute categories are extracted. LOB Location Specifiers are included in the *.del files to reference the LOBS in the auditlobs file.

By default the string data in the output files are delimited by the double quotation mark (") character. Each item in a row is separated by a comma. The string delimiter character can be overridden using the *delimiter* option and by specifying a single character (such as #) or a four-character string representing a hexadecimal number (such as 0xff) as the delimiter.

The delimited output files are written by default to the directory indicated by the audit archive path option specified in the audit configuration file. To change the output directory, use the *to* option and include the fully qualified path name.

The following three examples show all ways of exporting logs in ASCII delimited format.

Example 6-17 shows how to extract logs in delimited ASCII format into the default archive directory

Example 6-17 Extracting logs into the default archive directory

```
D:\tmp>db2audit extract delasc  from files
d:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.

D:\tmp>dir D:\auditLogs\archive
```

```
 Directory of D:\auditLogs\archive

10/02/2007  11:36 AM                 0 audit.del
10/02/2007  11:37 AM             1,494 auditlobs
10/02/2007  11:37 AM               813 checking.del
10/02/2007  11:37 AM             3,061 context.del
10/02/2007  11:37 AM             1,461 execute.del
10/02/2007  11:36 AM                 0 objmaint.del
10/02/2007  11:36 AM                 0 secmaint.del
10/02/2007  11:36 AM                 0 sysadmin.del
10/02/2007  11:36 AM                 0 validate.del
               9 File(s)          6,829 bytes
```

Example 6-18 shows how to extract logs in delimited ASCII format to a specific directory.

Example 6-18 Extracting logs in delimited ASCII format

```
D:\>db2audit extract delasc to d:\tmp  from files
d:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.

D:\>dir d:\tmp

 Directory of d:\tmp

10/02/2007  11:57 AM                 0 audit.del
10/02/2007  11:57 AM               498 auditlobs
10/02/2007  11:57 AM               271 checking.del
10/02/2007  11:57 AM             1,016 context.del
10/02/2007  11:57 AM               485 execute.del
10/02/2007  11:57 AM                 0 objmaint.del
10/02/2007  11:57 AM                 0 secmaint.del
10/02/2007  11:57 AM                 0 sysadmin.del
10/02/2007  11:57 AM                 0 validate.del
               9 File(s)          2,270 bytes
```

Example 6-19 shows how to extract logs in delimited ASCII format with custom delimiter character.

Example 6-19 Extracting logs with custom delimiter character

```
D:\>db2audit extract delasc delimiter ' to d:\tmp  from files
d:\db2audit.db.SAMPLE.log.0.20070928171143

AUD0000I  Operation succeeded.
```

Example 6-20 shows the same EXECUTE audit record from Example 6-15 on page 136 but in ASCII delimited format.

Example 6-20 Exported log in ASCII delimited format showing one audit record

```
"2007-09-28-17.11.37.831000","EXECUTE","STATEMENT",30,0,"SAMPLE","db2ad
min","DB2ADMIN","DB2ADMIN",0,0,"*LOCAL.DB2.070927223246","db2bp.exe",,,
,,,,,"DB2ADMIN","SQLC2G13",201,,"        ","
",23,1,0,0,"READ_DML","auditlobs.66.24/","CS","auditlobs.90.408/",0,14,
,
```

6.5.3 Load

The exported data from the audit logs either can be reviewed as a plain text or loaded into database tables and then analyzed using the SQL statements. Before we can work with the audit data in database tables, we need to create the actual tables to hold the data.

From a data security perspective, these tables should be created in a separate schema and access to these tables should be restricted only to authorized users, for example, users holding SYSADM or SECADM authorities.

To create the tables, use the *db2audit.ddl* script located in the sqllib/misc directory of the instance. The script assumes that a connection to the database exists and that an 8 K table space is available. The script creates these tables:

- AUDIT
- CHECKING
- OBJMAINT
- SECMAINT
- SYSADMIN
- VALIDATE
- CONTEXT
- EXECUTE

To create the tables for the audit data:

1. Create schema for the audit tables and set it as default (optional):

   ```
   CREATE SCHEMA AUDIT
   SET CURRENT SCHEMA = 'AUDIT'
   ```

2. Run the db2audit.ddl script:

   ```
   db2 -tf $HOME/sqllib/misc/db2audit.ddl
   ```

Loading DB2 audit data into tables

To load the audit data to the audit tables, the audit data must be extracted in ASCII delimited format. This type of file format is required when loading the tables using the LOAD command.

Example 6-21 shows the LOAD commands that we use to populate all the audit tables.

Example 6-21 Loading audit data into tables

```
LOAD FROM audit.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE INSERT
INTO AUDIT.AUDIT;

LOAD FROM checking.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.CHECKING;

LOAD FROM objmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.OBJMAINT;

LOAD FROM secmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.SECMAINT;

LOAD FROM sysadmin.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.SYSADMIN;

LOAD FROM validate.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.VALIDATE;

LOAD FROM context.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.CONTEXT;

LOAD FROM execute.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
INSERT INTO AUDIT.EXECUTE;
```

Note: When loading data:

- Specify the DELPRIORITYCHAR modifier to ensure proper parsing of binary data.
- Specify the LOBSINFILE option of the LOAD command (due to the restriction that any inline data for large objects must be limited to 32 K). In some situations, you might also need to use the LOBS FROM option.

When specifying the file name without the fully qualified path name, the LOAD assumes that the file is located in the current working directory.

Example 6-22 shows loading EXECUTE audit records into the AUDIT.EXECUTE table.

Example 6-22 Loading records into a table from a fie

```
D:\tmp>db2 LOAD FROM execute.del OF DEL MODIFIED BY DELPRIORITYCHAR
LOBSINFILE INSERT INTO AUDIT.EXECUTE
SQL3501W  The table space(s) in which the table resides will not be
placed in backup pending state since forward recovery is disabled for
the database.

SQL3109N  The utility is beginning to load data from file
"D:\tmp\execute.del".

SQL3500W  The utility is beginning the "LOAD" phase at time "10/02/2007
18:55:07.337322".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "2" rows were read
from the input file.

SQL3519W  Begin Load Consistency Point. Input record count = "2".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "10/02/2007
18:55:07.685628".

Number of rows read         = 2
Number of rows skipped      = 0
Number of rows loaded       = 2
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 2
```

6.5.4 Analysis

The data loaded into the audit tables can be analyzed using regular SQL statements. Given that each audit table contains a different set of audit data, by choosing the correct audit table, we can narrow the category of the audit events analyzed.

One of the new features of the DB2 9.5 is the EXECUTE audit category. The EXECUTE category provides you with the capability to accurately track the SQL statements a user issues. In previous DB2 versions, this function was in the CONTEXT category. However, the CONTEXT category also contained audit events other than the executed statements, which resulted in more difficult audit data analysis.

The EXECUTE category captures the SQL statement text as well as the compilation environment and other values that are needed to replay the statement at a later date. When auditing using the EXECUTE category, the statement text for both static and dynamic SQL is recorded, as are input parameter markers and host variables. The EXECUTE category can be audited with or without input values.

Example 6-23 is a snippet of an extracted text audit log where a STATEMENT audit event with input data is shown. The statement issued to extract the data is:

```
select * from department where admrdept=?
```

Because the auditing was set to capture also the input data, we can see that the value of the first parameter was ‘D01’ of type CHAR.

Example 6-23 STATEMENT event from the EXECUTE category with input data

```
timestamp=2007-10-04-11.47.11.700000;
  category=EXECUTE;
  audit event=STATEMENT;
  event correlator=4;
  event status=100;
  database=SAMPLE;
  userid=db2admin;
  authid=DB2ADMIN;
  session authid=DB2ADMIN;
```

```
origin node=0;
coordinator node=0;
application id=*LOCAL.DB2.071004184711;
application name=db2cli.exe;
client application name=audit1.cli;
package schema=DB2ADMIN;
package name=SYSSH200;
package section=4;
local transaction id=0xef73000000000000;
global transaction id=0x0000000000000000000000000000000000000000;
uow id=1;
activity id=1;
statement invocation id=0;
statement nesting level=0;
activity type=READ_DML;
statement text=select * from department where admrdept=?;
statement isolation level=CS;
Compilation Environment Description
  isolation: CS
  query optimization: 5
  min dec div 3: NO
  degree: 1
  SQL rules: DB2
  refresh age: +00000000000000.000000
  resolution timestamp: 2007-10-04-13.44.47.000000
  federated asynchrony: 0
  schema: DB2ADMIN
  maintained table type: SYSTEM;
rows modified=0;
rows returned=2;
value index = 1
  type = CHAR
  data = D01;
```

6.6 Maintenance and administration

The maintenance and administration of the DB2 audit facility is divided into two parts, based on who is authorized to perform them:

- The system administrator (SYSADM) is authorized to perform instance-level administration of the audit subsystem using the **db2audit** command.

- The security administrator (SECADM) can use audit policies in conjunction with the SQL statement, AUDIT, to configure and control the audit requirements for an individual database. The SECADM does not have access to the **`db2audit`** command, and thus cannot configure the instance-level auditing. This, however, also means that the security administrators cannot access the audit logs to archive database level logs and extract data from the audit logs, because this is done using the **`db2audit`** command as well.

  To enable the security administrators to work on the audit logs, DB2 provides audit routines SYSPROC.AUDIT_ARCHIVE, SYSPROC.AUDIT_LIST_LOGS, and SYSPROC.AUDIT_DELIM_EXTRACT to archive audit logs, locate logs of interest, and extract data into delimited files for analysis.

6.6.1 Audit routines

DB2 9.5 introduces new built-in routines for administering the audit subsystem. These routines are designed to be used by the security administrators. Some of the functionality these routines provide is present in the db2audit tool as well. Since the db2audit can be used by the system administrators only, these routines are here mainly for the security administrators.

All the built-in audit routines listed in Table 6-2 are located in the SYSPROC schema.

Table 6-2 Audit routines

Routine name	Operation	Equivalent db2audit command
AUDIT_ARCHIVE	Archives the current audit log	db2audit archive.
AUDIT_LIST_LOGS	Returns a list of the archived audit logs at the specified path, for the current database	<none>
AUDIT_DELIM_EXTRACT	Extracts data from the binary archived logs and loads it into delimited files	db2audit extract

AUDIT_ARCHIVE

The AUDIT_ARCHIVE routine is provided as a stored procedure and also as a table function. They both archive the audit log file for the connected database and have the same syntax.

Syntax

```
>>-AUDIT_ARCHIVE--(--directory--,--dbpartitionnum--)----------><
```

directory	An input argument of type VARCHAR(1024) that specifies the directory where the archived audit files will be written. The directory must exist on the server and the instance owner must be able to create files in that directory. If the argument is null or an empty string, the default directory is used.
dbpartitionnum	An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition and NULL or -2 for an aggregate of all database partitions.

For example:

- To archive the audit logs for all database partitions to the default directory using the procedure:

  ```
  CALL SYSPROC.AUDIT_ARCHIVE(NULL, NULL)
  ```

- To archive the audit logs for all database partitions to the default directory using the table function:

  ```
  SELECT * FROM TABLE(SYSPROC.AUDIT_ARCHIVE('', -2)) AS T1
  ```

AUDIT_LIST_LOGS

The AUDIT_LIST_LOGS table function lists the archived audit log files for a database, which are present in the specified directory.

Syntax

```
>>-AUDIT_LIST_LOGS--(--directory--)---------------------------><
```

directory	An optional input argument of type VARCHAR(1024) that specifies the directory where the archived audit files will be written. The directory must exist on the server and the instance owner must be able to create files in that directory. If the argument is null or an empty string, then the search default directory is used.

Example 6-24 shows using the AUDIT_LIST_LOGS routine to display logs in the default archive directory.

Example 6-24 Using AUDIT_LIST_LOGS routine

```
SELECT * FROM TABLE(SYSPROC.AUDIT_LIST_LOGS('')) AS T1
```

AUDIT_DELIM_EXTRACT

The AUDIT_DELIM_EXTRACT stored procedure performs an extract from archived audit files of the connected database to a delimited file, specifically, to those archived audit files that have file names that match the specified mask pattern.

Syntax

```
>>-AUDIT_DELIM_EXTRACT--(--delimiter--,--target_directory--,--source_di
rectory--,----file_mask--,--event_options--)----><
```

delimiter	An optional input argument of type VARCHAR(1) that specifies the character delimiter to be used in the delimited files. If the argument is null or an empty string, a double quotation mark is used as the delimiter.
target_directory	An optional input argument of type VARCHAR(1024) that specifies the directory where the delimited files are stored. If the argument is null or an empty string, the same directory as the source_directory is used
source_directory	An optional input argument of type VARCHAR(1024) that specifies the directory where the archived audit log files are stored. If the argument is null or an empty string, the audit default is used.
file_mask	An optional input argument of type VARCHAR(1024) is a mask for which files to extract. If the argument is null or an empty string, it extracts from all audit log files in the source directory.
event_options	An optional input argument of type VARCHAR(1024) that specifies the string that defines which events to extract. This matches the same string in the db2audit utility. If the argument is null or an empty string, it extracts all events.

Note: The actual format of the file_mask argument string is in SQL LIKE form (for example, ‘%2007%’), not the wildcard form used in OS shell.

Example 6-25 shows using the AUDIT_DELIM_EXTRACT stored procedure to extract all the audit data into ASCII delimited format files. The output is placed in the default audit directory.

Example 6-25 Extracting audit data using the AUDIT_DELIM_EXTRACT routine

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT(null,null,null,null,null)
```

6.6.2 Synchronous versus asynchronous auditing

The writing of the audit records to the active log can take place synchronously or asynchronously with the occurrence of the events causing the generation of those records. The value of the audit_buf_sz database manager configuration parameter determines when the writing of audit records is done. The auditing is set to synchronous mode by default.

If the value of audit_buf_sz is zero (0), the writing is done synchronously. If the value of audit_buf_sz is greater than zero, the record writing is done asynchronously.

In the asynchronous case, it could be possible for audit records to remain in an unfilled buffer for some time. To prevent this from happening for an extended period, the database manager forces the writing of the audit records regularly. An authorized user of the audit facility may also flush the audit buffer with an explicit request using the **db2audit flush** command.

6.7 Considerations

The DB2 audit subsystem has changed in DB2 9.5 compared to previous releases. The security administrator (SECADM) authority is now required to configure and manage database auditing using SQL statements at the database level. The system administrator (SYSADM) authority is required to configure instance-level auditing only using the **db2audit** command. These functional changes affect migration of the audit configuration and data to the new release.

6.7.1 Transitioning from previous versions to DB2 9.5

In order to be able to manage database auditing at the database level, a security administrator must be created to do the job. This must be done by the system administrator manually. The audit subsystem configuration is migrated automatically. The audit tables created in previous releases are not migrated.

Migration of the audit configuration

The following are the audit subsystem migration steps done automatically by DB2 9.5:

- When you migrate an instance, the audit configuration file is converted to DB2 9.5 format.
- When you migrate a database, the instance-level configuration settings for auditing are used to create an audit policy DB2AUDIT_CFG_MIGR in the database. If the audit facility is enabled at the instance level, the audit policy is associated with the migrated database to enable auditing. Otherwise, the audit policy is not associated. These actions ensure that you observe the same audit behavior on your database after migrating to DB2 9.5.

Migration of the audit tables

The audit tables are not automatically migrated when upgrading to a newer release of DB2. The format of the audit tables holding the audit data changes from release to release to accommodate the enhanced functionality. New columns are added or the size of an existing column can change. Each DB2 version comes with the details on how to create the audit tables that fit the current audit log records. In previous DB2 versions, the layout of the audit table definitions was specified only in documentation. DB2 9.5 introduces the script *db2audit.ddl* to automate creation of these tables.

The difference in audit tables between DB2 8 and DB2 9 is minimum. There is only one column CHKAUTHID added in table CHECKING in DB2 9. Going from DB2 9 to DB2 9.5 makes the audit tables completely different. Each table is heavily extended to hold the transaction IDs, client special registers, trusted context, role details, and other data. Also, there is a brand new table EXECUTE holding the data for the new EXECUTE audit category.

DB2 does not provide any migration tool or script to migrate the audit tables themselves. Given that the layout of the tables differs, the best way to work with the new audit data is to create a new set of the audit tables for the current release, for example, in a new schema, and keep the old audit data in the old schema.

In case you need to migrate the audit tables anyway, we put together two sets of simple scripts to do the job. The first set of scripts creates new audit tables as required by DB2 8, DB2 9, or DB2 9.5. You should use only *one* script for your new DB2 release when no audit tables exist. In case your audit tables already exist, do not use these scripts.

```
db2 -tf db2audit_80.ddl
db2 -tf db2audit_90.ddl
db2 -tf db2audit_95.ddl
```

The second set of scripts allows you to migrate existing audit tables from DB2 8 to DB2 9 or from DB2 9 to DB2 9.5.

```
db2 -tf db2audit_migrate_80_to_90.ddl
db2 -tf db2audit_migrate_90_to_95.ddl
```

These scripts alter the tables as required. For example, if you use DB2 9.5 and have existing audit tables from DB2 9 in a *migrated* database, but still using the old audit layout, you can use the db2audit_migrate_90_to_95.ddl script to update the audit table definitions.

Notes: The db2audit_migrate_80_to_90.ddl and db2audit_migrate_90_to_95.ddl scripts assume that the database is already migrated to the current release using the MIGRATE DATABASE command. The database migration is required when upgrading from DB2 8 to DB2 9 *as well as* from DB2 9 to DB2 9.5. The scripts also assume that the audit tables from the previous release exist.

We recommend that you run REORG on the audit table CONTEXT after the db2audit_migrate_90_to_95.ddl script is used (migrating from DB2 9 to DB2 9.5) due to the ALTER command used in the script.

These scripts are listed in Appendix A, “Sample applications and scripts” on page 229, and can be downloaded from IBM Redbooks Web site. The download instruction is in Appendix B, “Additional material” on page 263.

Data encryption

In today's business world, both governments and global organizations are experiencing an increasing number of malicious attacks from both internal and external sources aimed at sensitive information such as intellectual and financial data, consumer credit card information, medical records, and employee information. Many laws and regulations are now demanding organizations and executive officers not only to ensure the protection of the privacy and confidentiality of electronically stored data, but also to be held accountable if and when this data is compromised. The loss of intellectual and private data could bring severe financial loss and cause a company to lose its competitive advantage. The compromise of sensitive data such as clients' personal data and credit card information could lead to damaging your company's reputation and loss of clients' confidence in conducting business with your organization, which potentially can drive the company to go out of business. These directives specify data encryption as a requirement to protect sensitive data.

In this chapter we discuss the different tools IBM has to help you safeguard your DB2 database and online and offline data from unauthorized access by encrypting it. Most importantly, these tools help you achieve the compliance, governance, and regulations specified in such documents as Consumers's Personal Information (PCI), Gramm-Leach-Bliley Act (GLBA), Insurance Portability and Accountability Act (HIPPA), and US State Notification Laws (SB 1386).

7.1 Overview

As a data security administrator, one part of your security plan should include encryption of your data. Sensitive and personal data stored within the database tables and critical data transmitted across the network, such as user credentials (user ID and password) and production data, are vulnerable and should be protected against intruders.

IBM provides you with the solutions to accomplish this feat. Technologies such as DB2 internal encryption of user ID and password, DB2 column level encryption, and support of external encryption over SSL can help secure the data stored in the database and transmitted through network. Also, IBM Database Encryption Expert can help protect your DB2 database and sensitive data files in both online and offline environments against external and internal intruders, theft, and internal abuse by privileged users. These solutions can help you achieve and maintain compliance with data privacy acts and regulations.

7.2 Encrypting data in transit

DB2 natively supports two types of encryption:

- Internal encryption
  - User ID and password only
  - User ID, password, and user data
- External encryption
  - SSL

The internal encryption protects credentials (user ID and password), or credentials and data, while transferring between the DB2 client and DB2 server. It does not encrypt every single byte of the communication. It secures the data within the DRDA protocol only. The encryption can be enabled on the DB2 server side in the database manager using parameter AUTHENTICATION and value SERVER_ENCRYPT or DATA_ENCRYPT.

The external encryption using SSL is also built in within DB2. It uses SSL libraries from IBM Global Security Kit. It provides secure tunnel (envelope) for data sent between the server and the client. The SSL encryption is more secure and robust than the internal encryption. Every byte of the communication is encrypted. It can be combined with the SERVER_ENCRYPT or DATA_ENCRYPT internal encryption.

7.2.1 Password encryption

The first level of encryption available is encrypting the user ID and password. This is simple and can be used for an application that requires protecting password only.

Example 7-1 shows a DB2 trace of un-encrypted communication where the user ID and password are displayed in plain text (EBCDIC). This is the same data that attackers would see when capturing network traffic.

Example 7-1 Password transfer when no encryption is specified

```
.        SECCHK RQSDSS                (ASCII)          (EBCDIC)
      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF 0123456789ABCDEF
0000 003CD04100030036106E000611A20003 .<.A...6.n...... ..}......>...s..
0010 00162110E2C1D4D7D3C5404040404040 ..!.......@@@@@@ ....SAMPLE
0020 404040404040000E11A1C19393A8D781 @@@@@@..........       ....AllyPa
0030 A2A2A684000811A0C19393A8         ............     sswd....Ally
```

The password encryption can be enabled on the DB2 server by setting AUTHENTICATION to SERVER_ENCRYPT, as shown in Example 7-2.

Example 7-2 Enabling encryption of user ID and password

```
DB2 UPDATE DBM CFG USING AUTHENTICATION SERVER_ENCRYPT
```

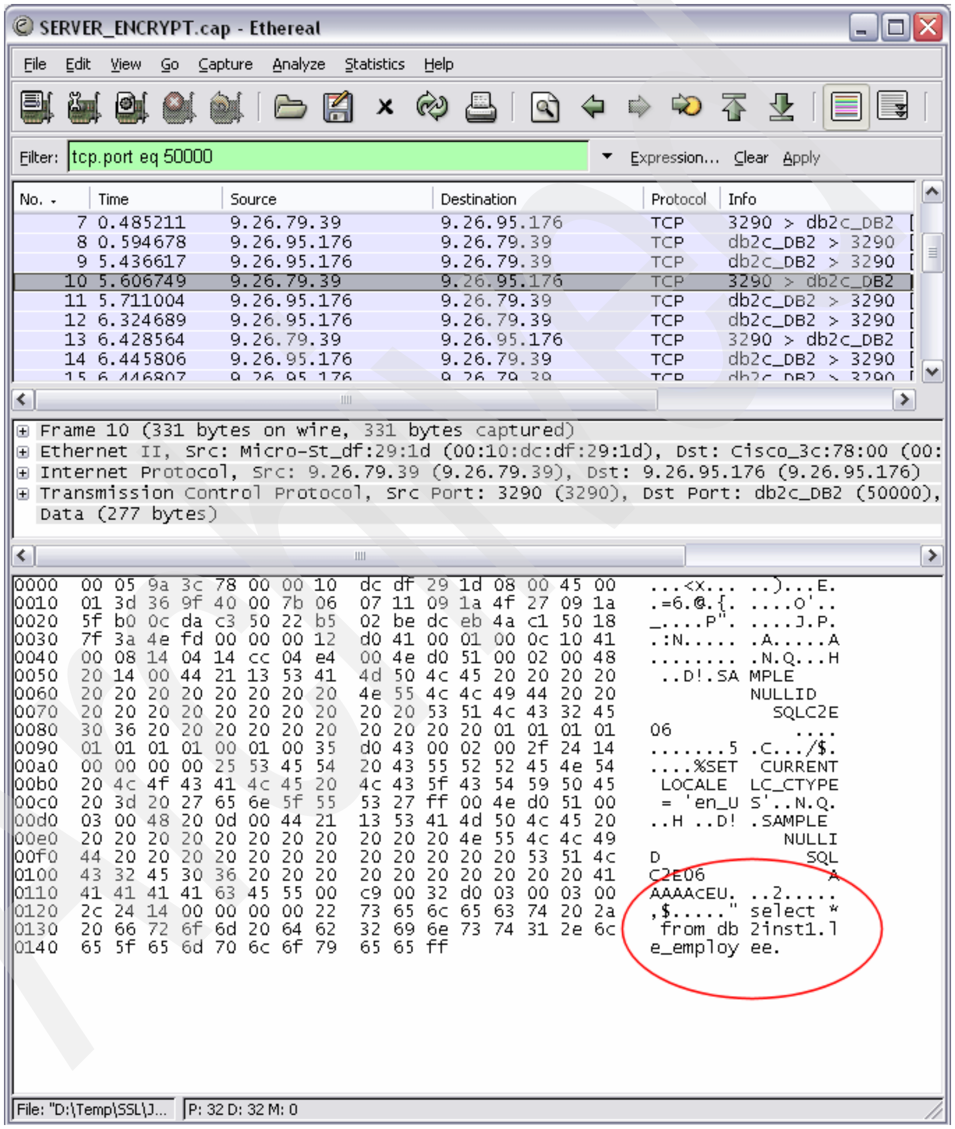
The change of the AUTHENTICATION parameter requires recycling the instance. Once it is activated, all incoming client connections have user IDs and passwords encrypted, as shown in Example 7-3.

Example 7-3 Password transfer when SERVER_ENCRYPT is used

```
.        ACCSEC RQSDSS                (ASCII)          (EBCDIC)
      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF 0123456789ABCDEF
0000 004AD00100020044106D000611A20009 .J.....D.m...... .[}......_...s..
0010 00162110E2C1D4D7D3C5404040404040 ..!.......@@@@@@ ....SAMPLE
0020 4040404040400002411DC769DB79A3CB8 @@@@@@.$..v...<.       ..........
0030 AC9D0DCCB0CC26960743AFBECEE09A33 ......&..C.....3 .......o.....\..
0040 B9E5F4565047B9BED21C             ...VPG....       .V4.&...K.
```

7.2.2 Data encryption

More secure communication requires encryption of the data as well as the user IDs and passwords. The password protection only encrypts the user ID and password. However, the data is still unsecured. It is transferred in plain text, as shown in Figure 7-1.


SERVER_ENCRYPT.cap - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter: tcp.port eq 50000
Expression... Clear Apply

```
No. -   Time        Source         Destination    Protocol Info
      7 0.485211    9.26.79.39     9.26.95.176    TCP      3290 > db2c_DB2 [
      8 0.594678    9.26.95.176    9.26.79.39     TCP      db2c_DB2 > 3290 [
      9 5.436617    9.26.95.176    9.26.79.39     TCP      db2c_DB2 > 3290 [
     10 5.606749    9.26.79.39     9.26.95.176    TCP      3290 > db2c_DB2 [
     11 5.711004    9.26.95.176    9.26.79.39     TCP      db2c_DB2 > 3290 [
     12 6.324689    9.26.95.176    9.26.79.39     TCP      db2c_DB2 > 3290 [
     13 6.428564    9.26.79.39     9.26.95.176    TCP      3290 > db2c_DB2 [
     14 6.445806    9.26.95.176    9.26.79.39     TCP      db2c_DB2 > 3290 [
```

```
⊞ Frame 10 (331 bytes on wire, 331 bytes captured)
⊞ Ethernet II, Src: Micro-St_df:29:1d (00:10:dc:df:29:1d), Dst: Cisco_3c:78:00 (00:
⊞ Internet Protocol, Src: 9.26.79.39 (9.26.79.39), Dst: 9.26.95.176 (9.26.95.176)
⊞ Transmission Control Protocol, Src Port: 3290 (3290), Dst Port: db2c_DB2 (50000),
  Data (277 bytes)
```

```
0000  00 05 9a 3c 78 00 00 10  dc df 29 1d 08 00 45 00   ...<x... ..)...E.
0010  01 3d 36 9f 40 00 7b 06  07 11 09 1a 4f 27 09 1a   .=6.@.{. ....O'..
0020  5f b0 0c da c3 50 22 b5  02 be dc eb 4a c1 50 18   _....P". ....J.P.
0030  7f 3a 4e fd 00 00 00 12  d0 41 00 01 00 0c 10 41   .:N..... .A.....A
0040  00 08 14 04 14 cc 04 e4  00 4e d0 51 00 02 00 48   ........ .N.Q...H
0050  20 14 00 44 21 13 53 41  4d 50 4c 45 20 20 20 20    ..D!.SA MPLE    
0060  20 20 20 20 20 20 20 20  4e 55 4c 4c 49 44 20 20            NULLID  
0070  20 20 20 20 20 20 20 20  20 20 53 51 4c 43 32 45              SQLC2E
0080  30 36 20 20 20 20 20 20  20 20 20 20 01 01 01 01   06           ....
0090  01 01 01 01 00 01 00 35  d0 43 00 02 00 2f 24 14   .......5 .C.../$.
00a0  00 00 00 00 25 53 45 54  20 43 55 52 52 45 4e 54   ....%SET  CURRENT
00b0  20 4c 4f 43 41 4c 45 20  4c 43 5f 43 54 59 50 45    LOCALE  LC_CTYPE
00c0  20 3d 20 27 65 6e 5f 55  53 27 ff 00 4e d0 51 00    = 'en_U S'..N.Q.
00d0  03 00 48 20 0d 00 44 21  13 53 41 4d 50 4c 45 20   ..H ..D! .SAMPLE 
00e0  20 20 20 20 20 20 20 20  20 20 20 4e 55 4c 4c 49               NULLI
00f0  44 20 20 20 20 20 20 20  20 20 20 20 20 53 51 4c   D             SQL
0100  43 32 45 30 36 20 20 20  20 20 20 20 20 20 20 41   C2E06          A
0110  41 41 41 41 63 45 55 00  c9 00 32 d0 03 00 03 00   AAAAcEU. ..2.....
0120  2c 24 14 00 00 00 00 22  73 65 6c 65 63 74 20 2a   ,$....." select *
0130  20 66 72 6f 6d 20 64 62  32 69 6e 73 74 31 2e 6c    from db 2inst1.l
0140  65 5f 65 6d 70 6c 6f 79  65 65 ff                  e_employ ee.
```

Figure 7-1 DB2 communication buffers with un-encrypted data (SERVER_ENCRYPT)

The data encryption can be enabled on the DB2 server by setting AUTHENTICATION to DATA_ENCRYPT, as shown in Example 7-4.

Example 7-4 Enabling data encryption

```
DB2 UPDATE DBM CFG USING AUTHENTICATION DATA_ENCRYPT
```

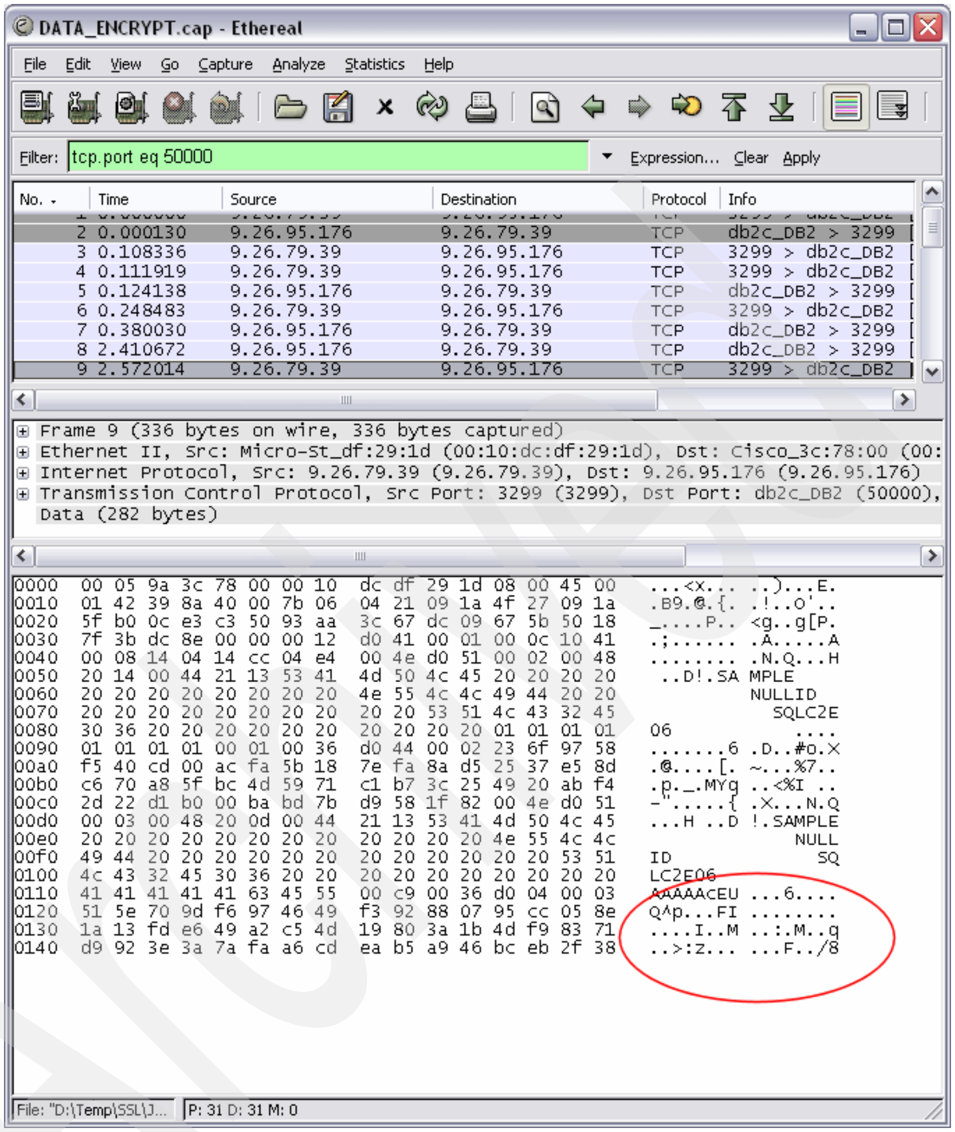
Using DATA_ENCRYPT means that user IDs and passwords are encrypted the same way as though the SERVER_ENCRYPT was used, but in addition to this, it encrypts the user data. Using DATA_ENCRYPT, however, does not encrypt every byte of the data transported, only the following data structures:

- SQL and XQuery statements
- SQL program variable data
- Some or all of the answer set data resulting from a query
- Large object (LOB) data streaming
- SQLDA descriptors
- Output data from the server processing an SQL statement and including a description of the data

Figure 7-2 on page 156 shows encrypted data as captured in the network. If you compare it with Figure 7-1 on page 154, you find that the SET CURRENT ... and the *select * from* ... statement were encrypted. However, not all the data in the buffer is encrypted. You can see the SQL environment settings:

- Relational Database Name=SAMPLE
- RDB Collection Identifier=NULLID
- RDB Package Identifier=SQLC2E06

To effectively hide everything you need to use SSL encryption.


DATA_ENCRYPT.cap - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
tcp.port eq 50000
```

```
No. -  Time      Source        Destination   Protocol Info
     2 0.000130  9.26.95.176   9.26.79.39    TCP      db2c_DB2 > 3299 [
     3 0.108336  9.26.79.39    9.26.95.176   TCP      3299 > db2c_DB2 [
     4 0.111919  9.26.79.39    9.26.95.176   TCP      3299 > db2c_DB2 [
     5 0.124138  9.26.95.176   9.26.79.39    TCP      db2c_DB2 > 3299 [
     6 0.248483  9.26.79.39    9.26.95.176   TCP      3299 > db2c_DB2 [
     7 0.380030  9.26.95.176   9.26.79.39    TCP      db2c_DB2 > 3299 [
     8 2.410672  9.26.95.176   9.26.79.39    TCP      db2c_DB2 > 3299 [
     9 2.572014  9.26.79.39    9.26.95.176   TCP      3299 > db2c_DB2 ]
```

```
⊞ Frame 9 (336 bytes on wire, 336 bytes captured)
⊞ Ethernet II, Src: Micro-St_df:29:1d (00:10:dc:df:29:1d), Dst: Cisco_3c:78:00 (00:
⊞ Internet Protocol, Src: 9.26.79.39 (9.26.79.39), Dst: 9.26.95.176 (9.26.95.176)
⊞ Transmission Control Protocol, Src Port: 3299 (3299), Dst Port: db2c_DB2 (50000),
  Data (282 bytes)
```

```
0000  00 05 9a 3c 78 00 00 10  dc df 29 1d 08 00 45 00   ...<x... ..)...E.
0010  01 42 39 8a 40 00 7b 06  04 21 09 1a 4f 27 09 1a   .B9.@.{. .!..O'..
0020  5f b0 0c e3 c3 50 93 aa  3c 67 dc 09 67 5b 50 18   _....P.. <g..g[P.
0030  7f 3b dc 8e 00 00 00 12  d0 41 00 01 00 0c 10 41   .;...... .A.....A
0040  00 08 14 04 14 cc 04 e4  00 4e d0 51 00 02 00 48   ........ .N.Q...H
0050  20 14 00 44 21 13 53 41  4d 50 4c 45 20 20 20 20    ..D!.SA MPLE
0060  20 20 20 20 20 20 20 20  4e 55 4c 4c 49 44 20 20            NULLID
0070  20 20 20 20 20 20 20 20  20 20 53 51 4c 43 32 45              SQLC2E
0080  30 36 20 20 20 20 20 20  20 20 20 20 01 01 01 01   06           ....
0090  01 01 01 01 00 01 00 36  d0 44 00 02 23 6f 97 58   .......6 .D..#o.X
00a0  f5 40 cd 00 ac fa 5b 18  7e fa 8a d5 25 37 e5 8d   .@....[. ~...%7..
00b0  c6 70 a8 5f bc 4d 59 71  c1 b7 3c 25 49 20 ab f4   .p._.MYq ..<%I ..
00c0  2d 22 d1 b0 00 ba bd 7b  d9 58 1f 82 00 4e d0 51   -".....{ .X...N.Q
00d0  00 03 00 48 20 0d 00 44  21 13 53 41 4d 50 4c 45   ...H ..D !.SAMPLE
00e0  20 20 20 20 20 20 20 20  20 20 20 20 4e 55 4c 4c                NULL
00f0  49 44 20 20 20 20 20 20  20 20 20 20 20 20 53 51   ID             SQ
0100  4c 43 32 45 30 36 20 20  20 20 20 20 20 20 20 20   LC2E06
0110  41 41 41 41 41 63 45 55  00 c9 00 36 d0 04 00 03   AAAAAcEU ...6....
0120  51 5e 70 9d f6 97 46 49  f3 92 88 07 95 cc 05 8e   Q^p...FI ........
0130  1a 13 fd e6 49 a2 c5 4d  19 80 3a 1b 4d f9 83 71   ....I..M ..:.M..q
0140  d9 92 3e 3a 7a fa a6 cd  ea b5 a9 46 bc eb 2f 38   ..>:z... ...F../8
```

```
File: "D:\Temp\SSL\J...   P: 31 D: 31 M: 0
```

Figure 7-2 DB2 communication with encrypted data (DATA_ENCRYPT)

7.2.3 Encryption using SSL

DB2 9 supports SSL connections starting with fix pack 2. You can use SSL to connect from Java client applications, using the IBM Data Server Driver for JDBC and SQLJ in Type 4 mode to DB2 servers or gateways. At the time of writing, SSL cannot be used between a DB2 client and a DB2 server or between any other applications that do not use IBM Data Server Driver for JDBC and SQLJ driver Type 4.

IBM Data servers with SSL support

SSL support is included in the following DB2 versions:

- DB2 9 Fix Pack 2 for AIX, HP, Linux, Solaris, Windows
- DB2 9 for z/OS

The SSL support is not available between the DB2 Connect™ (gateway) product and the host or System i™ database. SSL support is available, however, between IBM Data Server Driver for JDBC and SQLJ (type 4 connectivity) on the DB2 client and the gateway DB2 database product.

The connection concentrator is not supported either. SSL support is not enabled in the DB2 instance if the connection concentrator is running.

Client applications where the SSL connectivity is supported

All the Java applications using the IBM Data Server Driver for JDBC and SQLJ in Type 4 mode support SSL connectivity.

Configuring SSL support in DB2 instances

SSL support in DB2 is provided by IBM Global Security Kit (GSKit). GSKit provides Secure Sockets Layer (SSL) data encryption and also contains the IBM Key Management (iKeyMan) utility for creating key databases, public-private key pairs, and certificate requests.

The GSKit is *automatically* installed during the DB2 setup. Before the SSL support can be configured for a DB2 instance, the following must be done:

- The path to the GSKit libraries must be in the PATH environment variable on Windows and in the LIBPATH, SHLIB_PATH, or LD_LIBRARY_PATH environment variables on Linux and UNIX.
- A key database must be created with the certificates in the iKeyMan. These keys are then used for establishing the actual SSL connections.

Note: DB2 9 is compatible with the GSKit Version 7 (directory name gskit7). There may be other GSKit versions present on your system and used by different applications, for example, gskit6 or gskit2, but these are not compatible with DB2 9. When configuring the PATH, LIBPATH, SHLIB_PATH, or LD_LIBRARY_PATH with DB2 9, make sure that you use the gskit7.

Add GSKit libraries to the PATH on Windows

The GSKit libraries are located on Windows in directory C:\Program Files\ibm\gsk7\lib, regardless of which drive/directory DB2 is installed in. Thus, the system PATH must include the C:\Program Files\ibm\gsk7\lib.

Add GSKit libraries to the LIBPATH on UNIX

On AIX, the GSKit libraries are located in directory /usr/opt/ibm/gskta/lib (32 bit) and /usr/opt/ibm/gsksa/lib64 (64 bit). On HP-UX and Solaris, the GSKit libraries are located in directory /opt/ibm/gsk7/lib. The GSKit setup automatically creates links to these libraries in the /usr/lib directory. This means that you only have to add /usr/lib to the LIBPATH.

Add GSKit libraries to the LD_LIBRARY_PATH on Linux

The GSKit libraries are located on Linux in directory /usr/local/ibm/gsk7/lib. The GSKit setup automatically creates links to these libraries in the /usr/lib directory, so you only have to add /usr/lib to the LD_LIBRARY_PATH.

Configuring GSKit

To use SSL, you must create a key database using the gsk7ikm GUI tool from the gskit7\bin directory.

Note: The gsk7ikm is a Java application. It expects the JAVA_HOME to be set to the JDK/JRE™ root directory.

- On Windows this can be done using:

```
set JAVA_HOME=D:\SQLLIB\java\jdk
```

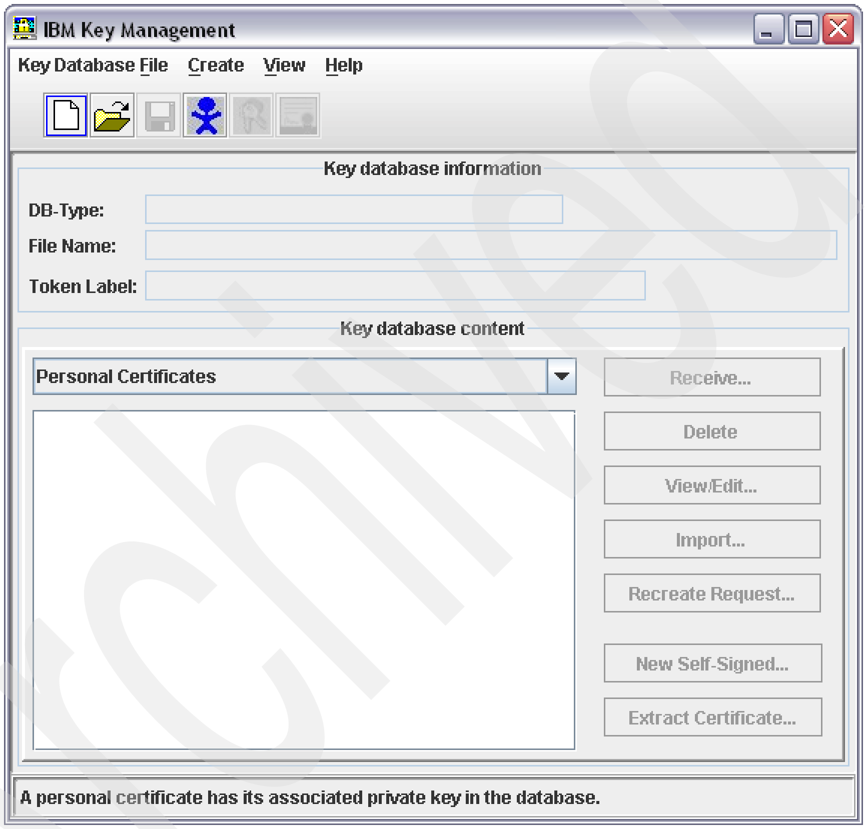
- On UNIX and Linux, this can be done using:

```
export JAVA_HOME=$HOME/sqllib/java/jdk
```

Use the following steps to configure GSKit:

1. Run **gsk7ikm**.

   The **gsk7ikm** is located in the gsk7\bin directory on Windows or in the gsk7/bin directory on Linux/UNIX. To run it use the Windows Explorer or command prompt on Windows, or the OS shell on Linux/UNIX. Figure 7-3 shows the main window of IBM Key Management tool when running **gsk7ikm**.

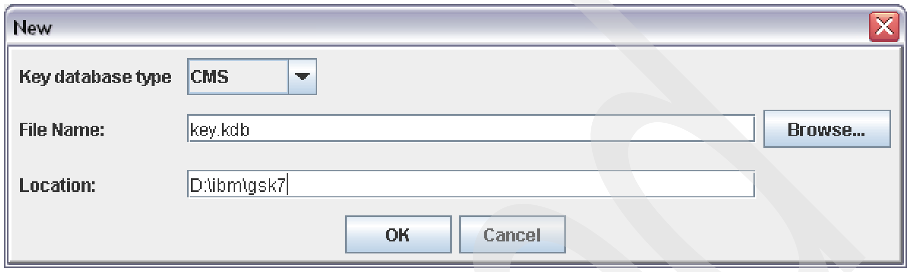


A personal certificate has its associated private key in the database.

Figure 7-3 IBM Key Management main window

2. Create a database for the keys.

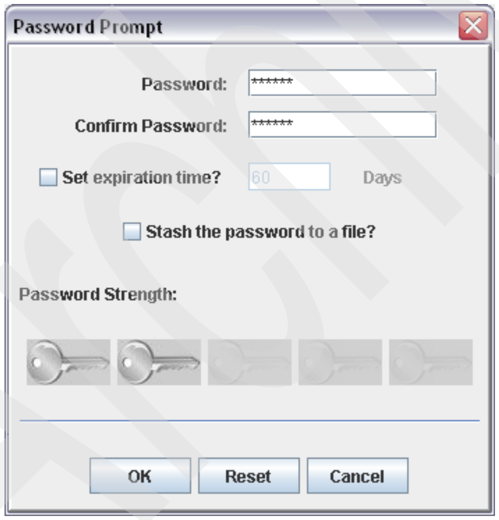
   Use menu item **Key Database File** → **New** to create the new database. In the New panel, select **CMS** as the Key database type and specify the name and the location of the file that stores the database keys, as shown in Figure 7-4. Click **OK**.



Key database type: CMS
File Name: key.kdb
Location: D:\ibm\gsk7

Figure 7-4 Creating a new key database file

3. In the Password Prompt panel (Figure 7-5), specify the password used to protect the key database file. Click **OK**.



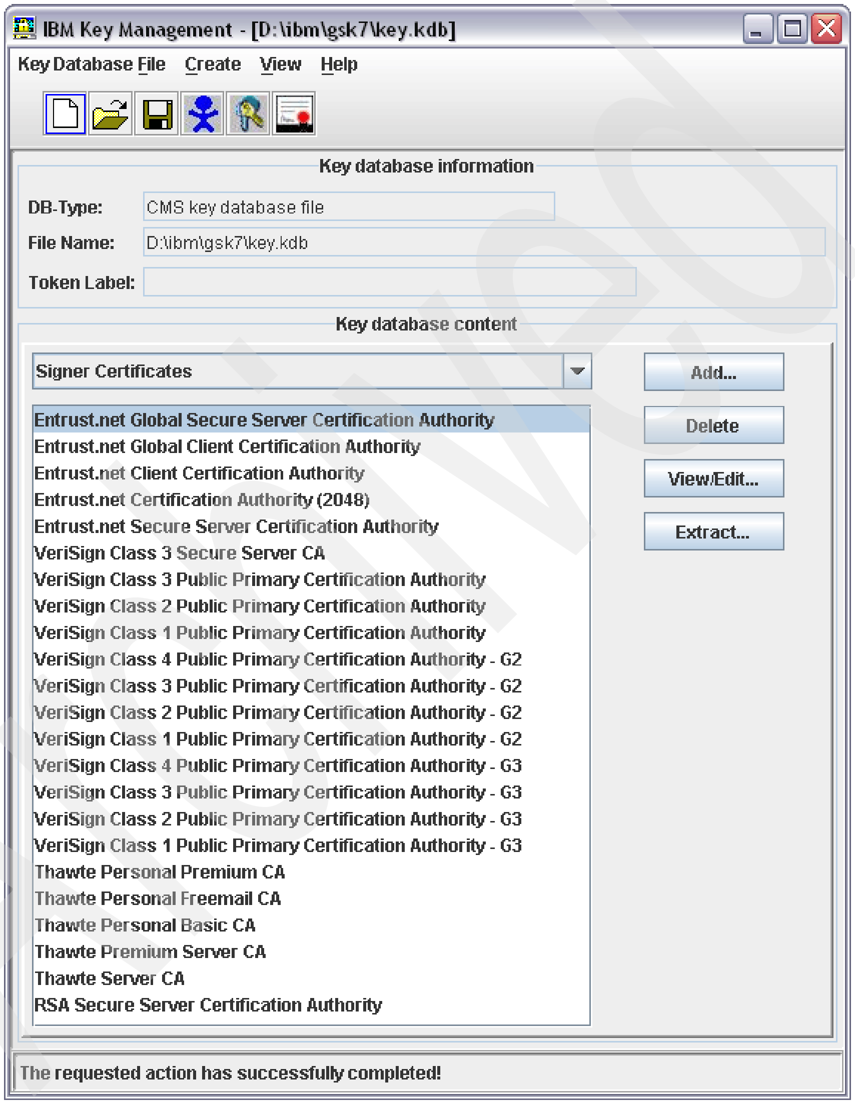
Password: ******
Confirm Password: ******
Set expiration time? 60 Days
Stash the password to a file?
Password Strength:

Figure 7-5 Password Prompt window

The Password Prompt panel requires the password to be entered twice and also shows graphically the strength of the password entered. The strongest password contains numbers, lowercase and uppercase letters, and non-alphanumeric characters.

Note: This password will be used later in configuring SSL in DB2.

4. Figure 7-6 shows that the key database file is created and is pre-filled with certification authorities.



DB-Type: CMS key database file
File Name: D:\ibm\gsk7\key.kdb
Token Label:

Signer Certificates

Entrust.net Global Secure Server Certification Authority
Entrust.net Global Client Certification Authority
Entrust.net Client Certification Authority
Entrust.net Certification Authority (2048)
Entrust.net Secure Server Certification Authority
VeriSign Class 3 Secure Server CA
VeriSign Class 3 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 4 Public Primary Certification Authority - G2
VeriSign Class 3 Public Primary Certification Authority - G2
VeriSign Class 2 Public Primary Certification Authority - G2
VeriSign Class 1 Public Primary Certification Authority - G2
VeriSign Class 4 Public Primary Certification Authority - G3
VeriSign Class 3 Public Primary Certification Authority - G3
VeriSign Class 2 Public Primary Certification Authority - G3
VeriSign Class 1 Public Primary Certification Authority - G3
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
RSA Secure Server Certification Authority

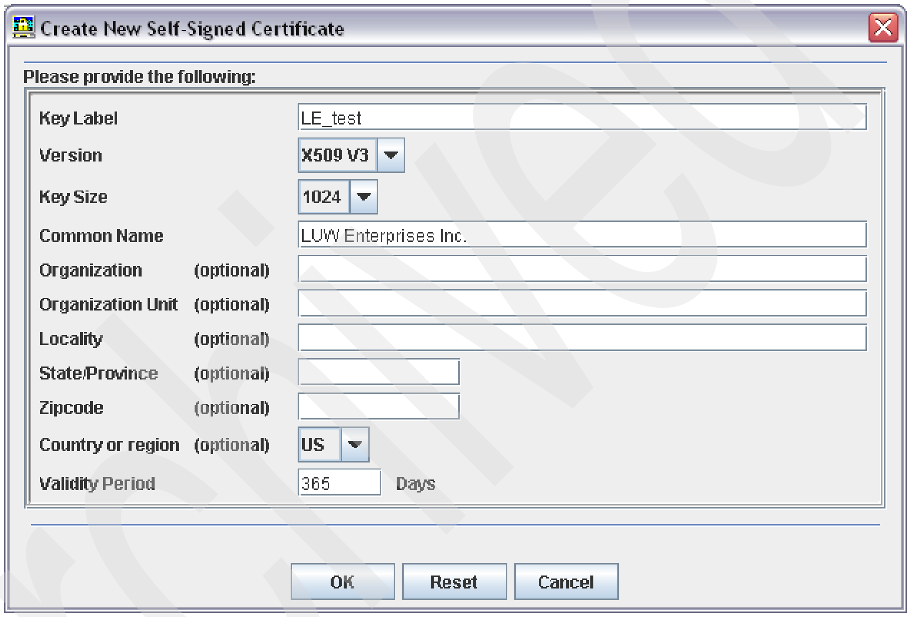
The requested action has successfully completed!

Figure 7-6 The iKeyMan window with a key database file opened

5. Add a certificate to the key database.

   Once the key database file is created, you need to add a certificate to it. This can be done either by creating a New Certificate Request and having this signed by the true Certification Authority or by creating a self-signed certificate.

   For demonstration purposes, we create a new self-signed certificate. Use menu item **New** → **New Self-Signed Certificate** to create the new certificate. In the Create New Self-Signed Certificate window, enter the required details, as shown in Figure 7-7.



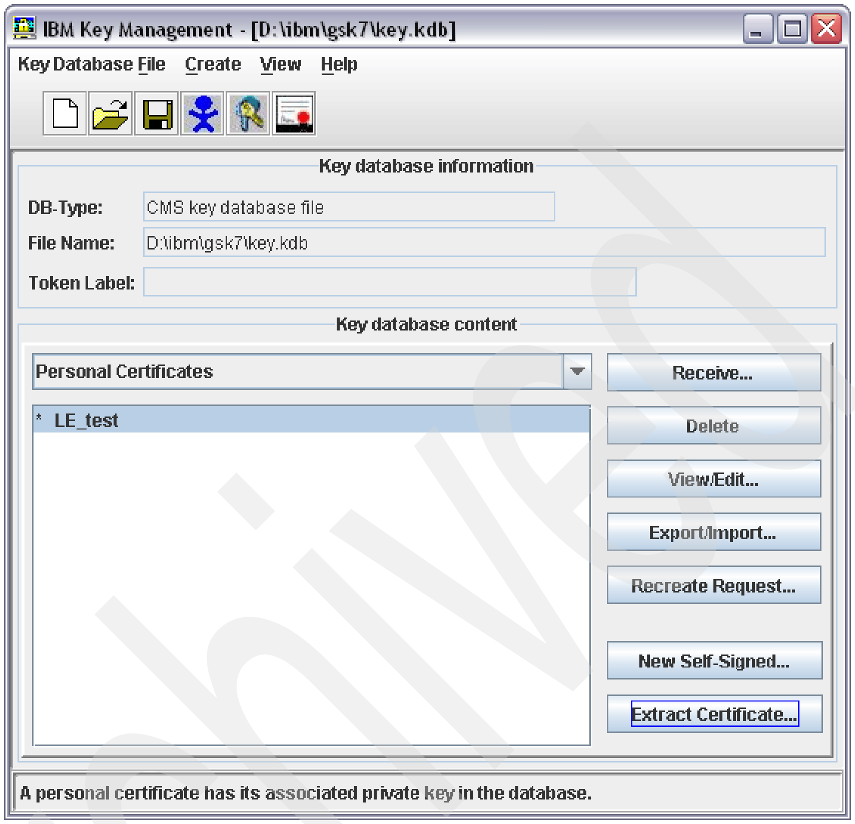
Create New Self-Signed Certificate

Please provide the following:

Key Label		LE_test
Version		X509 V3
Key Size		1024
Common Name		LUW Enterprises Inc.
Organization	(optional)	
Organization Unit	(optional)	
Locality	(optional)	
State/Province	(optional)	
Zipcode	(optional)	
Country or region	(optional)	US
Validity Period		365 Days

Figure 7-7 Creating self-signed certificate

6. Figure 7-8 shows main iKeyMan panel with the self-signed certificated.



DB-Type: CMS key database file

File Name: D:\ibm\gsk7\key.kdb

Token Label:

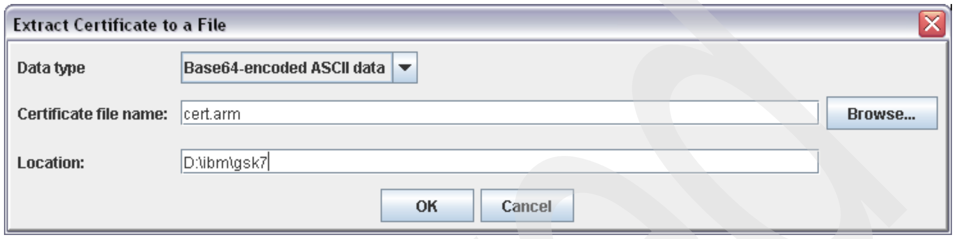
Personal Certificates

* LE_test

A personal certificate has its associated private key in the database.

Figure 7-8 The iKeyMan window with one personal certificate

This concludes the steps needed for creating the key database in the GSKit. Before we proceed to configure a DB2 instance, we extract the certificate we just created, because they are needed on the client side for establishing the actual SSL connection. To extract the certificate click **Extract Certificate** in the iKeyMan panel while a personal certificate is selected. In the Extract Certificate to a File panel, enter file name and location for the certificate to be stored, as shown in Figure 7-9.


Extract Certificate to a File

Data type: Base64-encoded ASCII data

Certificate file name: cert.arm

Location: D:\ibm\gsk7

Figure 7-9 Extracting the certificate to a file d:\ibm\gsk7\cert.arm

Configuring DB2 instance

The configuration of SSL in a DB2 instance requires a DB2 instance owner to perform the following steps:

1. Create an SSL configuration file.

   – Linux and UNIX: *INSTHOME*/cfg/SSLconfig.ini
   – Windows: *INSTHOME*/SSLconfig.ini

   Where *INSTHOME* is the home directory of the instance. We recommend that the file permission is set to limit access to the SSLconfig.ini, as the file contains sensitive data, like password to the key database.

2. Configure SSL using the SSLconfig.ini file.

   This is required for DB2 to be able to load and start SSL. The required data in the file includes:

   – DB2_SSL_KEYSTORE_FILE

     Fully qualified file name of the KeyStore that stores the server certificate.

   – DB2_SSL_KEYSTORE_PW

     Password of the KeyStore that stores the server certificate.

   – DB2_SSL_LISTENER

     Service name or port number for the SSL listener. This port number is something that you choose. It can be any port number currently not in use. When the DB2 instance with SSL support starts, it creates a listener who listens on this port number for incoming SSL connections.

Example 7-5 shows the SSLconfig.ini in our lab.

Example 7-5 Sample SSLconfig.ini file

```
DB2_SSL_KEYSTORE_FILE=d:\ibm\gsk7\key.kdb
DB2_SSL_KEYSTORE_PW=LUWEnt
DB2_SSL_LISTENER=40000
```

3. Enable SSL communication for the instance.

   Use the **db2set DB2COMM** command to enable the SSL communication for the instance as follows:

   – Enable only SSL access:

     ```
     db2set DB2COMM=SSL
     ```

   – Enable both SSL and TCP/IP:

     ```
     db2set DB2COMM=SSL,TCPIP
     ```

4. Restart the instance.

   If there is no error when starting the instance, SSL is configured correctly and the DB2 listens for incoming SSL connections from the clients at the port specified by the DB2_SSL_LISTENER keyword in the SSLconfig.ini.

   In case of a problem with the SSL configuration, when starting the DB2 instance, an error SQL5043N is returned. More details about the actual problem are captured in the db2diag.log. For example, using a wrong password for the key database file produces the following entries in the db2diag.log:

```
2007-10-19-15.27.10.605000-420 I122427H375        LEVEL: Error
PID     : 7240                 TID  : 6936        PROC :
db2syscs.exe
INSTANCE: DB2                  NODE : 000
EDUID   : 6936                 EDUNAME: db2sysc 0
FUNCTION: DB2 UDB, common communication, sqlccMapSSLErrorToDB2Error,
probe:530
MESSAGE : ssl return code in sqlcctcpconnmgrLoadSSLLibrary : 408

2007-10-19-15.27.10.635000-420 I122804H430        LEVEL: Error
PID     : 7240                 TID  : 6936        PROC :
db2syscs.exe
INSTANCE: DB2                  NODE : 000
EDUID   : 6936                 EDUNAME: db2sysc 0
FUNCTION: DB2 UDB, common communication,
sqlcctcpconnmgrLoadSSLLibrary, probe:998
MESSAGE : SSL wasn't setup
DATA #1 : Hexdump, 4 bytes
```

```
0x01C1FC8C : 4C00 0000                                  L...
```

By examining the SSL error code 408 we can find this explanation:

```
The supplied password for the keyfile was not correct
```

Configuring SSL support for Java applications

The SSL connectivity in Java applications can be used when these applications use IBM Data Server Driver for JDBC and SQLJ, and the driver type is Type 4. Other drivers or driver types cannot be configured to work with SSL. The IBM Data Server Driver for JDBC and SQLJ provides support for SSL through the Java Secure Socket Extension (JSSE).

In order to set up SSL for the Java applications it is necessary to configure the Java subsystem as well as modify the Java application.

Configuring the Java subsystem

In order to use SSL it is necessary to configure the Java subsystem, JDK™. We need to specify what security provider we want to use and also import certificates to the Java truststore so that they can be later used for securing the communication. In general, it is possible to have more than one JDK on a system. We need to configure the JDK that is used by our Java application. Because we are going to use an IBM security provider, it has to be an IBM JDK.

Note: The DB2 9 installation contains an IBM JDK that is used by DB2 to run the Java tools and stored procedures. It is located under the SQLLIB\java\jdk directory on Windows or under the sqllib/java/jdk directory on Linux/UNIX. This JDK can be used for running our Java application with SSL connections.

Perform these steps to configure the Java subsystem:

1. Import a certificate from the DB2 server to a Java truststore on the client.

   From operating system shell/prompt run a **keytool** command located in the jdk\bin directory on Windows or in the jdk/bin directory on Linux/UNIX. This command imports a certificate from the server into the Java truststore on the client, as shown in the Example 7-6.

   Example 7-6 Importing a server certificate to the client

   ```
   keytool -import -file d:\ibm\gsk7\cert.arm -keystore LE_cacerts

   Enter keystore password:  LE_keystore
   Owner: CN=LUW Enterprises Inc., C=US
   Issuer: CN=LUW Enterprises Inc., C=US
   Serial number: 47192434
   Valid from: 10/18/07 2:40 PM until: 10/18/08 2:40 PM
   ```

```
Certificate fingerprints:
         MD5:  DA:85:6C:88:26:49:AE:26:A8:C3:C4:72:82:92:80:58
         SHA1:
05:21:03:72:50:4E:9F:BF:00:12:C9:47:74:0B:EE:9E:35:F0:DB:9E
Trust this certificate? [no]:  y
Certificate was added to keystore
```

The **keytool** command creates a file LE_cacerts in the current directory. This is the keystore the Java application will use for the SSL connections.

2. Configure the Java security provider.

   IBM Data Server Driver for JDBC and SQLJ can use the IBM FIPS-compliant provider for SSL or the SunJSSE provider. In our example we use the IBMJSSE2 provider in FIPS mode.

   Note: FIPS stands for Federal Information Processing Standard. FIPS 140-2 is a standard that describes US Federal government requirements that computer systems must meet for protecting sensitive but unclassified information. A Java version of the crypto modules (JSSE/JCE) is included in the IBM JDK and the hybrid Sun/HP JDK.

   The security providers are configured through the java.security file located in the lib\security directory of the current Java Runtime Environment (JRE), for example, in the D:\SQLLIB\java\jdk\jre\lib\security\ directory. This file contains a list of security providers in their order of preference. We need to add the IBMJCEFIPS cryptographic provider, as shown in Example 7-7.

   Example 7-7 Adding the IBMJCEFIPS provider

```
security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
security.provider.2=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
security.provider.6=com.ibm.security.sasl.IBMSASL
```

   The order number of the IBMJCEFIPS provider must be smaller than the order number of the IBMJCE provider and IBMJCE has to be kept in the list because it is needed for the keystore support.

In addition to adding the security provider to the java.security file, you also have to configure the SSLSocketFactory by adding two lines to the java.security file, as shown in Example 7-8.

Example 7-8 Configuring the SSLSocketFactory

```
ssl.SocketFactory.provider=com.ibm.jsse2.SSLSocketFactoryImpl
ssl.ServerSocketFactory.provider=com.ibm.jsse2.SSLServerSocketFactoryImpl
```

3. Set the IBMJSSE2 provider to FIPS mode.

   This is done by setting the com.ibm.jsse2.JSSEFIPS Java system property to true. We set the mode on the command line as shown in Example 7-10 on page 169.

4. Configure the Java system properties to use the trustStore.

   This is done by setting the following Java system properties:

   – javax.net.ssl.trustStore
   – javax.net.ssl.trustStorePassword

   As a simple test, we set these properties on the command line, as shown in Example 7-10 on page 169.

Modifying the Java application

From the application's point of view, enabling SSL in IBM Data Server Driver for JDBC and SQLJ requires only one change: the property *sslConnection* must be set to true. This forces the IBM Data Server Driver for JDBC and SQLJ to use an SSL socket to connect to DB2 database servers. This property can be set through both DB2DataSource and DriverManager interfaces as follows:

- DB2DataSource interface:

```
com.ibm.db2.jcc.DB2DataSource = new com.ibm.db2.jcc.DB2DataSource();
dataSource.setSslConnection(true);
java.sql.Connection con = dataSource.getConnection();
```

- DriverManager interface:

```
java.util.Properties properties = new java.util.Properties();
properties.put("sslConnection", "true");
java.sql.Connection con = java.sql.DriverManager.getConnection(url,
properties);
```

Running a Java application in SSL mode

To show how securing a connection using SSL for a Java application works, we put together a simple command-line Java application named *SSL.java*, which uses IBM Data Server Driver for JDBC and SQLJ driver Type 4. This application establishes an SSL connection using the DB2DataSource interface and issues a SELECT statement to retrieve data.

Example 7-9 explains the syntax of the test application. The source code of SSL.java is listed in “ssl.java” on page 258 and is available for download. The download instructions are in Appendix B, “Additional material” on page 263.

Example 7-9 Syntax of the test application

```
Usage:
SSL database server port userId password
```

database	database alias name
server	host name or IP address of the DB2 server
port	port number where the DB2 server listens for the *SSL* connections
userId	user account used to login to the database
password	password for the user ID above

Example 7-10 shows running the application and retrieving data from the DB2 server through an SSL connection.

Example 7-10 Running the Java application that connects to the DB2 server using SSL

```
java -Dcom.ibm.jsse2.JSSEFIPS=true
-Djavax.net.ssl.trustStore=LE_cacerts
-Djavax.net.ssl.trustStorePassword=LE_keystore SSL sample localhost
40000 Ally AllyPasswd

Columns: 15
EMPNO  FIRSTNME     MIDINIT LASTNAME        WORKDEPT PHONENO HIREDATE
JOB      EDLEVEL SEX BIRTHDATE  SALARY      BONUS       COMM
ROWSEC
------ ------------ ------- --------------- -------- ------- ----------
-------- ------- --- ---------- ----------- ----------- -----------
----------------
012214 EVAN         B       WELSH           C21      3782    2006-10-11
SeniorSR 17      M   1980-04-12 42250.00    400.00      10780.00
9999000000000000
Rows: 1
```

Note: When connecting to DB2 using SSL, the port the application uses to connect to the database must be the SSL listener port, as configured in the DB2 server in the SSLconfig.ini file. An attempt to connect using SSL to the regular TCP/IP port, where the DB2 server listens for the regular connections, results in an error.

Using the SSL connection is transparent to the application. The connection behaves the same way as a regular TCP/IP connection. For diagnostic purposes, if we want to know whether an application is using SSL. The only way is to check the DB2BaseDataSource.sslConnection property. If it is set to *true* then SSL is used. This can be checked, for example, using a JCC trace, where the tracing facility can dump out all properties, as shown in Example 7-11.

Example 7-11 Snippet of a JCC trace showing the sslConnection property set

```
[jcc] Using properties: { PLOAD_LO_PATH=null, DBANSIWARN=false,
kerberosServerPrincipal=null, DBSPACETEMP=null,
optimizationProfileToFlush=null, currentDegree=null, description=null,
clientProgramId=null, maxRetriesForClientReroute=-1,
currentPackagePath=null, retrieveMessagesFromServerOnGetMessage=false,
useRowsetCursor=true, loginTimeout=0, IFX_FLAT_UCSQ=null,
connectNode=-1, currentSQLID=null, useCachedCursor=true, DUMPCORE=null,
useTransactionRedirect=false, sslConnection=true, ... }
```

IBM Data Server Driver for JDBC and SQLJ does not show many details when there is a problem with the SSL setup. In Example 7-12, we show running the test application with an incorrect password for the trustStore.

Example 7-12 Running the test application with an incorrect truststore password

```
java -Dcom.ibm.jsse2.JSSEFIPS=true
-Djavax.net.ssl.trustStore=LE_cacerts
-Djavax.net.ssl.trustStorePassword=wrongPassword SSL sample localhost
40000 Ally AllyPasswd

Execute exception: com.ibm.db2.jcc.b.gm:
[jcc][t4][2043][11550][3.50.152] Exception java.net.SocketException:
Error opening socket to server localhost/127.0.0.1 on port 40,000 with
message: Unconnected sockets not implemented. ERRORCODE=-4499,
SQLSTATE=08001.
com.ibm.db2.jcc.b.gm: [jcc][t4][2043][11550][3.50.152] Exception
java.net.SocketException: Error opening socket to server
localhost/127.0.0.1 on port 40,000 with message: Unconnected sockets
not implemented. ERRORCODE=-4499, SQLSTATE=08001
at com.ibm.db2.jcc.b.wc.a(wc.java:283)
at com.ibm.db2.jcc.t4.xb.a(xb.java:354)
at com.ibm.db2.jcc.t4.xb.<init>(xb.java:73)
at com.ibm.db2.jcc.t4.a.w(a.java:260)
at com.ibm.db2.jcc.t4.b.a(b.java:1775)
at com.ibm.db2.jcc.b.eb.a(eb.java:452)
at com.ibm.db2.jcc.b.eb.<init>(eb.java:417)
at com.ibm.db2.jcc.t4.b.<init>(b.java:296)
```

```
at
com.ibm.db2.jcc.DB2DataSource.getSimpleConnection(DB2DataSource.java:14
0)
at com.ibm.db2.jcc.DB2DataSource.getConnection(DB2DataSource.java:116)
at com.ibm.db2.jcc.DB2DataSource.getConnection(DB2DataSource.java:91)
at SSL.main(SSL.java:41)

Caused by: java.net.SocketException: Unconnected sockets not
implemented
at javax.net.SocketFactory.createSocket(SocketFactory.java:2)
at com.ibm.db2.jcc.t4.z.run(z.java:37)
at
java.security.AccessController.doPrivileged(AccessController.java:242)
at com.ibm.db2.jcc.t4.xb.a(xb.java:344)
... 10 more
```

The exception messages in Example 7-12 on page 170 only state that there is a problem opening a socket and that unconnected sockets are not implemented. To obtain more details about the actual problem, you can enable debugging for the Java network component, which lists much more useful information, as shown in Example 7-13.

Example 7-13 Running the test application with incorrect password in debug mode

```
java -Dcom.ibm.jsse2.JSSEFIPS=true
-Djavax.net.ssl.trustStore=LE_cacerts
-Djavax.net.ssl.trustStorePassword=wrongPassword -Djavax.net.debug=true
SSL sample localhost 40000 Ally AllyPasswd

setting up default SSLSocketFactory
use class specified by ssl.SocketFactory.provider:
com.ibm.jsse2.SSLSocketFactoryImpl
class com.ibm.jsse2.SSLSocketFactoryImpl is loaded
init keymanager of type IbmX509
IBMJSSEProvider2 Build-Level: -20070313
keyStore is: D:\SQLLIB\java\jdk\jre\lib\security\cacerts
keyStore type is: jks
keyStore provider is:
init keystore
trustStore is: LE_cacerts
trustStore type is: jks
trustStore provider is:
init truststore
```

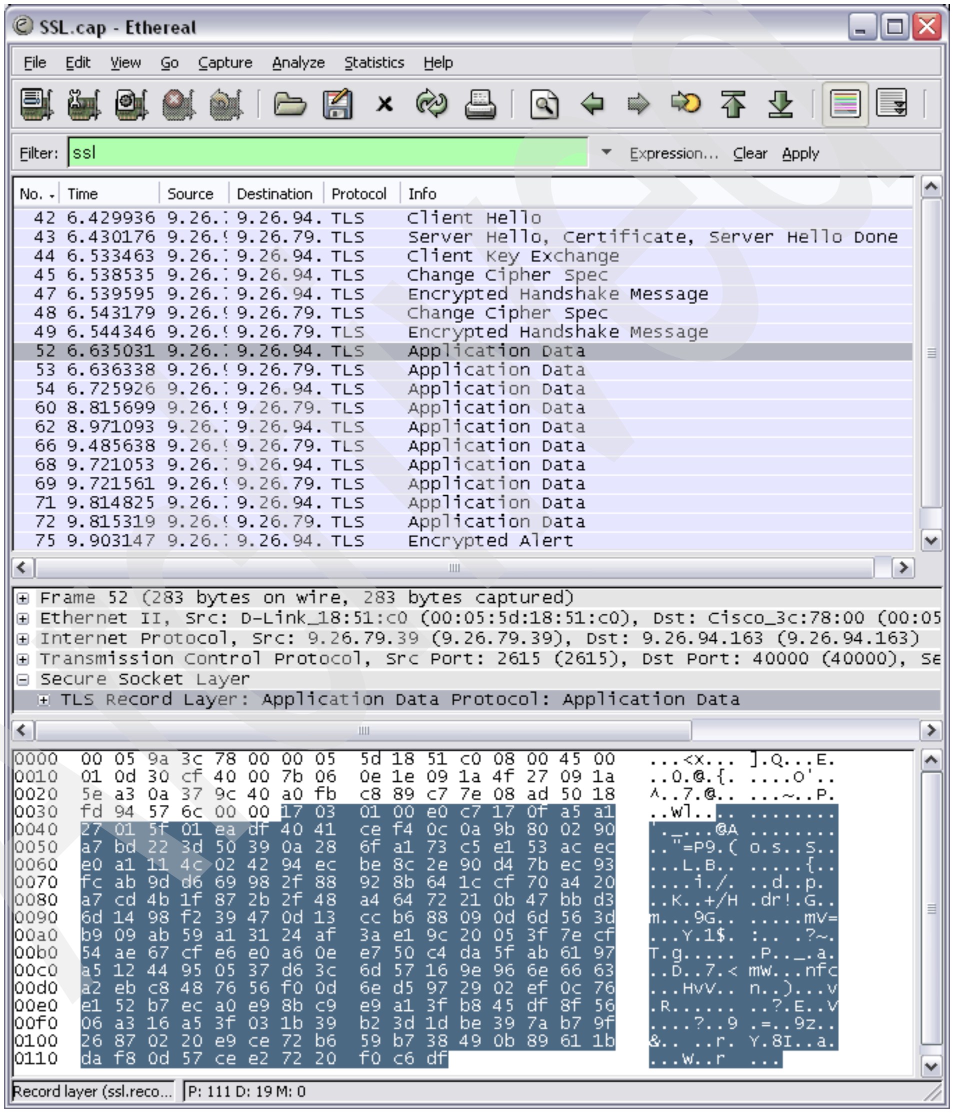
```
IBMTrustManager: Exception accessing default truststore:
java.io.IOException: Keystore was tampered with, or password was
incorrect
Default SSL context init failed: java.security.KeyStoreException:
IBMTrustManager: Problem accessing trust store java.io.IOException:
Keystore was tampered with, or password was incorrect
use dummy SSLSocketFactory due to java.lang.RuntimeException: Default
SSL context init failed: IBMTrustManager: Problem accessing trust store
java.io.IOException: Keystore was tampered with, or password was
incorrect

Execute exception: com.ibm.db2.jcc.b.gm:
[jcc][t4][2043][11550][3.50.152] Exception java.net.SocketException:
Error opening socket to server localhost/127.0.0.1 on port 40,000 with
message: Unconnected sockets not implemented. ERRORCODE=-4499,
SQLSTATE=08001.
com.ibm.db2.jcc.b.gm: [jcc][t4][2043][11550][3.50.152] Exception
java.net.SocketException: Error opening socket to server
localhost/127.0.0.1 on port 40,000 with message: Unconnected sockets
not implemented. ERRORCODE=-4499, SQLSTATE=08001
at com.ibm.db2.jcc.b.wc.a(wc.java:283)
at com.ibm.db2.jcc.t4.xb.a(xb.java:354)
at com.ibm.db2.jcc.t4.xb.<init>(xb.java:73)
at com.ibm.db2.jcc.t4.a.w(a.java:260)
at com.ibm.db2.jcc.t4.b.a(b.java:1775)
at com.ibm.db2.jcc.b.eb.a(eb.java:452)
at com.ibm.db2.jcc.b.eb.<init>(eb.java:417)
at com.ibm.db2.jcc.t4.b.<init>(b.java:296)
at
com.ibm.db2.jcc.DB2DataSource.getSimpleConnection(DB2DataSource.java:14
0)
at com.ibm.db2.jcc.DB2DataSource.getConnection(DB2DataSource.java:116)
at com.ibm.db2.jcc.DB2DataSource.getConnection(DB2DataSource.java:91)
at SSL.main(SSL.java:41)

Caused by: java.net.SocketException: Unconnected sockets not
implemented
at javax.net.SocketFactory.createSocket(SocketFactory.java:2)
at com.ibm.db2.jcc.t4.z.run(z.java:37)
at
java.security.AccessController.doPrivileged(AccessController.java:242)
at com.ibm.db2.jcc.t4.xb.a(xb.java:344)
... 10 more
```

Network traffic analyze

When a SSL connection is established, all the network traffic between the server and the client is encrypted. Only the server and the client can understand the communication, as they have the encryption key. This key is negotiated during initial SSL communication setup and it is always different for each session. It is not possible to see or get any data by capturing network traffic between the server and client. The communication is secure. Figure 7-10 shows captured network traffic secured by SSL, where everything is encrypted.


SSL.cap - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter: ssl
Expression... Clear Apply

```
No. - Time      Source  Destination Protocol Info
 42 6.429936 9.26.: 9.26.94. TLS     Client Hello
 43 6.430176 9.26.! 9.26.79. TLS     Server Hello, Certificate, Server Hello Done
 44 6.533463 9.26.: 9.26.94. TLS     Client Key Exchange
 45 6.538535 9.26.: 9.26.94. TLS     Change Cipher Spec
 47 6.539595 9.26.: 9.26.94. TLS     Encrypted Handshake Message
 48 6.543179 9.26.! 9.26.79. TLS     Change Cipher Spec
 49 6.544346 9.26.! 9.26.79. TLS     Encrypted Handshake Message
 52 6.635031 9.26.: 9.26.94. TLS     Application Data
 53 6.636338 9.26.! 9.26.79. TLS     Application Data
 54 6.725926 9.26.: 9.26.94. TLS     Application Data
 60 8.815699 9.26.! 9.26.79. TLS     Application Data
 62 8.971093 9.26.: 9.26.94. TLS     Application Data
 66 9.485638 9.26.! 9.26.79. TLS     Application Data
 68 9.721053 9.26.: 9.26.94. TLS     Application Data
 69 9.721561 9.26.! 9.26.79. TLS     Application Data
 71 9.814825 9.26.: 9.26.94. TLS     Application Data
 72 9.815319 9.26.! 9.26.79. TLS     Application Data
 75 9.903147 9.26.: 9.26.94. TLS     Encrypted Alert
```

```
⊞ Frame 52 (283 bytes on wire, 283 bytes captured)
⊞ Ethernet II, Src: D-Link_18:51:c0 (00:05:5d:18:51:c0), Dst: Cisco_3c:78:00 (00:05
⊞ Internet Protocol, Src: 9.26.79.39 (9.26.79.39), Dst: 9.26.94.163 (9.26.94.163)
⊞ Transmission Control Protocol, Src Port: 2615 (2615), Dst Port: 40000 (40000), Se
⊟ Secure Socket Layer
   ⊞ TLS Record Layer: Application Data Protocol: Application Data
```

```
0000  00 05 9a 3c 78 00 00 05  5d 18 51 c0 08 00 45 00   ...<x... ].Q...E.
0010  01 0d 30 cf 40 00 7b 06  0e 1e 09 1a 4f 27 09 1a   ..0.@.{. ....O'..
0020  5e a3 0a 37 9c 40 a0 fb  c8 89 c7 7e 08 ad 50 18   ^..7.@.. ...~..P.
0030  fd 94 57 6c 00 00 17 03  01 00 e0 c7 17 0f a5 a1   ..Wl.... ........
0040  27 01 5f 01 ea df 40 41  ce f4 0c 0a 9b 80 02 90   '._...@A ........
0050  a7 bd 22 3d 50 39 0a 28  6f a1 73 c5 e1 53 ac ec   .."=P9.( o.s..S..
0060  e0 a1 11 4c 02 42 94 ec  be 8c 2e 90 d4 7b ec 93   ...L.B.. .....{..
0070  fc ab 9d d6 69 98 2f 88  92 8b 64 1c cf 70 a4 20   ....i./. ..d..p.
0080  a7 cd 4b 1f 87 2b 2f 48  a4 64 72 21 0b 47 bb d3   ..K..+/H .dr!.G..
0090  6d 14 98 f2 39 47 0d 13  cc b6 88 09 0d 6d 56 3d   m...9G.. .....mV=
00a0  b9 09 ab 59 a1 31 24 af  3a e1 9c 20 05 3f 7e cf   ...Y.1$. :.. .?~.
00b0  54 ae 67 cf e6 e0 a6 0e  e7 50 c4 da 5f ab 61 97   T.g..... .P.._.a.
00c0  a5 12 44 95 05 37 d6 3c  6d 57 16 9e 96 6e 66 63   ..D..7.< mW...nfc
00d0  a2 eb c8 48 76 56 f0 0d  6e d5 97 29 02 ef 0c 76   ...HvV.. n..)...v
00e0  e1 52 b7 ec a0 e9 8b c9  e9 a1 3f b8 45 df 8f 56   .R...... ..?.E..V
00f0  06 a3 16 a5 3f 03 1b 39  b2 3d 1d be 39 7a b7 9f   ....?..9 .=..9z..
0100  26 87 02 20 e9 ce 72 b6  59 b7 38 49 0b 89 61 1b   &.. ..r. Y.8I..a.
0110  da f8 0d 57 ce e2 72 20  f0 c6 df                  ...W..r  ...
```

Figure 7-10 Network traffic encrypted using SSL

7.3 Encrypting data at rest

A database stores our digital information and holds our most critical data assets. You should consider giving additional protection to sensitive information such as storing the password in the database. DB2 offers an encryption capability that allows you to encrypt data at the column level to secure the sensitive data.

7.3.1 Column-level encryption

DB2 security control techniques offer secure methods to protect data within the database. For column-level encryption, DB2 provides built-in encryption and decryption functions, ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR, and GETHIN.

ENCRYPT

The ENCRYPT SQL scalar function encrypts data using a password-based encryption method. The encryption algorithm used is RC2 block cipher with padding and the secret encryption key is MD5 hash value derived from the password. Figure 7-11 describes the syntax for the ENCRYPT statement.

```
>>-ENCRYPT------------------------------------------------------->

>--(--data-string-expression--+----------------------------------------------------------------+--)-
><
                              '-,--password-string-expression--+----------------------------+-'
                                                               '-,--hint-string-expression-'
```

Figure 7-11 Encrypt statement.

The ENCRYPT function takes a value of type CHAR, VARCHAR and stores it as VARCHAR FOR BIT DATA. The length of the encrypted value is determined using the following formula:

```
Length of input value + 8 bytes + roundup to next 8 bytes boundary + 32
bytes for password hint (optional)
```

The password value used for encryption can be either an explicit password string expression or an assigned value determined by setting the SET ENCRYPTION PASSWORD statement. It must be a CHAR or a VARCHAR value of at least six bytes and not greater than 127 bytes in size. The function also allows you to specify a password hint that is embedded in the encrypted data. The password hint is an expression string of CHAR or VARCHAR limited to 32 bytes.

The ENCRYPT function can only encrypt columns with the following data types:

- CHAR FOR BIT DATA.
- VARCHAR FOR BIT DATA.

Example 7-14 shows how to encrypt a column in a table called employee using the SET ENCRYPTION PASSWORD statement to set an encrypted password for the session. It is important to know that the password set by the ENCRYPTION PASSWORD statement is only used during the DB2 session connecting to the database and it does not effect new DB2 sessions connecting to the database. For this reason, password management is the responsibility of the data owners and should be managed responsibly. Once the data is encrypted, only the password used to encrypt it can be used to decrypt it.

Example 7-14 ENCRYPT function with SET ENCRYPTION PASSWORD statement

```
db2 => CREATE TABLE emloyee (SSN varchar(24) FOR BIT DATA)
DB20000I  The SQL command completed successfully.

db2 => SET ENCRYPTION PASSWORD = 'testpwd'
DB20000I  The SQL command completed successfully.

db2 => INSERT INTO employee VALUES ENCRYPT('123-45-6789')
DB20000I  The SQL command completed successfully.

db2 => SELECT SSN FROM employee

SSN
---------------------------------------------------
x'0807F9FFB804A0D5EFA8F9CA02213C7EF517CEAF5CBD6717'

  1 record(s) selected.
```

As an alternative, instead of using the ENCRYPTION PASSWORD statement, you can explicitly pass the encryption password and a password hint, as shown in Example 7-15. The hint string expression is used to help the data owner remember the password. The password hint is embedded in the result along with the data value and the password. It is important to remember that passwords are case sensitive and the same case must be used when decrypting the data.

Example 7-15 ENCRYPT function using explicit password and hint

```
db2 => CREATE TABLE emloyee (SSN varchar(48) FOR BIT DATA)
DB20000I  The SQL command completed successfully.

db2 => INSERT INTO employee VALUES ENCRYPT('123456789', 'testpwd', 'mypasswd')
DB20000I  The SQL command completed successfully.
```

```
db2 => SELECT SSN FROM employee

SSN
--------------------------------------------------------------------------------
x'088E720AB804A0D56D7970617373776F7264E6CACE96650F59D150CB4DD7AF59A34C'

  1 record(s) selected.
```

The hint string can be retreived using the GETHINT built-in function, as shown in Example 7-16.

Example 7-16 Retrieving password hint using GETHINT function

```
db2 => SELECT GETHINT (ssn) password_hint FROM employee

PASSWORD_HINT
--------------------------------
mypasswd

  1 record(s) selected.
```

Since the password hint is embedded in the result along with the data value and the password, the column size must be sufficient to store the entire encrypted value. Otherwise, DB2 fails with SQL0117N when attempting to insert the value into the table, as shown in Example 7-17.

Example 7-17 Insufficient password column length

```
db2 => CREATE TABLE employee (SSN varchar(24) FOR BIT DATA)
DB20000I  The SQL command completed successfully.

db2 => INSERT INTO employee values ('123-45-6789', 'testpwd', 'mypasswd')
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0117N  The number of values assigned is not the same as the number of
specified or implied columns or variables.  SQLSTATE=42802
```

DECRYPT_BIN and DECRYPT_CHAR

Once the data is encrypted, the only way to decrypt it is by using the correct password. Use the DECRYPT_BIN and DECRYPT_CHAR functions only to decrypt data stored using the ENCRYPT function.

Figure 7-12 provides the syntax for both scalar functions.

```
>>-+-DECRYPT_BIN--+--------------------------------------------->
   '-DECRYPT_CHAR-'

>--(--encrypted-data--+-------------------------------+--)-----><
                      '-,--password-string-expression-'
```

Figure 7-12   Syntax for DECRYPT_BIN and DECRYPT_CHAR functions

The encrypted data must be a column of type CHAR FOR BIT DATA or VARCHAR FOR BIT DATA that was encrypted using the ENCRYPT function.

The DECRYPT_BIN function returns a VARCHAR FOR BIT DATA (binary format). The DECRYPT_CHAR function returns a VARCHAR value. The actual length of the value returned by the functions matches the length of the original string that was encrypted. Therefore, if the encrypted value included a hint, the hint is not returned by the functions.

Example 7-18 shows how to use decrypt functions to retrieve the encrypted data. Only the data portion is returned. Password and hit are not returned.

Example 7-18   DECRYPT_CHAR output

```
db2 => CREATE TABLE emloyee (SSN varchar(48) FOR BIT DATA)
DB20000I  The SQL command completed successfully.

db2 => INSERT INTO employee VALUES ENCRYPT('123456789', 'TESTPWD', 'mypasswd')
DB20000I  The SQL command completed successfully.

db2 => SELECT DECRYPT_CHAR (SSN, 'testpwd') SSN FROM employee

SSN
----------------------------------------
123-45-6789

  1 record(s) selected.

db2 => SELECT DECRYPT_BIN (SSN, 'testpwd') SSN FROM CUST

SSN
--------------------------------------------------------------------------------
x'3132332D34352D36373839'

  1 record(s) selected.
```

Note: Use the SET ENCRYPTION PASSWORD statement to set an encrypted password for the user session.

Considerations

The main advantage of using column-level encryption is protecting sensitive information or data from attacks against the data within the tables. However, they cannot protect the tables data files stored on the operating system from privileged users who have a significant amount of systems access. Without careful planning, column-level encryption can have a negative impact on the database performance and how data is accessed. For example:

- Column-level encryption drastically diminishes the effectiveness of indexes, particularly for range queries.
- Column-level encryption does not prevent users with SYSADM or DBADM database authorities from deleting the data or dropping the tables hosting the encrypted data.
- Column-level encryption is easily affected by changes to the underlying table, for example, altering the column types and altering objects that reference them such as referential constraints.
- Column-level encryption is not transparent to the application.
- Column-level encryption cannot be used to protect against attacks on log files such as audit, import, and export files.
- Column-level encryption keys are stored in files in the OS and can be easily compromised, eliminating the benefits of encryption.

From a performance perspective, column-level encryption and decryption reduce the benefits of caching data rows:

- Column-level encryption adds significant overhead, not because of cryptography but because of the manner in which column-level encryption *inserts itself* in the database.
- Column-level encryption reduces the performance benefits of caching data rows, since the rows must first be decrypted.

For these reasons, IBM offers a solution called IBM Database Encryption Expert to complement DB2 column-level encryption. With both solutions you are able to specify when and who can have access to sensitive data and files and also protect offline data files such as database backups and archived files.

We discuss Database Encryption Expert more detail in Chapter 8, "IBM security solutions" on page 179.

8

IBM security solutions

One long-standing commitment of IBM is to provide its clients with solutions to meet evolving security needs and to protect one of their most valuable assets, data. IBM solutions of regulatory compliance for DB2 Linux, UNIX, and Windows give you the capabilities to encrypt your data by enforcing policy-based data security to protect sensitive data in both online and offline environments, offer detailed auditing capabilities to help security administrators and auditors deliver accurate and timely audit reports, and help you with the routine task of creating test data.

In this chapter we introduce different IBM security solutions:

- IBM Database Encryption Expert
- DB2 Audit Management Expert
- IBM Optim - Enterprise Data Management

8.1 IBM Database Encryption Expert

IBM Database Encryption Expert for Linux, UNIX, and Windows provides additional protection to safeguard DB2 sensitive information using a transparent, high-speed encryption methodology. It provides comprehensive and detailed auditing reports, context-aware access control, and advanced host and application integrity protection. This tool transparently interoperates with DB2 to protect data in both online and offline environments providing centralized policy and key administration. Its architecture enables easy-to-manage, flexible mandatory access control, and separation of policy management from host management, enforcing the separation of duties between security administration and system administration.

There are many features offered by the Database Encryption Expert. The key features include:

- Centralized key and policy management that allows you to control which keys are used, and where and when restores are allowed.
- Authentication certificates are generated automatically during software installation.
- Protects sensitive information without disrupting data management.
- Mutual agent/server authentication that occurs during policy evaluation.
- Does not require changes to current DB2 backup and restore processes.
- Supports native DB2 data compression.
- Supports high performance encryption methodology.
- Supports standard AES 128/256 encryption.
- The DB2 DBA on the system running the agent system can create backups and perform restores of static data-at-rest and active online data with Database Encryption Expert protection without making drastic changes.
- Audits and tracks encryption usage to meet compliance requirements.
- Provides high availability (failover support).
- Provides privileged users with the ability to view and manage metadata while at the same time limiting access to private or sensitive data.

8.1.1 Architecture

IBM Database Encryption Expert is made up of an agent host and a security server. A security server is a system with a Database Encryption Expert security server software package installed. The server must be installed on a dedicated machine to provide maximum security and control. An agent host is a system with a Database Encryption Expert agent software package installed.

Figure 8-1 shows the major components making up the Database Encryption Expert.

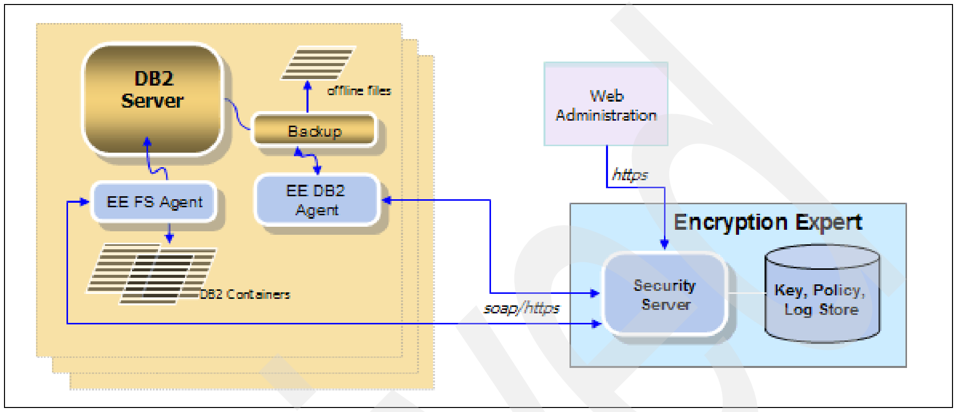


Figure 8-1 Database Encryption Expert architeture.

The communication between server and agent, as well as server and server, is configured to use Security Socket Layer (SSL). Encrypted data is ineffective if the key for decrypting is intercepted as transmitted over the network. For this reason, SSL authentication is used to encrypt all transmitted data between the agents and the security server as well as between different Database Encryption Expert servers. The server implements X.509 digital security certificates for agent-to-server and server-to-server communication. The certificate is created or issued by the server. It is signed using the Database Encryption Expert private key. The security certificate is required to configure the Database Encryption Expert agents to be able to communicate properly with the Database Encryption Expert server.

The Database Encryption Expert security server is the heart of the Database Encryption Expert system. This is where policies and encryption keys are created and stored. It is the centralized key management repository for the Database Encryption Expert system. The encryption keys are maintained on the security server and not on the agent host systems that are being protected. It is an System x™ server running RedHat Linux with a Web server and an embedded DB2 Express as part of it. The data stored within the database along with the DB2 backup API are used to create DB2 database backups (offline) and file system (online), running on remote agent systems. The Database Encryption Expert security server supports all DB2 database backup functions that are normally performed on the agent system including the native DB2 database compression algorithm. Most importantly, the changes a DBA has to make to the DB2 backup and restore processes and commands are almost negligible.

8.1.2 Protecting data

Database Encryption Expert provides data protection for DB2 LUW environments including the database files accessed from outside of DB2 by users with high operating system privileges. Database Encryption Expert offer the following data protection:

- Offline data protection

  Offline data protection is for the protection of backups. You can use Database Encryption Expert to encrypt and compress database online and offline backups.

- Online data protection

  Online data protection protects files in the live DB2 system, such as containers, logs, and so on. You can encrypt these online database files and give the access to privileged users.

Offline data protection

Once Database Encryption Expert is installed and configured properly, you can incorporate the Database Encryption Expert data protection into your database backup and restore procedures. The security administrator has to create a security policy that permits you to perform backup and restore operations. The security policy defines the encryption method and where encrypted backups can be restored.

The changes a DBA has to make to the DB2 backup and restore procedures and commands are trivial. The key change is on the DB2 backup command. You specify the compression library provided by Database Encryption Expert in the compression library option. This is Database Encryption Expert's *hook* into DB2 backup and restore. DB2 thinks it is dealing with a compression library. It does not know, nor does it need to know, that the library is being used to protect the backup through encryption rather than just compressing it. If you want both compression and encryption in the backup, you can specify your choice in the Database Encryption Expert policies.

Policy evaluation

The backup and restore requests are evaluated by the Database Encryption Expert security server before the agent can perform the requested operation. The requests are evaluated by using agent system parameters and security administrator defined policy constraints.

Figure 8-2 provides a general overview of the Database Encryption Expert policy evaluation process.

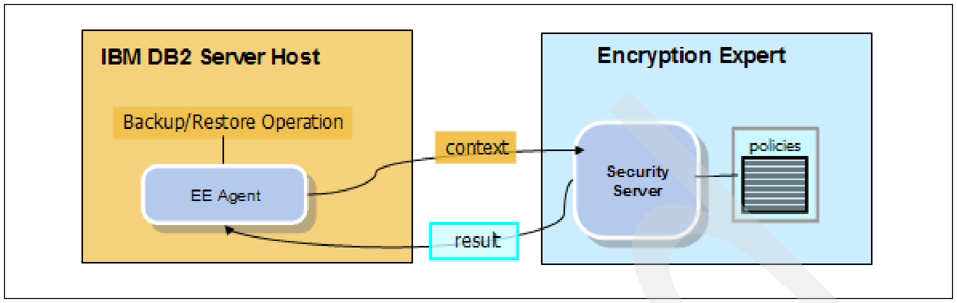


Figure 8-2 Database Encryption Expert policy evaluation

The concept of policy evaluation for offline and online backups is basically the same. In both cases, the Database Encryption Expert backup and restore libraries on the system get loaded for communication with the security server. During a backup or restore the following communications take place:

- The agent sends context information about the operation to the security server.
- The server searches for the first policy whose attributes match the context.
- The server evaluates the matching policy rules and returns either a permit or a deny.
- A permit also returns information such as the required encryption key that the agent needs to enforce the policy.
- If no policy matches the context, the operation is denied by default.

Back up DB2 database

To create an online or offline DB2 database backup with Database Encryption Expert encryption, use the regular DB2 online or offline backup procedure and commands. In the database backup command, specify the full path name to the Database Encryption Expert encryption library in the COMPRESS COMPRLIB option.

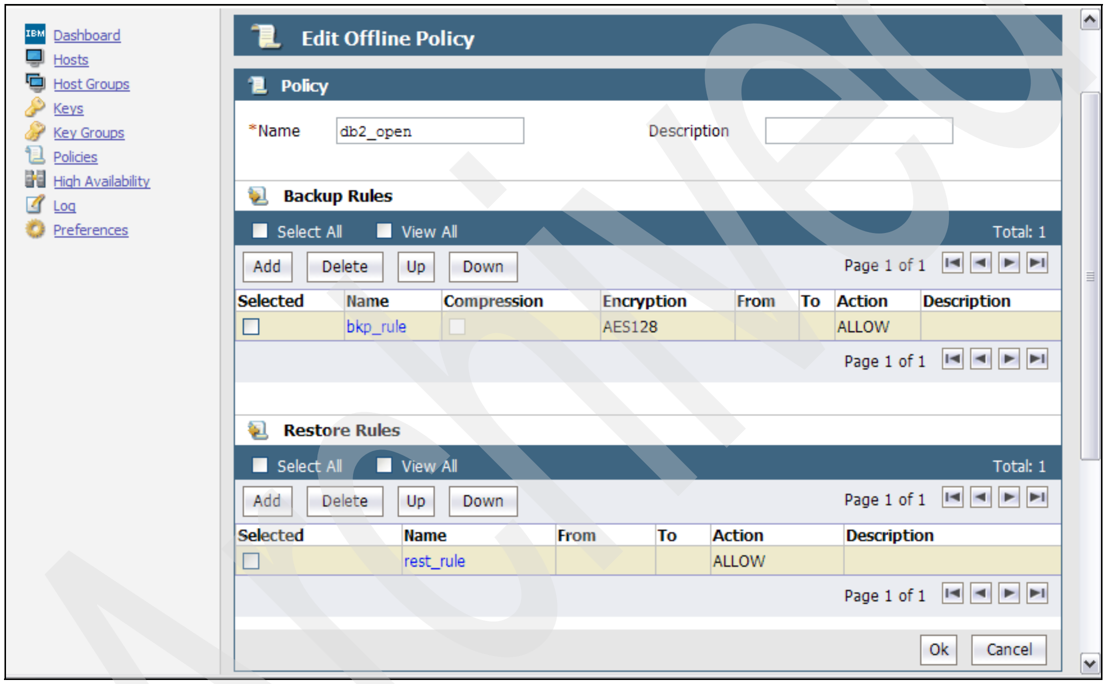
When the Database Encryption Expert agent library is used during a backup or restore operation, it contacts the Database Encryption Expert Security Server to determine whether the operation is allowed. The server makes this determination by matching a security policy and then evaluating the policy’s rules.

Example 8-1 shows how to back up the DB2 database with the Database Encryption Expert.

Example 8-1 Database backup with Database Encryption Expert

```
db2 BACKUP DATABASE sample TO /db2back COMPRESS COMPRLIB
/opt/IBM/DB2TOOLS/LUWEncryptionExpert/agent/db2/lib/libeetdb2.so
```

In the security policy, you can specify whether you want encryption, compression, or both. See Figure 8-3. Compression is based upon the *zlib* algorithm.



*Name: db2_open
Description:

Backup Rules

Selected	Name	Compression	Encryption	From	To	Action	Description
☐	bkp_rule	☐	AES128			ALLOW	

Restore Rules

Selected	Name	From	To	Action	Description
☐	rest_rule			ALLOW	

Figure 8-3 Specify backup security policy

You can utilize a script and an alias to append COMPRLIB automatically such that user need not specify the COMPRLIB and backups are governed by Database Encryption Expert policies. We provide a simple script and the alias as follows:

1. Create a script to append the COMPRLIB automatically for the DB2 backup command. Example 8-2 shows a sample shell script.

Example 8-2 db.sh

```
#!/bin/sh
ocmd=$*
comprs="COMPRESS COMPRLIB
/opt/IBM/DB2TOOLS/LUWEncryptionExpert/agent/db2/lib/libeetdb2.so"
dcmd=`echo $ocmd|awk '$1 !~ /[Bb][Aa][Cc][Kk][Uu][Pp]/ {print $0}'`
if [ "$dcmd" != '' ] ; then
  exec db2 "$ocmd"
else
    dcmd=`echo $ocmd|awk '/[Cc][Oo][Mm][Pp][Rr][Oo][Pp][Tt][Ss]/
{print $0}'`
    if [ "$dcmd" != '' ] ; then
    dcmd=`echo $ocmd|awk -v comprs="$comprs"
'sub(/[Cc][Oo][Mm][Pp][Rr][Oo][Pp][Tt][Ss]/,comprs" &")1' `
    exec db2 "$dcmd"
    else
    dcmd=`echo $ocmd|awk
'/[Uu][Tt][Ii][Ll]_[Ii][Mm][Pp][Aa][Cc][Tt]_[Pp][Rr][Ii][Oo][Rr][Ii]
[Tt][Yy]/ {print $0}'`
    if [ "$dcmd" != '' ] ; then
      dcmd=`echo $ocmd|awk -v comprs="$comprs"
'sub(/[Uu][Tt][Ii][Ll]_[Ii][Mm][Pp][Aa][Cc][Tt]_[Pp][Rr][Ii][Oo][Rr]
[Ii][Tt][Yy]/,comprs" &")1' `
      exec db2 "$dcmd"
    else
      dcmd=`echo $ocmd|awk '/[Ee][Xx][Cc][Ll][Uu][Dd][Ee]
[Ll][Oo][Gg][Ss]/ {print $0}'`
      if [ "$dcmd" != '' ] ; then
        dcmd=`echo $ocmd|awk -v comprs="$comprs"
'sub(/[Ee][Xx][Cc][Ll][Uu][Dd][Ee] [Ll][Oo][Gg][Ss]/,comprs" &")1'`
        exec db2 "$dcmd"
      else
        dcmd=`echo $ocmd|awk '/[Ii][Nn][Cc][Ll][Uu][Dd][Ee]
[Ll][Oo][Gg][Ss]/ {print $0}'`
        if [ "$dcmd" != '' ] ; then
            dcmd=`echo $ocmd|awk -v comprs="$comprs"
'sub(/[Ii][Nn][Cc][Ll][Uu][Dd][Ee] [Ll][Oo][Gg][Ss]/,comprs" &")1'`
```

```
            exec db2 "$dcmd"
        else
          dcmd=`echo $ocmd|awk '/[Ww][Ii][Tt][Hh][Oo][Uu][Tt]
[Pp][Rr][Oo][Mm][Pp][Tt][Ii][Nn][Gg]/ {print $0}'`
          if [ "$dcmd" != '' ] ; then
              dcmd=`echo $ocmd|awk -v comprs="$comprs"
'sub(/[Ww][Ii][Tt][Hh][Oo][Uu][Tt]
[Pp][Rr][Oo][Mm][Pp][Tt][Ii][Nn][Gg]/,comprs" &")1'`
              exec db2 "$dcmd"
          else
            exec db2 "$ocmd" $comprs
          fi
        fi
      fi
    fi
  fi
fi
```

2. Create an operating system alias for the **db2** command:

   ```
   alias db2='/home/db2inst1/bin/db2.sh db2'
   ```

Now, when the user runs a command like:

```
db2 backup database prod
```

it is automatically expanded and run as:

```
db2 backup database prod COMPRESS COMPRLIB
/opt/IBM/DB2TOOLS/LUWEncryptionExpert/agent/db2/lib/libeetdb2.so
```

Restoring a DB2 database

To restore a DB2 database you must have configured an agent host to have access to the appropriate private keys. If the system is not configured with agent software, the restore completes but the data is encrypted and unusable. Otherwise, there are no changes required to perform the database restore.

Online data protection

The Database Encryption Expert online data protection feature can be used to protect critical DB2 files such as tablespace container files, logs, audit files, and imported/extracted data files. These files are exposed to many threats mainly by users with high operating system privileges. You can create encryption security policies to protect DB2 database files and folders. Because they serve different purposes, policies for file system (online) protection and policies for database backup (offline) protection are different, and they are handled differently by the agents to which the policies are applied.

The Database Encryption Expert file agent is a process that is always running in the background. It can administer all applications, processes, and files that start after the agent is started. The Database Encryption Expert encryption engine encrypts just the file contents, and leaves the file metadata intact. That is, you do not have to decrypt an encrypted file to see its name, timestamps, file type, and so on. This allows data management applications to perform their functions without exposing the file contents.

Figure 8-4 illustrates the Database Encryption Expert Agent within the system architecture.

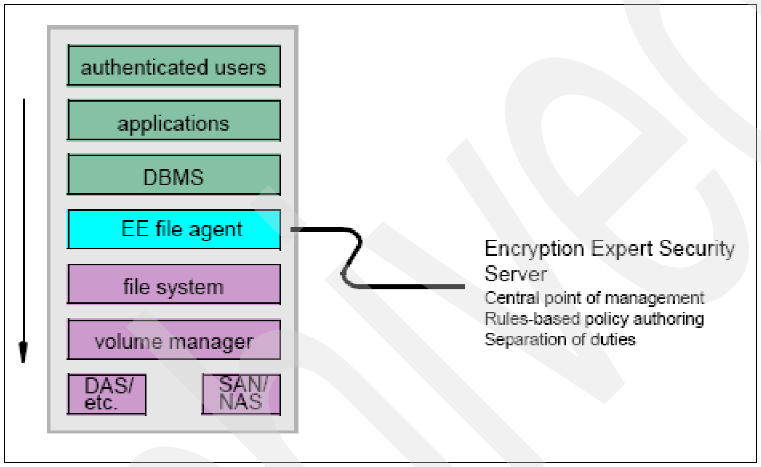


Figure 8-4 Database Encryption Expert Agent within the system architecture

The online policy comprises security rules and sometimes key rules. The security rule defines the users or user groups authorized to have specified access to specific files and directory paths for a designated period of time. In short, it defines who is accessing data, what they are doing with the data, where the data is located, when the security rule is applicable, and how the data can be accessed.

Figure 8-5 shows the online policy composer for adding a new policy.

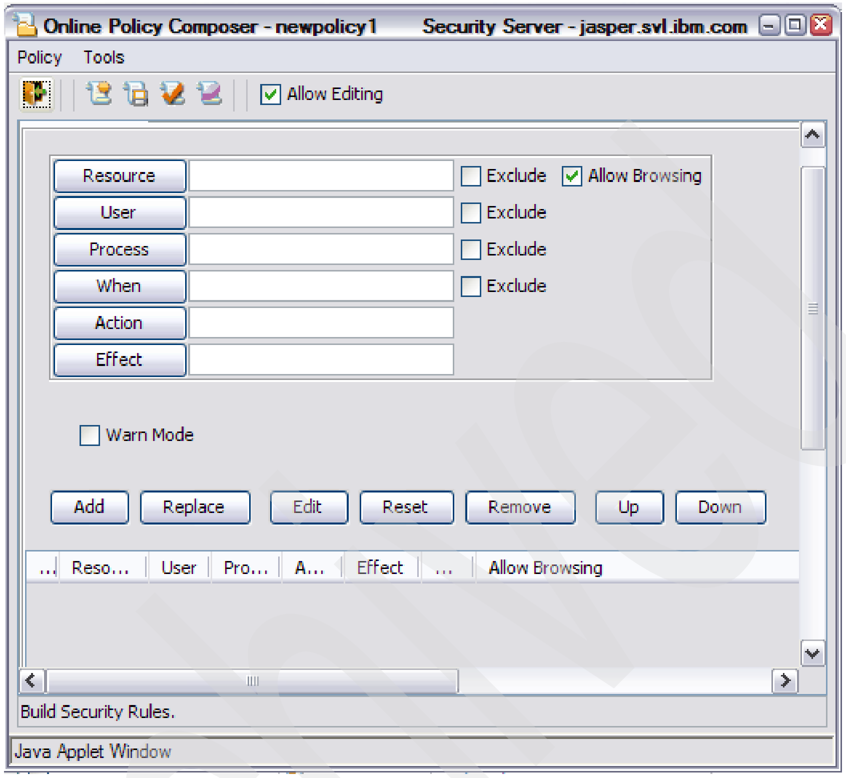
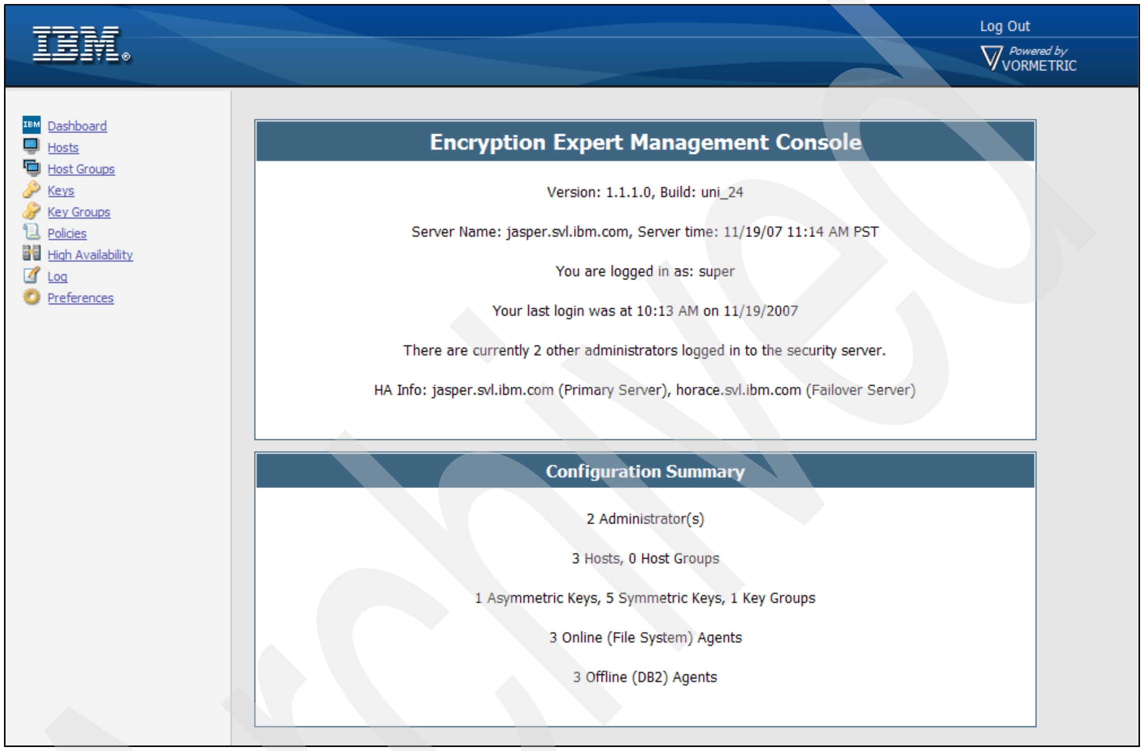


Figure 8-5 Online policy composer

8.1.3 Administration

The Database Encryption Expert provides centralized administration for managing policies, keys, hosts, users, and logs through a Web-based GUI, which is also referred to as *management console*. Figure 8-6 shows the management console dashboard.



Encryption Expert Management Console

Version: 1.1.1.0, Build: uni_24

Server Name: jasper.svl.ibm.com, Server time: 11/19/07 11:14 AM PST

You are logged in as: super

Your last login was at 10:13 AM on 11/19/2007

There are currently 2 other administrators logged in to the security server.

HA Info: jasper.svl.ibm.com (Primary Server), horace.svl.ibm.com (Failover Server)

Configuration Summary

2 Administrator(s)

3 Hosts, 0 Host Groups

1 Asymmetric Keys, 5 Symmetric Keys, 1 Key Groups

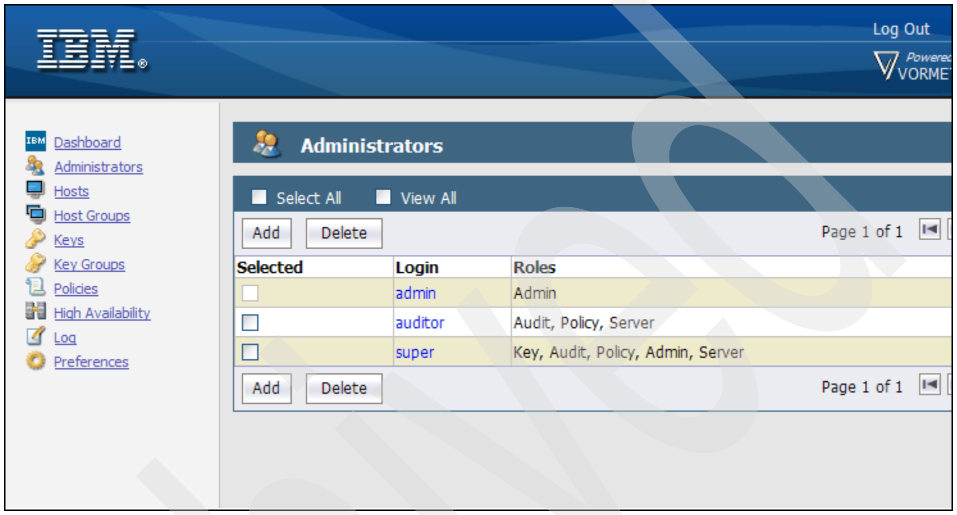
3 Online (File System) Agents

3 Offline (DB2) Agents

Figure 8-6 Database Encryption Expert dashboard

From the management console the security administrator can perform the following:

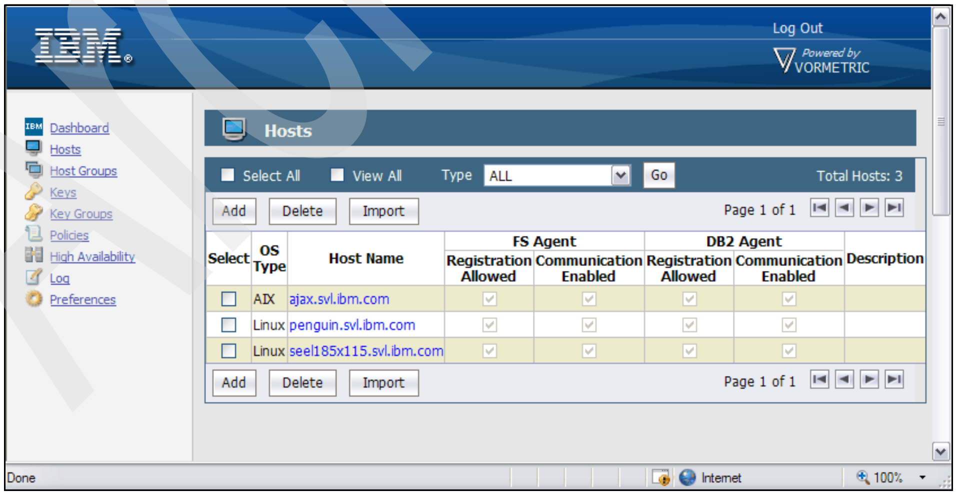
- Manage users including adding and removing user accounts and granting different roles such as administrator, audit, key, policy, and authorities to different users. Figure 8-7 shows the Administrator panel.



Selected	Login	Roles
☐	admin	Admin
☐	auditor	Audit, Policy, Server
☐	super	Key, Audit, Policy, Admin, Server

Figure 8-7 Adding new user account.

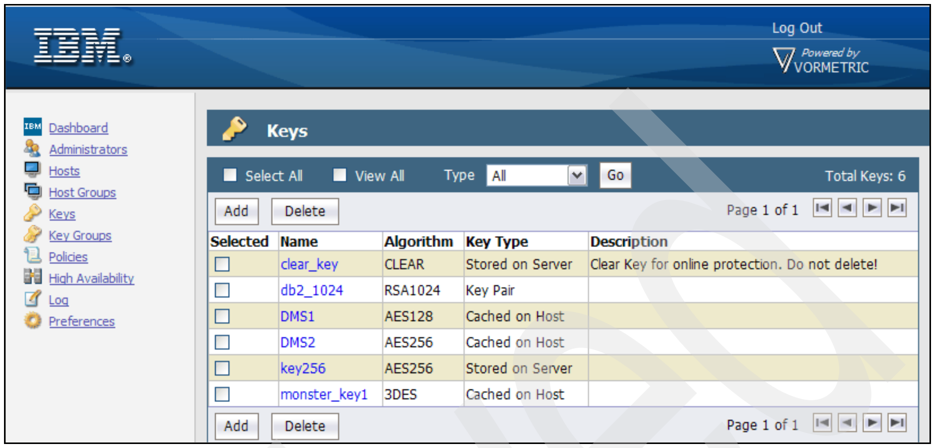
- Add, remove, and administrate hosts and host groups. Figure 8-8 shows the Hosts managing panel.



Select	OS Type	Host Name	FS Agent		DB2 Agent		Description
			Registration Allowed	Communication Enabled	Registration Allowed	Communication Enabled	
☐	AIX	ajax.svl.ibm.com	✓	✓	✓	✓	
☐	Linux	penguin.svl.ibm.com	✓	✓	✓	✓	
☐	Linux	seel185x115.svl.ibm.com	✓	✓	✓	✓	

Figure 8-8 Administrating hosts

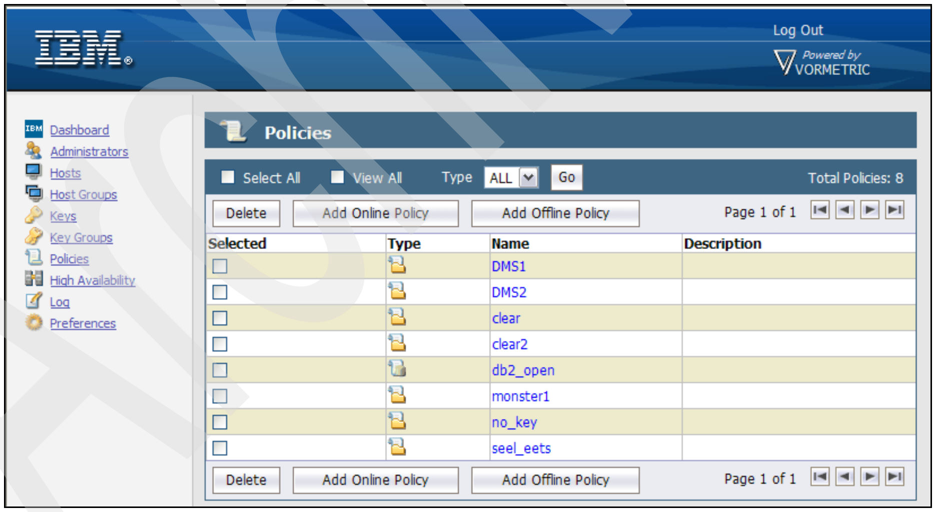
- Manage and create keys, affiliate keys, and key groups. Figure 8-9 shows the Key managing panel.



Selected	Name	Algorithm	Key Type	Description
☐	clear_key	CLEAR	Stored on Server	Clear Key for online protection. Do not delete!
☐	db2_1024	RSA1024	Key Pair	
☐	DMS1	AES128	Cached on Host	
☐	DMS2	AES256	Cached on Host	
☐	key256	AES256	Stored on Server	
☐	monster_key1	3DES	Cached on Host	

Figure 8-9 Administrating keys

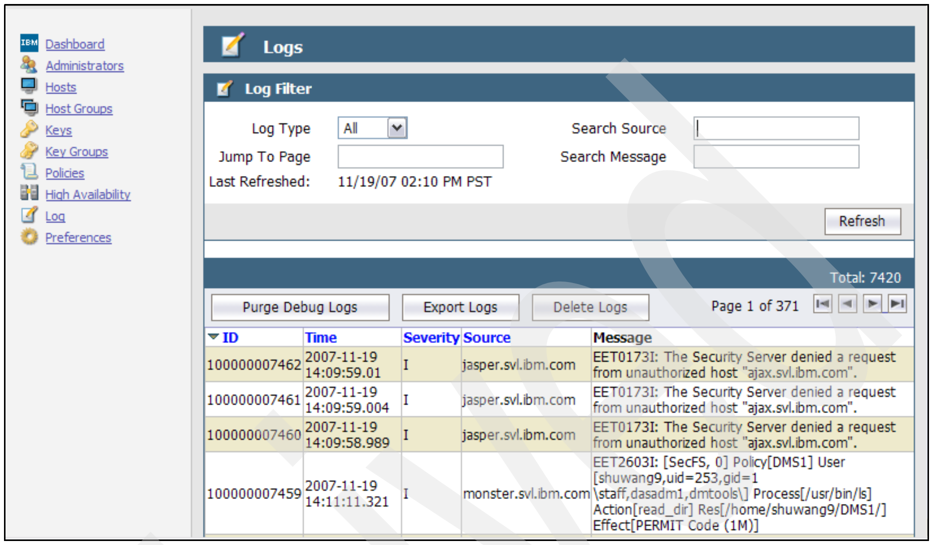
- Create and manage online and offline policies. Figure 8-10 shows the initial panel for managing policies.



Selected	Type	Name	Description
☐		DMS1	
☐		DMS2	
☐		clear	
☐		clear2	
☐		db2_open	
☐		monster1	
☐		no_key	
☐		seel_eets	

Figure 8-10 Administrating policies

- View logs. You can display the security server log and configure the events that it displays in the management interface. Figure 8-11 shows a sample log panel.



Log Type: All
Jump To Page:
Last Refreshed: 11/19/07 02:10 PM PST
Search Source:
Search Message:

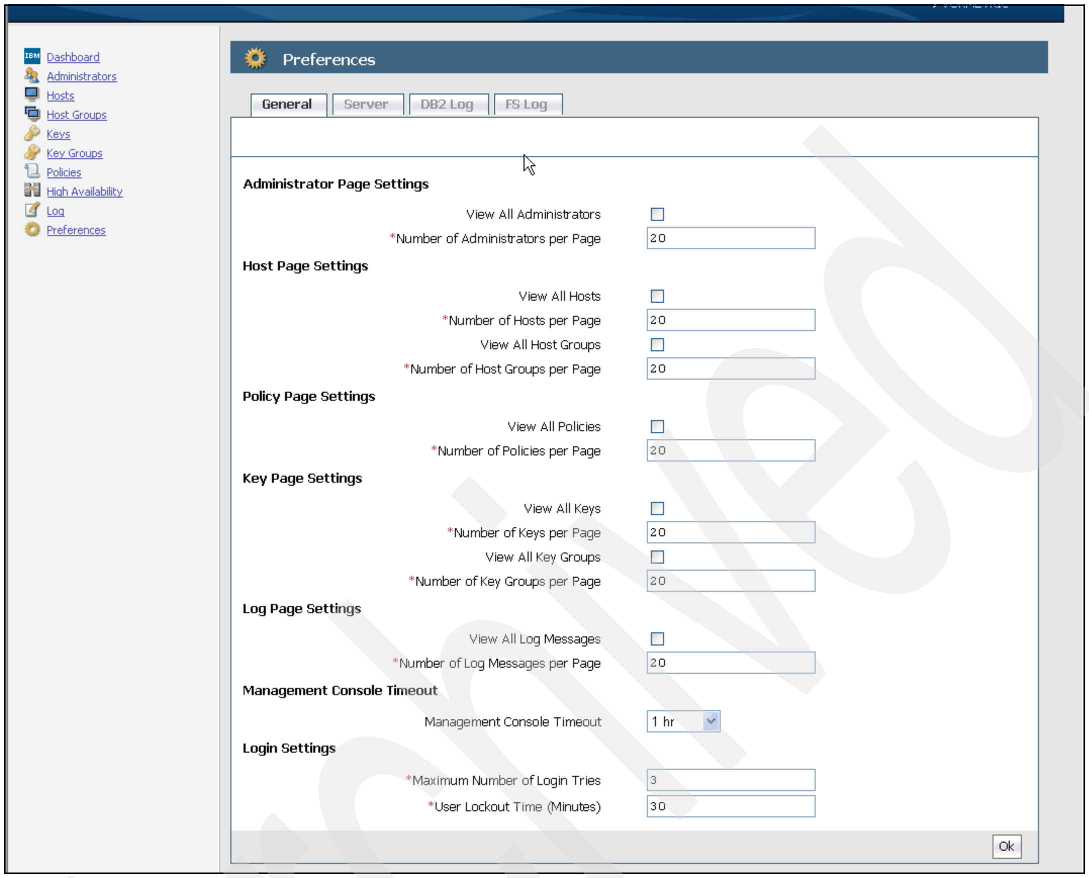
Total: 7420

Page 1 of 371

ID	Time	Severity	Source	Message
100000007462	2007-11-19 14:09:59.01	I	jasper.svl.ibm.com	EET0173I: The Security Server denied a request from unauthorized host "ajax.svl.ibm.com".
100000007461	2007-11-19 14:09:59.004	I	jasper.svl.ibm.com	EET0173I: The Security Server denied a request from unauthorized host "ajax.svl.ibm.com".
100000007460	2007-11-19 14:09:58.989	I	jasper.svl.ibm.com	EET0173I: The Security Server denied a request from unauthorized host "ajax.svl.ibm.com".
100000007459	2007-11-19 14:11:11.321	I	monster.svl.ibm.com	EET2603I: [SecFS, 0] Policy[DMS1] User [shuwang9,uid=253,gid=1 \staff,dasadm1,dmtools\] Process[/usr/bin/ls] Action[read_dir] Res[/home/shuwang9/DMS1/] Effect[PERMIT Code (1M)]

Figure 8-11 View logs

- Configure Web interface preferences, as shown in Figure 8-12.



Administrator Page Settings

View All Administrators	☐
*Number of Administrators per Page	20

Host Page Settings

View All Hosts	☐
*Number of Hosts per Page	20
View All Host Groups	☐
*Number of Host Groups per Page	20

Policy Page Settings

View All Policies	☐
*Number of Policies per Page	20

Key Page Settings

View All Keys	☐
*Number of Keys per Page	20
View All Key Groups	☐
*Number of Key Groups per Page	20

Log Page Settings

View All Log Messages	☐
*Number of Log Messages per Page	20

Management Console Timeout

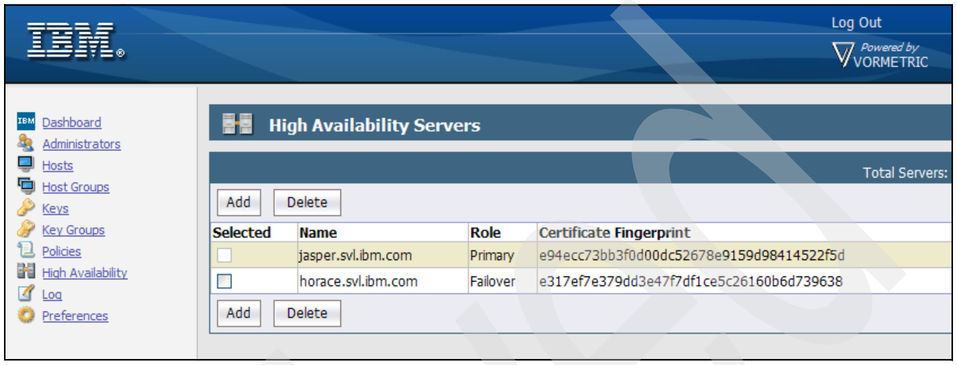
Management Console Timeout	1 hr

Login Settings

*Maximum Number of Login Tries	3
*User Lockout Time (Minutes)	30

Figure 8-12 Preferences

- Add a failover server designed to perform the backup and restore functions in case the primary security server becomes unavailable. Failover servers contain replicated data from the primary server with the same keys, policies, host information, and so on. Figure 8-13 shows the High Availability Servers interface.



Selected	Name	Role	Certificate Fingerprint
☐	jasper.svl.ibm.com	Primary	e94ecc73bb3f0d00dc52678e9159d98414522f5d
☐	horace.svl.ibm.com	Failover	e317ef7e379dd3e47f7df1ce5c26160b6d739638

Figure 8-13 Managing high availability servers

Conclusion

With DB2 access control methods and database Database Encryption Expert, organizations can maintain compliance with government regulations and legislative acts and industry regulations such as PCI, HIPPA, and GLBA, thus allowing them to ensure that private, intellectual, and confidential data is strongly protected, retain client confidence, avoid negative publicity, and maintain their competitive advantage in today’s market.

8.2 DB2 Audit Management Expert

The DB2 Audit Management Expert (DB2 AME) is a tool that offers comprehensive, detailed auditing capabilities to help database administrators, security administrators, and auditors deliver accurate, timely data and reports for use in auditing activities. This tool is targeted to be used by auditors, both internal and external, and has the following features:

- Provides centralized auditing tools to bring together information from many different sources into a correlated, coherent view.
- Enables auditors to collect, view, analyze, and report on data and save it into an audit repository.
- Empowers non-technical users to define collection criteria and automatically launch traces to collect data.

- Allows auditors to automatically generate their own reports and export the data into other applications such as Excel spreadsheets.
- Does *not* require auditors to be DB2-defined users within the monitored DB2 systems.
- Does *not* require the auditors to log on to the operating system where the monitored system is running.
- Does *not* require extensive interaction between the auditor and the system support personnel (DBA/SYS administrator).
- Auditor is *not* able to directly manipulate any DB2 resources.

The DB2 AME tool has two versions: DB2 Audit Management Expert for z/OS and DB2 Audit Management Expert for Multiplatform. This book covers the DB2 Audit Management Expert for Multiplatform only.

8.2.1 Overview

The DB2 Audit Management Expert operates in a client-server environment using a combination of one DB2 AME server and one or more DB2 AME agents. Reporting and administration is done using the DB2 AME clients user interface (UI).

Figure 8-14 illustrates the DB2 AME architecture. The administration and reporting user interfaces can communicate to any defined DB2 AME server. Each server, and associated audit repository, is standalone and does not interact with other DB2 AME servers and audit repositories.

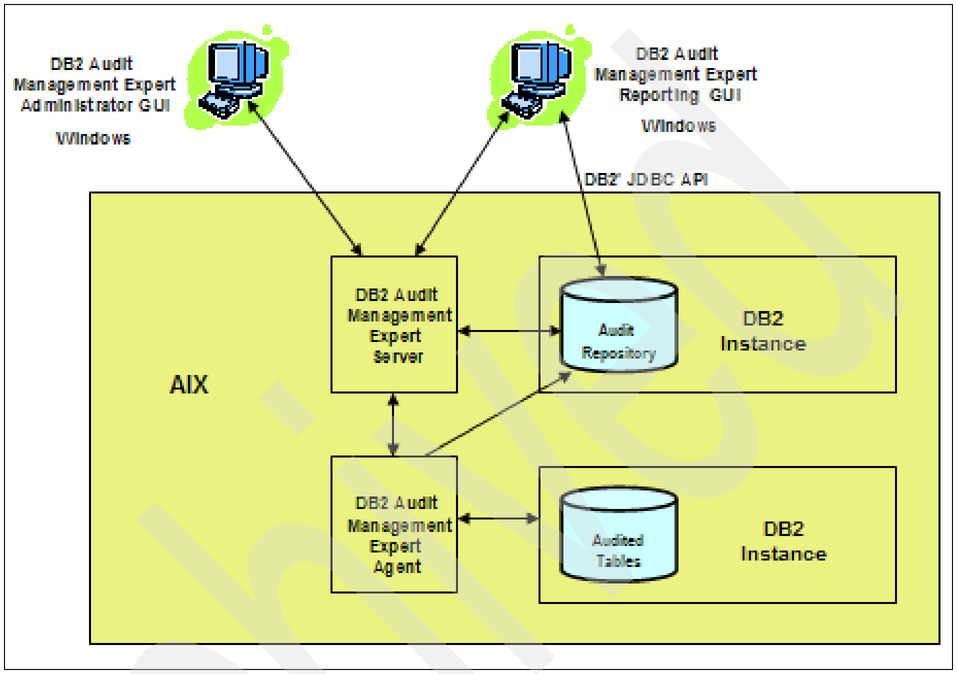

DB2 Audit Management Expert Administrator GUI
Windows
DB2 Audit Management Expert Reporting GUI
Windows
DB2' JDBC API
AIX
DB2 Audit Management Expert Server
Audit Repository
DB2 Instance
DB2 Audit Management Expert Agent
Audited Tables
DB2 Instance

Figure 8-14 DB2 Audit Management Expert architecture

An DB2 AME server can have multiple agents distributed over a connected network. However, we recommend that the DB2 AME server resides on the same AIX system as the DB2 AME repository.

At the time of writing, the current version of the DB2 Audit Management Expert for Multiplatform is 1.1. This version has these system requirements:

- DB2 AME server and agent
  - AIX 5L™ V5.2 or AIX 5L V5.3
  - DB2 8 FP9a or later, DB2 9 or later (only 64-bit)

Note: At the time of writing, the DB2 Audit Management Expert 1.1 does not support DB2 9.5

- DB2 AME UI clients
  - Windows XP
  - IBM Data Server Driver for JDBC and SQLJ 2.3.63 (minimum level)

> **Note:** This driver is a requirement for the reporting client only. It is available as part of DB2 8 FP7a or later. The DB2 AME does not include this driver. It needs to be present on the client system.

DB2 Audit Management Expert profiles

The profiles determine the DB2 AME behavior. The DB2 AME profiles are created and maintained through the DB2 AME Administration client.

Collection profile

A collection profile determines which objects, events, or users are audited, and how often these items are audited. The collected data can be filtered based on any combination of the following items:

- Schedule
- General audits:
  - All failed authorizations
  - Grants and revokes
  - Connect successes
  - Connect failures
- DB2 objects (tables, views, and packages)
- Events:
  - Tables and views
    - All events
    - Select
    - Insert
    - Update
    - Delete
    - Create, alter, and drop
  - Packages
    - All events
    - Execute
    - Bind
    - Control
    - Drop
    - Create

- Identity (AuthID, Workstation, or Execution ID)
- Applications

Authorization

The authorization profile describes which audit data, once collected, can be viewed by associated users or groups. A DB2 AME administrator can create authorizations and give permissions to allow an administrator to:

- View the data collected by the collection profile by means of the reporting user interface (read).
- Edit the authorization profile (write).

8.2.2 Setup

The DB2 AME setup consists of installing and configuring these four components:

- DB2 AME server
  a. Install the server using the **`./adh11serveraix`** command.
  b. Manually configure the server:
     i. Tailor and run the script to set up the server environment.
     ii. Create a DB2 repository database for DB2 AME.
     iii. Run the script to create the DB2 repository objects.
     iv. Bind packages to the DB2 repository.
     v. Run **`adhuap`** to insert the administration user ID into the repository.
     vi. Tailor the server configuration file.
     vii. Run the server.
- DB2 AME agents.
  a. Install the agent using the **`./adh11serveraix`** command.
  b. Manually configure the agent:
     i. Tailor and run the script to set up the agent environment.
     ii. Configure the db2audit facility.
     iii. Bind packages to the source databases and to the DB2 repository.
     iv. Tailor the agent configuration file.
     v. Run the agent.
- DB2 AME Administration client
  a. Install the Administration client using the **`AuditManagementExpertAdmin.exe`** command.
  b. Set up the DB2 AME profiles using the DB2 AME Administration client UI.

- DB2 AME Reporting client
    a. Make sure that the DB2TEMPDIR points to the SQLLIB directory where the DB2 server, gateway, or client is installed. The DB2TEMPDIR is used by the DB2 AME reporting client to find the required IBM Data Server Driver for JDBC and SQLJ.
    b. Install the reporting client using the **`AuditManagementExpertReporting.exe`** command.

Note: DB2 AME server and agent configuration requires manually altering of several configuration files and scripts. The details of what and how the files have to be modified are listed in the *IBM DB2 AME for Multiplatforms User's Guide*, which is available for download at:

`http://publib.boulder.ibm.com/epubs/pdf/adhu1a10.pdf`

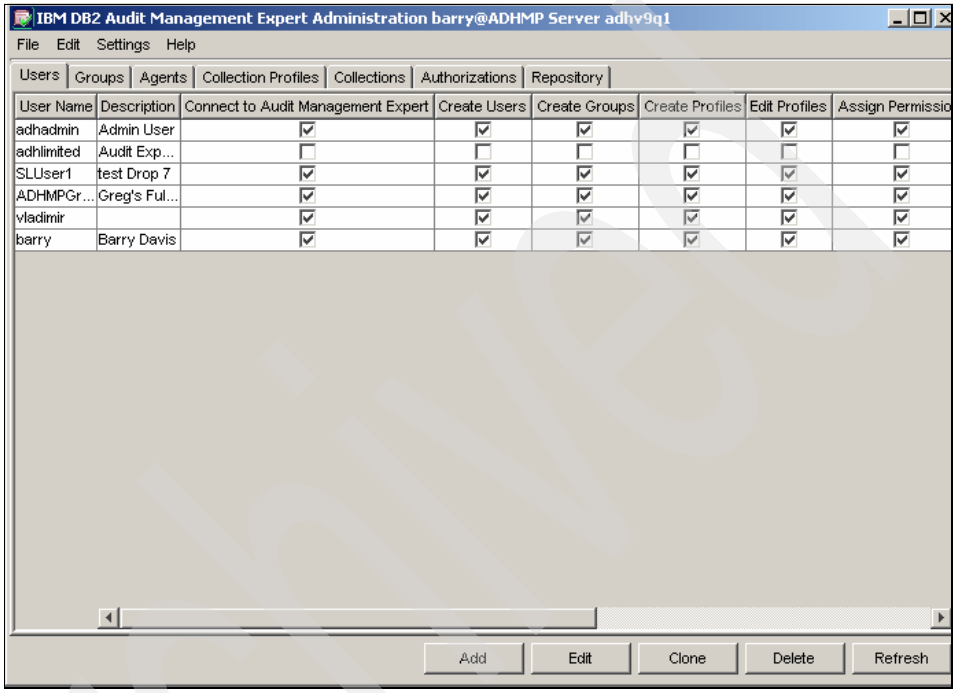
The guide is also accessible from the follow Web site:

`http://www-306.ibm.com/software/data/db2imstools/db2tools-library.html`

The user's guide also describes the steps required to configure the DB2 AME clients.

Administration client

The DB2 AME Administration client is designed for the administrator to configure all aspects of auditing within the DB2 AME. Figure 8-15 shows the main DB2 AME Administration client window, which includes the Users, Groups, Agents, Collection Profiles, Collections, Authorizations, and Repository tabs.



User Name	Description	Connect to Audit Management Expert	Create Users	Create Groups	Create Profiles	Edit Profiles	Assign Permissio
adhadmin	Admin User	✓	✓	✓	✓	✓	✓
adhlimited	Audit Exp...	☐	☐	☐	☐	☐	☐
SLUser1	test Drop 7	✓	✓	✓	✓	✓	✓
ADHMPGr...	Greg's Ful...	✓	✓	✓	✓	✓	✓
vladimir		✓	✓	✓	✓	✓	✓
barry	Barry Davis	✓	✓	✓	✓	✓	✓

Figure 8-15 The DB2 AME Administration client main window

The administrator can perform the administration tasks in each category through these tabs:

- Users: displays DB2 AME users and enables privileged users to add, edit, clone, or delete users.
- Groups: displays DB2 AME groups and enables privileged users to add, edit, clone, or delete groups.
- Agents: displays the names of the agents connected to the DB2 AME server, their status, and information about their logging activities. Privileged users can modify the level of logging from this tab.
- Collection profiles: displays collection profiles and enables privileged users to add, edit, clone, or delete collection profiles.

- Collections: displays an overview of the application of collection profiles to monitored databases and enables privileged users to add collections and apply them to databases.
- Authorizations: displays which users have authorizations for a particular database and enables privileged users to perform add or edit operations.
- Repository: displays JDBC connection information for use with the reporting interface and enables privileged users to edit the repository connection information.

Reporting client

The DB2 AME Reporting client is the client where audit results are displayed. This is the only client that the auditors need in order to view and analyze data that has been collected by DB2 AME into an audit repository. The interface consists of three tabs from which you can:

- Log in to an DB2 AME server or disconnect from a server (Log in tab).
- View audit data and generate reports (Reporting tab).
- Perform log analysis (Log Analysis tab).

Reporting

The Reporting tab in the main window enables auditors to view and analyze the data in a variety of ways through the following reporting options:

- DB2 INSTANCES

  This option enables the auditor to view activities for a selected instance. Detail reports are available to view details for a particular type of activity in the instance and can be filtered by user, application, tables, and so on. Figure 8-16 shows an sample instance level report.

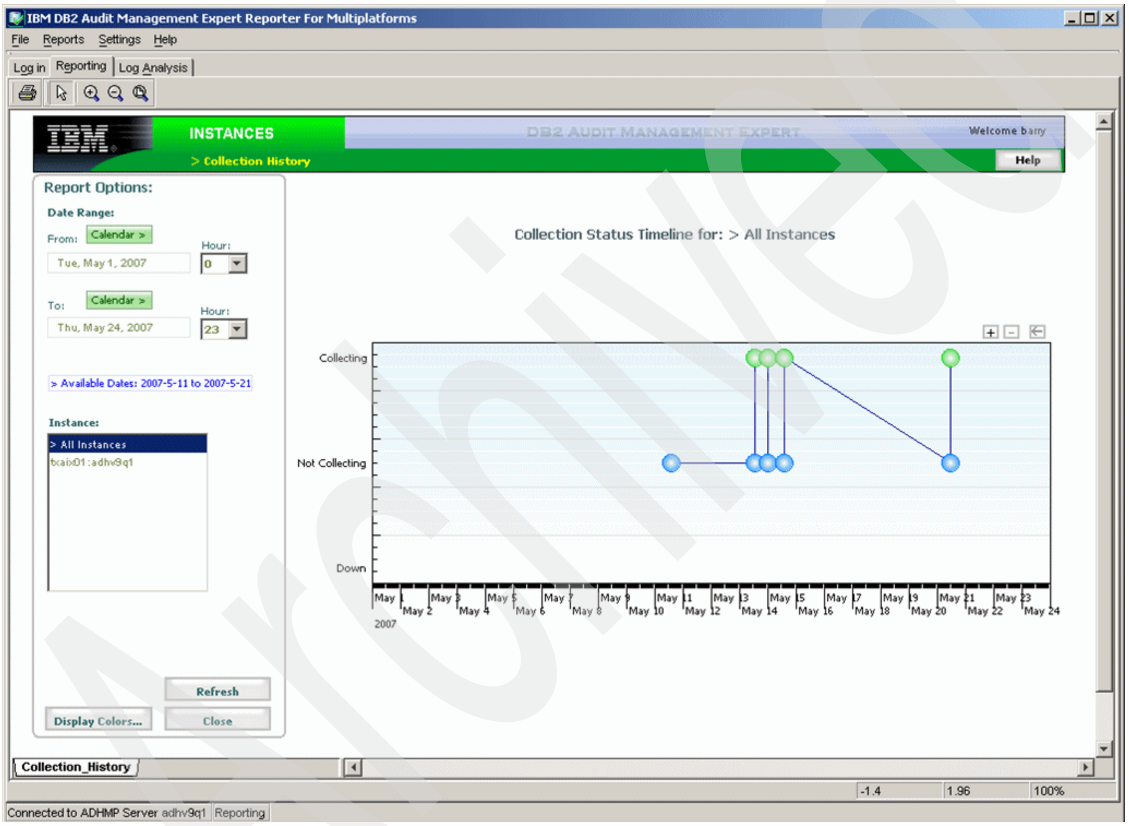
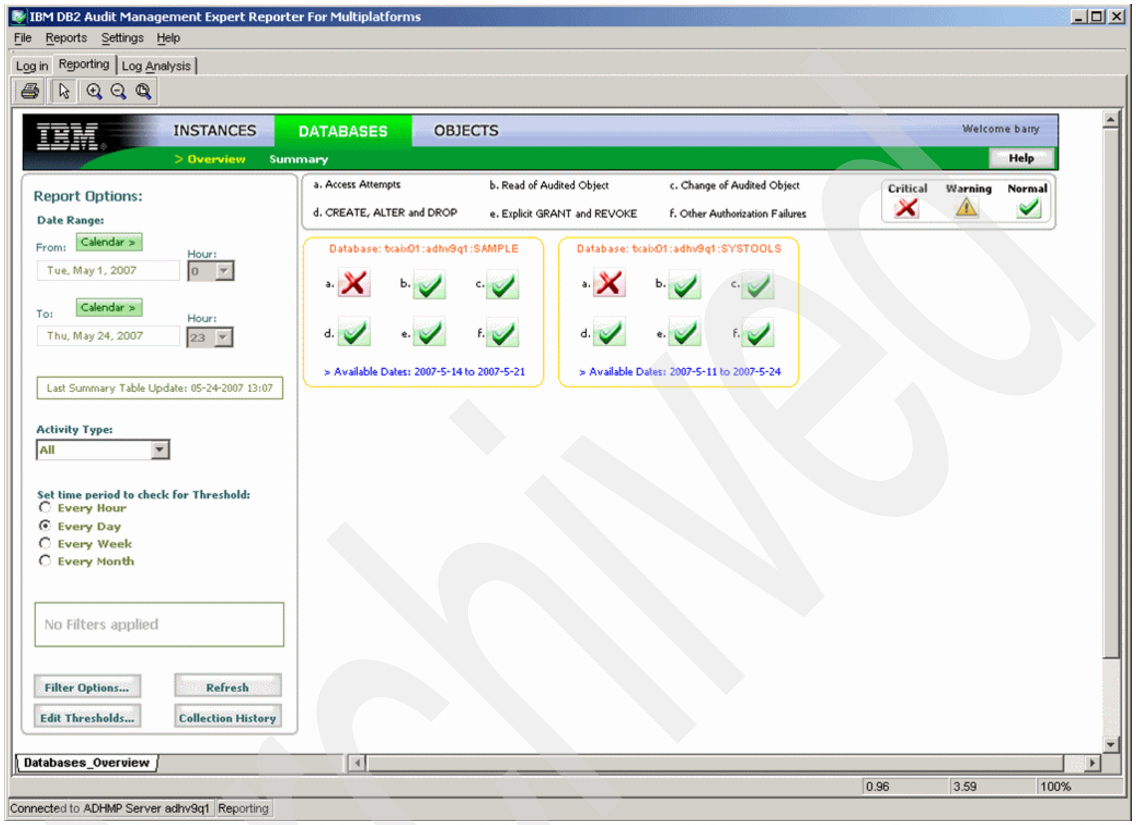


Figure 8-16 Reporting instance level audit data

- DB2 DATABASES (see Figure 8-17 on page 203)

  The auditor can use this option to view DB2 databases audit data from an overview or summary perspective and filter the data display using a variety of criteria. The overview report provides an overview of the auditing activities that have occurred on one or more DB2 databases. The summary reports provide information for a specific database in the form of a historical trend and bar charts.

Figure 8-17 and Figure 8-18 on page 204 are sample database level reports.


IBM DB2 Audit Management Expert Reporter For Multiplatforms
File Reports Settings Help
Log in Reporting Log Analysis
INSTANCES DATABASES OBJECTS
Welcome barry
> Overview Summary
Help

Report Options:

Date Range:

From: Calendar > Hour: Tue, May 1, 2007 0

To: Calendar > Hour: Thu, May 24, 2007 23

Last Summary Table Update: 05-24-2007 13:07

Activity Type: All

Set time period to check for Threshold:
- Every Hour
- [✓ Every Day]
- Every Week
- Every Month

No Filters applied

Filter Options... Refresh
Edit Thresholds... Collection History

a. Access Attempts b. Read of Audited Object c. Change of Audited Object
d. CREATE, ALTER and DROP e. Explicit GRANT and REVOKE f. Other Authorization Failures

Critical	Warning	Normal
✗	⚠	✓

Database: txaix01:adhv9q1:SAMPLE

a. ✗	b. ✓	c. ✓
d. ✓	e. ✓	f. ✓

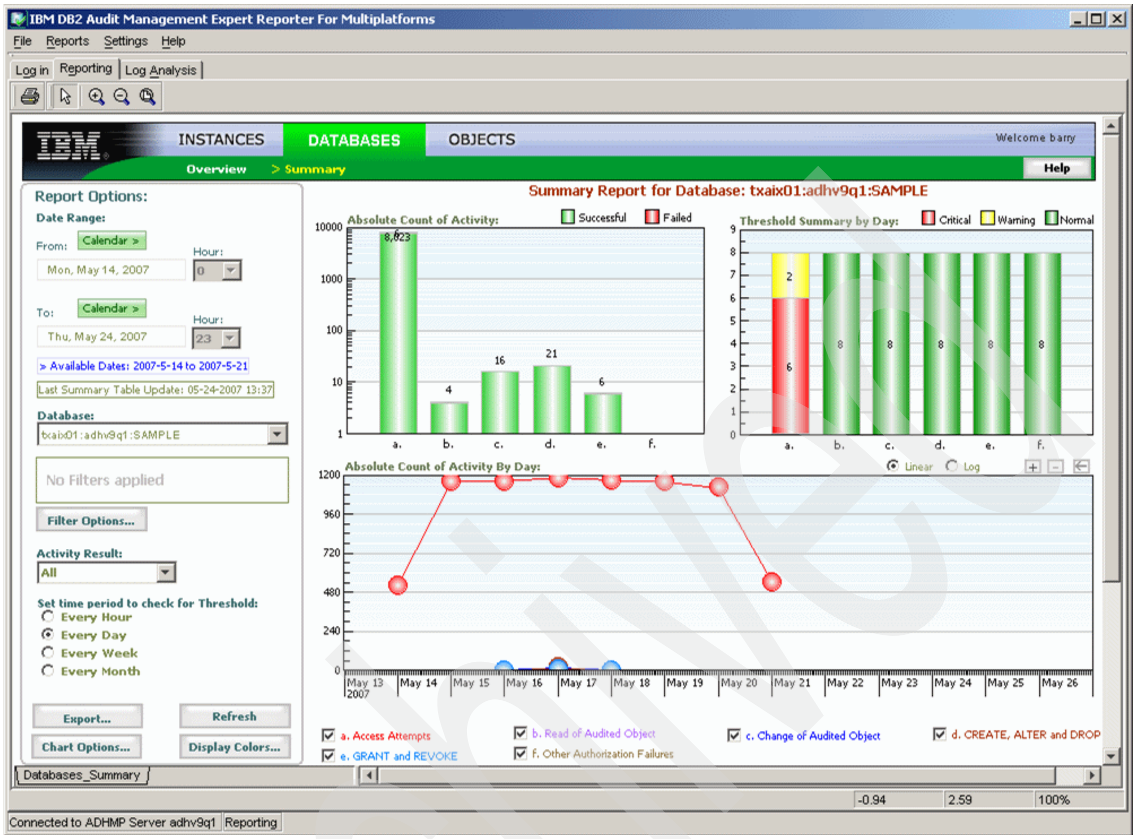
> Available Dates: 2007-5-14 to 2007-5-21

Database: txaix01:adhv9q1:SYSTOOLS

a. ✗	b. ✓	c. ✓
d. ✓	e. ✓	f. ✓

> Available Dates: 2007-5-11 to 2007-5-24

Figure 8-17 Reporting database level audit data - overview



IBM DB2 Audit Management Expert Reporter For Multiplatforms
File Reports Settings Help
Log in Reporting Log Analysis
INSTANCES DATABASES OBJECTS
Welcome barry
Overview > Summary
Help
Absolute Count of Activity:
Successful Failed
8,623
4
16
21
6
a. b. c. d. e. f.
Threshold Summary by Day:
Critical Warning Normal
2
6
8
Absolute Count of Activity By Day:
Linear Log
May 13 2007 May 14 May 15 May 16 May 17 May 18 May 19 May 20 May 21 May 22 May 23 May 24 May 25 May 26
a. Access Attempts
b. Read of Audited Object
c. Change of Audited Object
d. CREATE, ALTER and DROP
e. GRANT and REVOKE
f. Other Authorization Failures
Filter Options...
Export...
Refresh
Chart Options...
Display Colors...
Databases_Summary
-0.94 2.59 100%

Summary Report for Database: txaix01:adhv9q1:SAMPLE

Report Options:

Date Range:

From: Mon, May 14, 2007 Hour: 0

To: Thu, May 24, 2007 Hour: 23

> Available Dates: 2007-5-14 to 2007-5-21

Last Summary Table Update: 05-24-2007 13:37

Database: txaix01:adhv9q1:SAMPLE

No Filters applied

Activity Result: All

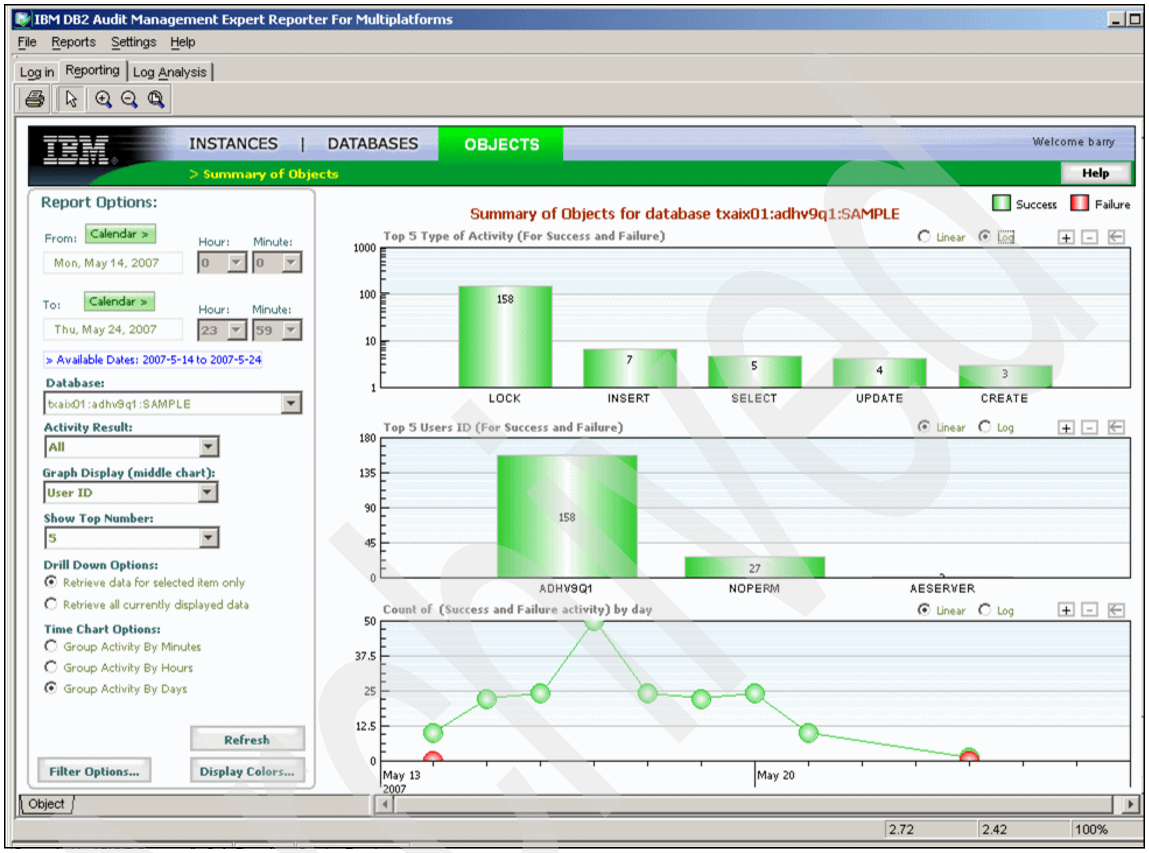
Set time period to check for Threshold:
Every Hour
[✓ Every Day]
Every Week
Every Month

Connected to ADHMP Server adhv9q1 Reporting

Figure 8-18 Reporting database level audit data - summary

- DB2 OBJECTS (see Figure 8-19)

  The auditor can view a summary of the activity for a particular table in a selected database. Figure 8-19 shows a sample report at the object level.



Report Options:

From: Calendar > Mon, May 14, 2007 Hour: 0 Minute: 0

To: Calendar > Thu, May 24, 2007 Hour: 23 Minute: 59

> Available Dates: 2007-5-14 to 2007-5-24

Database: txaix01:adhv9q1:SAMPLE

Activity Result: All

Graph Display (middle chart): User ID

Show Top Number: 5

Drill Down Options:
[✓ Retrieve data for selected item only]
Retrieve all currently displayed data

Time Chart Options:
Group Activity By Minutes
Group Activity By Hours
[✓ Group Activity By Days]

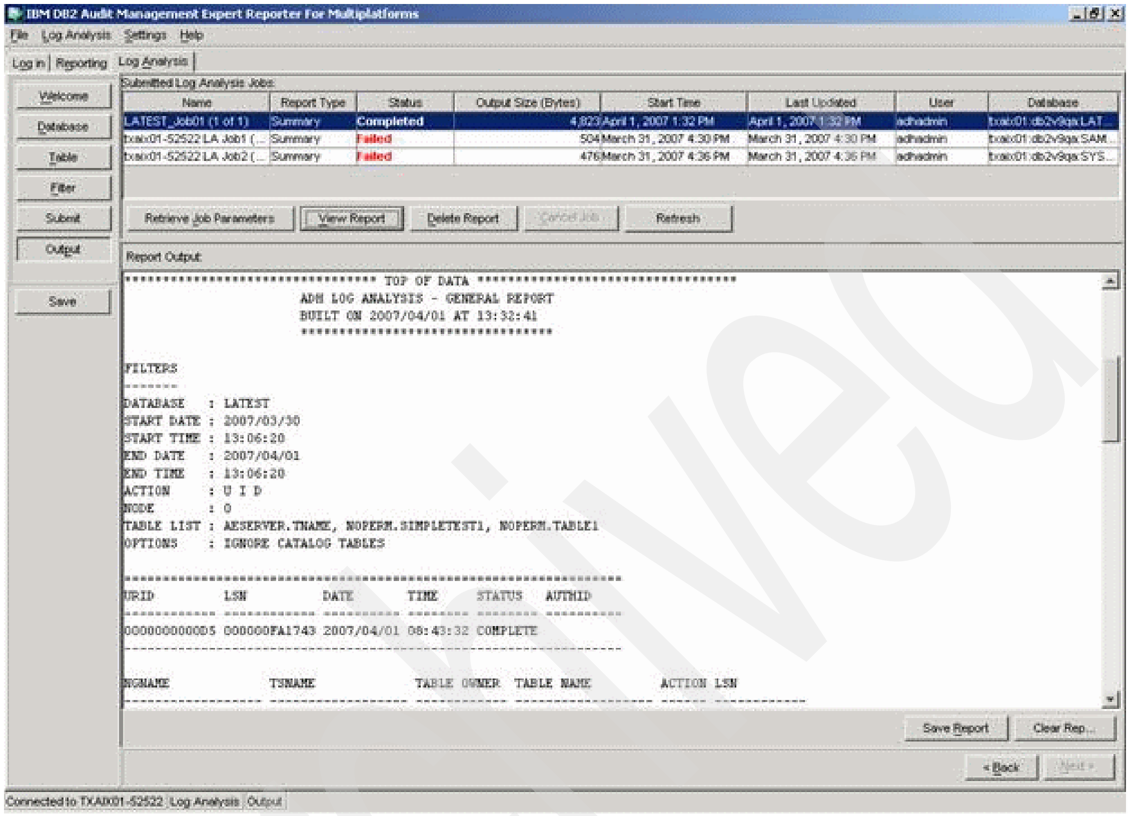
Figure 8-19 Reporting object level audit data

Log Analysis

Log Analysis Advisor provides authorized DB2 AME users with the following capabilities:

- View who modified audited tables in a DB2 system.
- See the actual changes made to the tables.
- Input the information needed for log analysis.
- Refine the amount of information returned.
- Create a report using a script.
- View reports from Reporting UI.

The Log Analysis Advisor is accessible from the Log Analysis tab in the DB2 AME Reporting client UI, as shown at Figure 8-20.



Name	Report Type	Status	Output Size (Bytes)	Start Time	Last Updated	User	Database
LATEST_Job01 (1 of 1)	Summary	Completed	4,823	April 1, 2007 1:32 PM	April 1, 2007 1:32 PM	adhadmin	txaix01:db2v9qa:LAT...
txaix01-52522 LA Job1 (...	Summary	Failed	504	March 31, 2007 4:30 PM	March 31, 2007 4:30 PM	adhadmin	txaix01:db2v9qa:SAM...
txaix01-52522 LA Job2 (...	Summary	Failed	476	March 31, 2007 4:36 PM	March 31, 2007 4:36 PM	adhadmin	txaix01:db2v9qa:SYS...

```
********************************** TOP OF DATA **********************************
                      ADH LOG ANALYSIS - GENERAL REPORT
                      BUILT ON 2007/04/01 AT 13:32:41
                      ********************************


FILTERS
-------
DATABASE   : LATEST
START DATE : 2007/03/30
START TIME : 13:06:20
END DATE   : 2007/04/01
END TIME   : 13:06:20
ACTION     : U I D
NODE       : 0
TABLE LIST : AESERVER.TNAME, NOPERM.SIMPLETEST1, NOPERM.TABLE1
OPTIONS    : IGNORE CATALOG TABLES

****************************************************************
URID         LSN          DATE       TIME     STATUS   AUTHID
------------ ------------ ---------- -------- -------- ----------
0000000000D5 000000FA1743 2007/04/01 08:43:32 COMPLETE
----------------------------------------------------------------

NGNAME            TSNAME            TABLE OWNER  TABLE NAME        ACTION LSN
----------------- ----------------- ------------ ----------------- ------ ------------
```

Figure 8-20 Log Analysis panel

8.3 IBM Optim - Enterprise Data Management

IBM Optim provides a single solution suite for managing enterprise application data throughout every stage of the information life cycle. You can apply business rules to archive, subset, access, store, retain, and protect your enterprise data. Optim capabilities are based on a consistent and proven data management methodology that aligns with your business objectives and scales across applications, databases, operating systems, and hardware platforms.

IBM Optim supports all major enterprise databases and operating systems, including DB2, Oracle®, Sybase, SQL Server®, Informix, IMS™, VSAM, Windows, UNIX, Linux, and z/OS. It supports the key ERP and CRM applications in use today: Oracle E Business Suite, PeopleSoft® Enterprise, JD Edwards® EnterpriseOne, Siebel®, and Amdocs CRM, as well as your custom and packaged applications.

Optim's comprehensive database archiving capabilities (archive, browse, restore, and delete) support managing data in production environments and address critical business issues for data growth management, information governance, business continuity, data retention compliance, e-discovery, as well as application upgrades, migrations, and retirements. Optim's Test Data Management capabilities (extract/move, edit, access, compare, convert/mask) support managing data in non-production (development, testing, and training) environments and address business issues for improving application reliability and protecting data privacy.

Protecting privacy data in production and non-production environments

Depending on the industry, operations, and types of applications, many production and non-production databases process sensitive information. The challenge is to provide the appropriate protection, while meeting business needs and ensuring that data is managed on a need-to-know basis.

Sensitive application data must be protected, whether it resides in a production system or non-production (development, testing, and training) environment. Most production environments have established standard security and access restrictions at the network level, the application level, and the database level to protect against data breaches. Physical entry access controls can be extended by implementing multi-factor authentication schemes. However, these protective measures cannot be simply replicated across every environment because the methods that protect data in production may not necessarily meet the unique requirements for protecting non-production environments.

What makes non-production environments so vulnerable? The answer lies in the nature of how non-production databases are created and used. For example, applications must be tested outside the production environment so that when testing reveals application errors, the live production system is not affected. Realistic data is essential for testing application functionality and to ensure accuracy and reliability.

In addition, adequate application development, testing, or training coverage can require multiple environments. Most often, one or more non-production environments can be created by simply cloning copies of the production database. By definition, this means that sensitive information is propagated from a secure production environment to one or more vulnerable non-production environments.

Industry analysts recognize that data privacy in the application development, testing, and training, as well as other non-production environments, is essential. They also concur that as a best practice, masking or de-identifying the data is a viable approach. De-identifying data in non-production environments is simply the process of systematically removing, masking, or transforming data elements that could be used to identify an individual. Data de-identification enables developers, testers, and trainers to use realistic data and produce valid results, while still complying with privacy protection rules. Data that has been scrubbed or cleansed in such a manner is generally considered acceptable to use in non-production environments and ensures that even if the data is stolen, exposed, or lost, it will be of no use to anyone.

IBM Optim Test Data Management and Data Privacy Solutions provide comprehensive capabilities for improving testing efficiencies and de-identifying application data that can be used effectively across non-production environments. Optim's data masking technology preserves the integrity of the data and produces consistent and accurate results that reflect the application logic. Masked data can be propagated accurately across multiple non-production environments to generate valid test results.

8.3.1 Fundamentals of effective data masking

Optim enables organizations to meet even the most complex data privacy challenges by providing the following fundamental components of effective data masking.

Application-aware data masking

Optim's application-aware data masking capabilities understand, capture, and process data elements accurately so that the masked data does not violate application logic. For example, surnames are replaced with random surnames, not with meaningless text strings. Numeric fields retain the appropriate structure and pattern. For example, if diagnostic codes are four digits, and range in value from 0001 to 1000, then a masked value of 2000 would be invalid in the context of the application test. Checksums remain valid so that functional tests pass all application validity checks. Most importantly, Optim propagates all masked data elements consistently throughout a test database and to other related applications and databases.

Context-aware data masking

Optim's context-aware, prepackaged data masking routines de-identify key data elements. Optim provides a variety of proven data masking techniques that can be used to de-identify many types of sensitive information, such as birth dates, bank account numbers, national identifiers (like Canada's social insurance numbers or Italy's codice fiscale), benefits information, health insurance identification numbers, and so on.

Prepackaged Transformation Library routines allow for accurately masking complex data elements, such as social security numbers, credit card numbers, and e-mail addresses. Built-in lookup tables support masking names and addresses. You can also incorporate site-specific data transformation routines that integrate processing logic from multiple related applications and databases and provide greater flexibility and creativity in supporting even the most complex data masking requirements.

Persistent data masking

Optim's persistent masking capability generates transformed replacement values for source columns and propagates the replacement values consistently and accurately across applications, databases, operating systems, and platforms. Persistent data masking capabilities ensure scalability for protecting privacy across multiple development, testing, and training environments.

Creating realistic test databases

Creating a test database using Optim is basically a two-step operation:

- The extract step copies data from one or more related tables in one or more database instances to an external file called an extract file. This step also includes the capability to mask data immediately following the extract and saving the masked data to the extract file.
- The insert step copies data from the extract file to one or more destination databases. If the destination tables do not exist, Optim generates the SQL to create the tables in the destination database before inserting the data.

Figure 8-21 briefly describes an overview of Optim's functions, features, and basic processing flow for creating and masking test data.

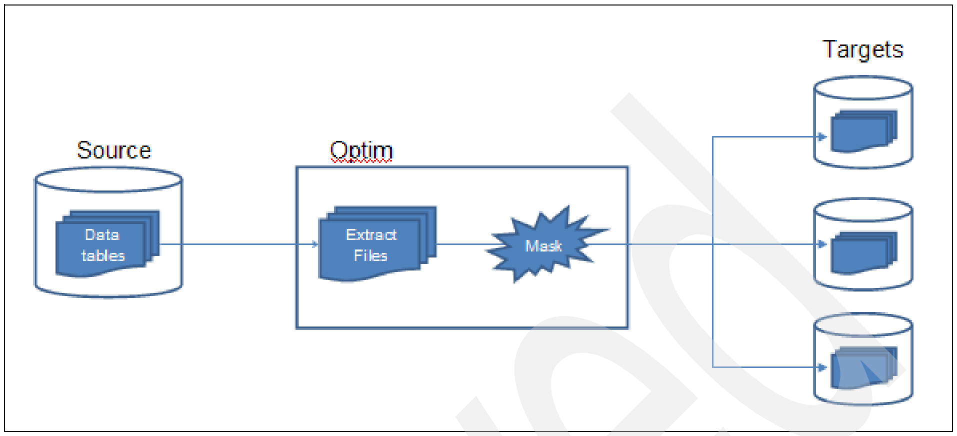
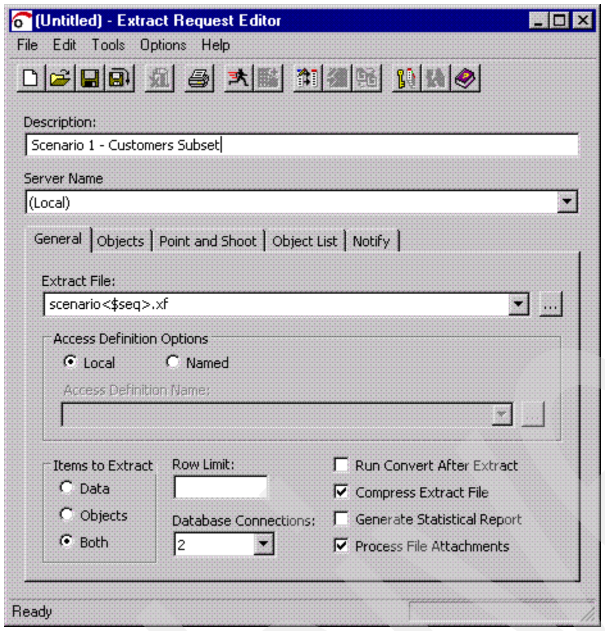


Figure 8-21 Optim's test data management and data masking basic data flow

8.3.2 Extracting realistic test data

To begin the extract process, select the Optim program from your desktop. Select **New** from the File menu in the main window and then select **Extract** from the Actions submenu to display the extract request editor.

Figure 8-22 shows the extract request editor.


(Untitled) - Extract Request Editor
File Edit Tools Options Help
General | Objects | Point and Shoot | Object List | Notify
Ready

Description: Scenario 1 - Customers Subset

Server Name: (Local)

Extract File: scenario<$seq>.xf

Access Definition Options: [✓ Local] Named

Access Definition Name:

Items to Extract: Data Objects [✓ Both]

Row Limit:

Database Connections: 2

☐ Run Convert After Extract
☑ Compress Extract File
☐ Generate Statistical Report
☑ Process File Attachments

Figure 8-22 Extract request editor

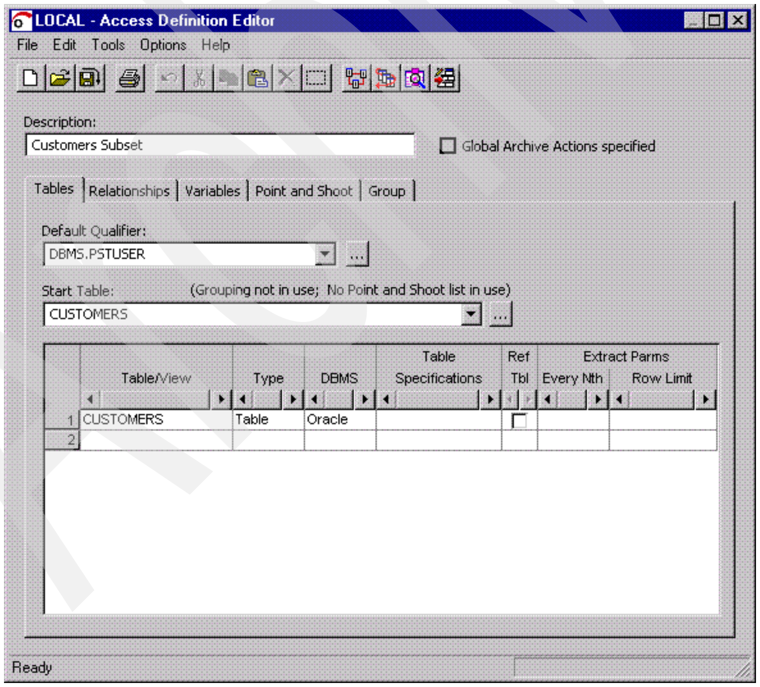
Use the extract request editor to create and edit requests for extracting data from a database. Use the General tab to name the extract file and open the access definition, in which you list the tables that contain the desired data and define specifications for the data in the listed tables.

- Description - indicates the type of information in the extract request or provides other useful information.
- Server Name - indicates the server or workstation used to run the extract process.
- Extract File - contains a copy of the specified source data.
- Access Definition Options - lets you specify the type of access definition used in the extract request. A local access definition is for immediate use and can be saved. A named access definition has been previously defined and saved for reuse.
- Items to Extract - You can you choose to extract data rows only, object definitions only, or both. The extract process always extracts column and table definition metadata, which can be used to create the destination tables, if necessary.
- Row Limit - the maximum number of rows to extract.

- Run Convert After Extract - The convert process can transform or mask data in the extract file. When you select this check box, the Convert tab in the extract request editor is enabled. Selecting this option allows you to run the convert (data masking) process immediately following the extract process. This offers an advantage because the masked data is saved to the extract file and is ready to be inserted into the destination database.
- Compress Extract File - Optim automatically compresses the extract file.
- Generate Statistical Report - includes statistical information in the extract process report.

After specifying the extract process parameters, you must provide an access definition. The access definition references the tables that contain the data, provides selection criteria for the data, and prescribes the way that relationships between tables are traversed. In other words, an access definition defines the data that you want to extract and provides table-specific parameters for processing.

In the extract request editor, select **Edit Access Definition** from the Tools menu. The access definition editor (Figure 8-23) is displayed with Tables as the active tab.


LOCAL - Access Definition Editor
File Edit Tools Options Help
Tables
Relationships
Variables
Point and Shoot
Group

Description: Customers Subset

☐ Global Archive Actions specified

Default Qualifier: DBMS.PSTUSER

Start Table: CUSTOMERS (Grouping not in use; No Point and Shoot list in use)

	Table/View	Type	DBMS	Table Specifications	Ref Tbl	Extract Parms Every Nth	Extract Parms Row Limit
1	CUSTOMERS	Table	Oracle		☐		
2							

Ready

Figure 8-23 Access definition editor

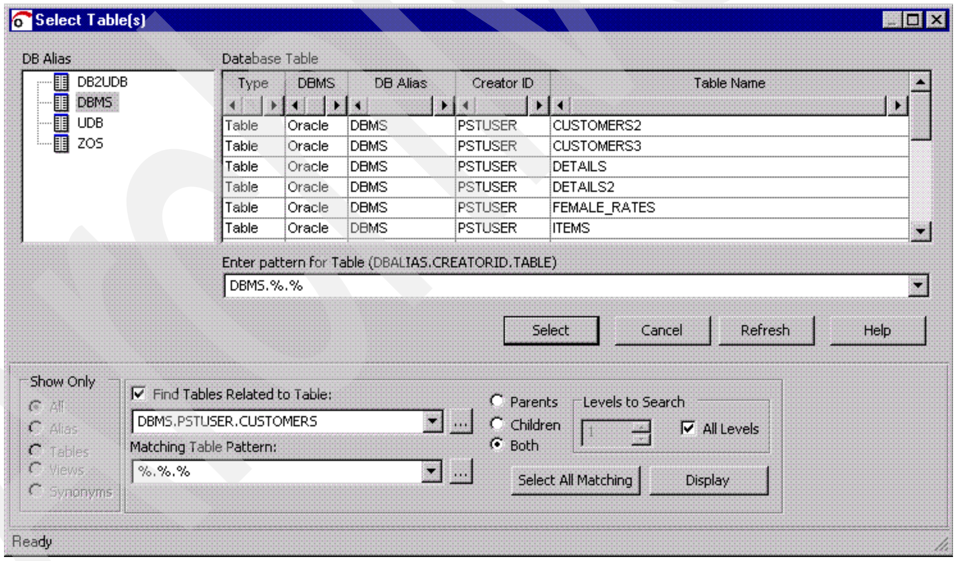
Use the access definition editor to create and edit processing specifications. In the Description box, you can enter the purpose or function of the access definition.

The default qualifier saves time when you enter table names in the grid, or table list.

Use the Tables tab to list the names of tables that contain the desired data and to define specifications for the data in the listed tables.

Each extract process begins with a start table, the table from which data is first extracted, and proceeds to obtain related data from the remaining tables in the table list. Optim automatically places the start table in the first line of the table list.

The table list references tables from which data is extracted. When you add a table name to the table list, Optim automatically displays the type of object referenced by the table name and the name of the DBMS for the table. To display the Select Table(s) dialog (Figure 8-24), right-click the start table name in the table list and select **Add Tables** from the shortcut menu. The dialog provides a list of tables, based on the default qualifier.



Type	DBMS	DB Alias	Creator ID	Table Name
Table	Oracle	DBMS	PSTUSER	CUSTOMERS2
Table	Oracle	DBMS	PSTUSER	CUSTOMERS3
Table	Oracle	DBMS	PSTUSER	DETAILS
Table	Oracle	DBMS	PSTUSER	DETAILS2
Table	Oracle	DBMS	PSTUSER	FEMALE_RATES
Table	Oracle	DBMS	PSTUSER	ITEMS

Enter pattern for Table (DBALIAS.CREATORID.TABLE)
DBMS.%.%

[✓] Find Tables Related to Table:
DBMS.PSTUSER.CUSTOMERS

Matching Table Pattern:
%.%.%

Parents / Children / [● Both]

Levels to Search: 1 [✓] All Levels

Ready

Figure 8-24 Select Table(s) dialog

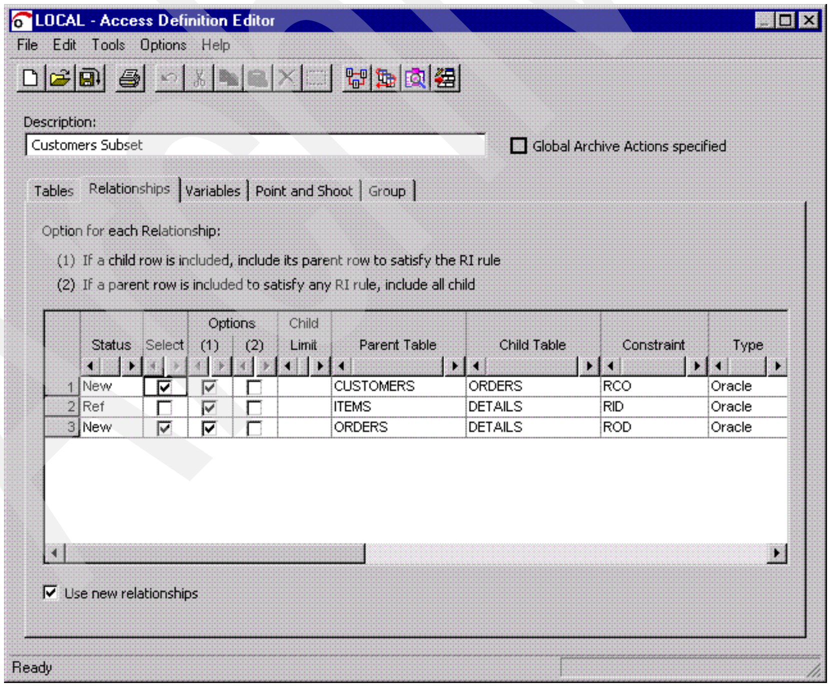
You can use the table list to define limits and selection criteria. You can set specifications for any table in the table list by right-clicking the table name and selecting a table specifications option from the shortcut menu. Optim provides several methods to limit the scope of the extracted data including:

- Columns - You can select the columns from which you want to extract data.
- SQL - You can define an SQL WHERE clause to subset the data.

Under the heading Extract Parms, you can limit the number of rows to extract by entering values under the following subheadings:

- Every Nth - Enter a numerical value N to extract every Nth row from the corresponding table.
- Row Limit - Enter the maximum number of rows to extract from a table.

After selecting the tables and specifying selection criteria for the data, you must define the traversal path. That is, select the relationships to be used and the direction in which the relationships are traversed, from parent to child, from child to parent, or in both directions, during the extract process. Use the options on the Relationships tab to define the traversal path for selecting data from the tables referenced in the access definition. All relationships between pairs of tables in the table list are displayed. See Figure 8-25.


LOCAL - Access Definition Editor
File Edit Tools Options Help
Tables Relationships Variables Point and Shoot Group
Ready

Description: Customers Subset

☐ Global Archive Actions specified

Option for each Relationship:

(1) If a child row is included, include its parent row to satisfy the RI rule

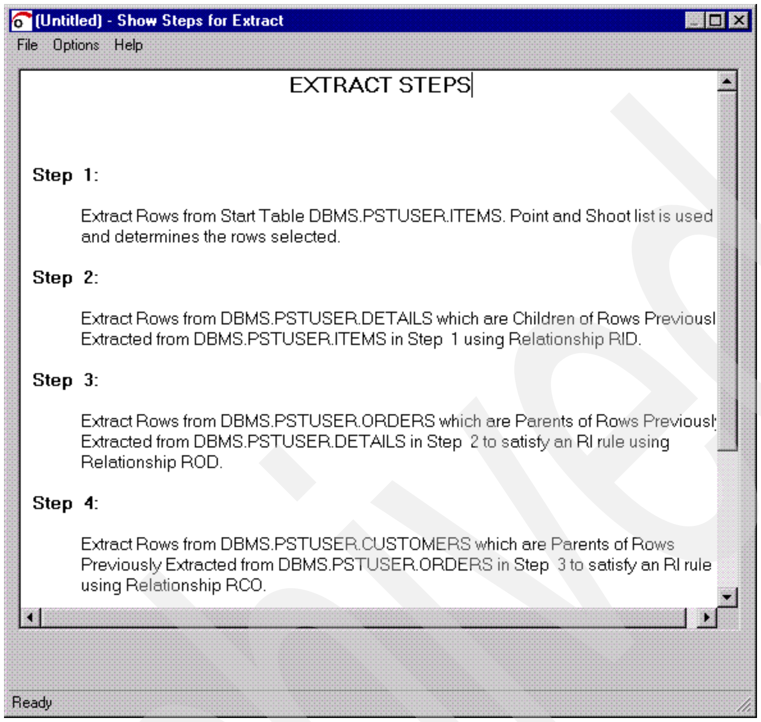
(2) If a parent row is included to satisfy any RI rule, include all child

	Status	Select	Options (1)	(2)	Child Limit	Parent Table	Child Table	Constraint	Type
1	New	✓	✓	☐		CUSTOMERS	ORDERS	RCO	Oracle
2	Ref	☐	✓	☐		ITEMS	DETAILS	RID	Oracle
3	New	✓	✓	☐		ORDERS	DETAILS	ROD	Oracle

✓ Use new relationships

Figure 8-25 Table relationships

Select **Show Steps** from the Tools menu to display a narrative that describes the traversal path of the extract request. See Figure 8-26.



EXTRACT STEPS

Step 1:

Extract Rows from Start Table DBMS.PSTUSER.ITEMS. Point and Shoot list is used and determines the rows selected.

Step 2:

Extract Rows from DBMS.PSTUSER.DETAILS which are Children of Rows Previousl Extracted from DBMS.PSTUSER.ITEMS in Step 1 using Relationship RID.

Step 3:

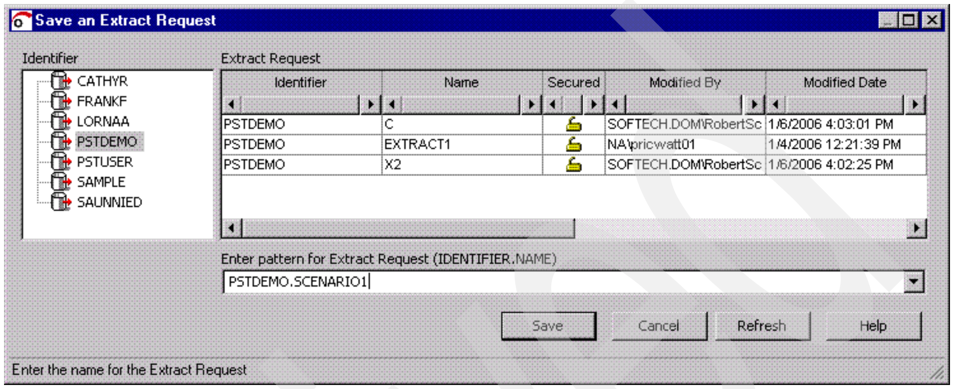
Extract Rows from DBMS.PSTUSER.ORDERS which are Parents of Rows Previousl Extracted from DBMS.PSTUSER.DETAILS in Step 2 to satisfy an RI rule using Relationship ROD.

Step 4:

Extract Rows from DBMS.PSTUSER.CUSTOMERS which are Parents of Rows Previously Extracted from DBMS.PSTUSER.ORDERS in Step 3 to satisfy an RI rule using Relationship RCO.

Figure 8-26 Extract steps in text

Although you need not save an extract request to process it, you should save the extract request that you just created so that it may be used again. In the extract request editor, select **Save** from the File menu to display the Save the Extract Request dialog (Figure 8-27). Type a suitable name in the Enter pattern for Extract Request box and click **Save**.



Identifier	Name	Secured	Modified By	Modified Date
PSTDEMO	C	🔓	SOFTECH.DOM\RobertSc	1/6/2006 4:03:01 PM
PSTDEMO	EXTRACT1	🔓	NA\pricwatt01	1/4/2006 12:21:39 PM
PSTDEMO	X2	🔓	SOFTECH.DOM\RobertSc	1/6/2006 4:02:25 PM

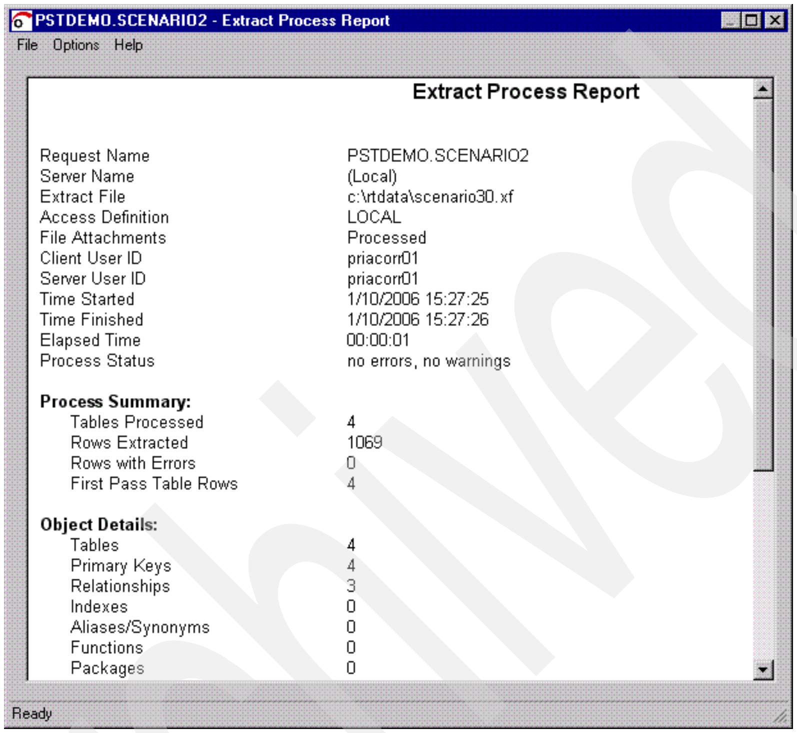
Enter pattern for Extract Request (IDENTIFIER.NAME)

PSTDEMO.SCENARIO1

Enter the name for the Extract Request

Figure 8-27 Save an Extract Request

Process the extract request from the extract request editor by selecting **Run** from the File menu. When the extract process has completed, the extract process report (Figure 8-28) is displayed. In the File menu, select **Save As** to save the report, or select **Print** to print the report.



Extract Process Report

```
Request Name            PSTDEMO.SCENARIO2
Server Name             (Local)
Extract File            c:\rtdata\scenario30.xf
Access Definition       LOCAL
File Attachments        Processed
Client User ID          priacorr01
Server User ID          priacorr01
Time Started            1/10/2006 15:27:25
Time Finished           1/10/2006 15:27:26
Elapsed Time            00:00:01
Process Status          no errors, no warnings

Process Summary:
    Tables Processed        4
    Rows Extracted          1069
    Rows with Errors        0
    First Pass Table Rows   4

Object Details:
    Tables                  4
    Primary Keys            4
    Relationships           3
    Indexes                 0
    Aliases/Synonyms        0
    Functions               0
    Packages                0
```

Figure 8-28 Extract Process Report

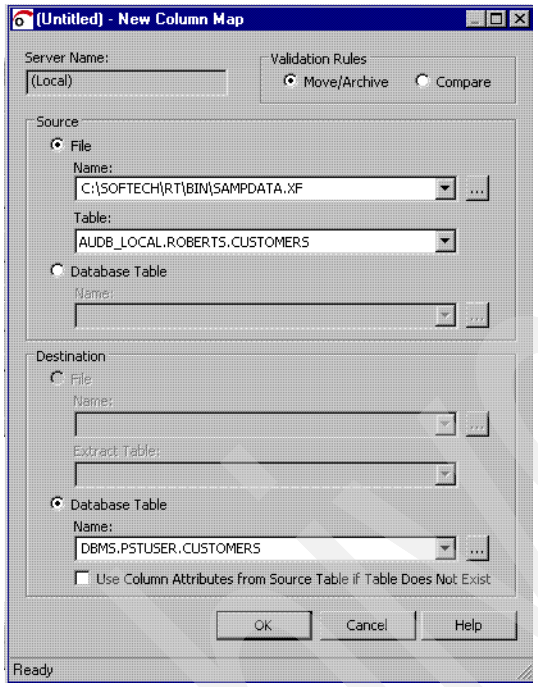
8.3.3 Convert process

You can mask data immediately following the extract process if you select that option on the extract request editor. You also can save the extracted data to an extract file and use it as the input to the convert process. The Convert process requires a column map to specify the masking techniques that you want to use for each column.

Column maps can be generated and saved for use with the insert process. To create a column map:

1. In the main window, select **New** from the File menu.

2. Select **Column Map** from the Definitions submenu to open the New Column Map dialog. See Figure 8-29.



Server Name: (Local)

Validation Rules: [✓ Move/Archive] Compare

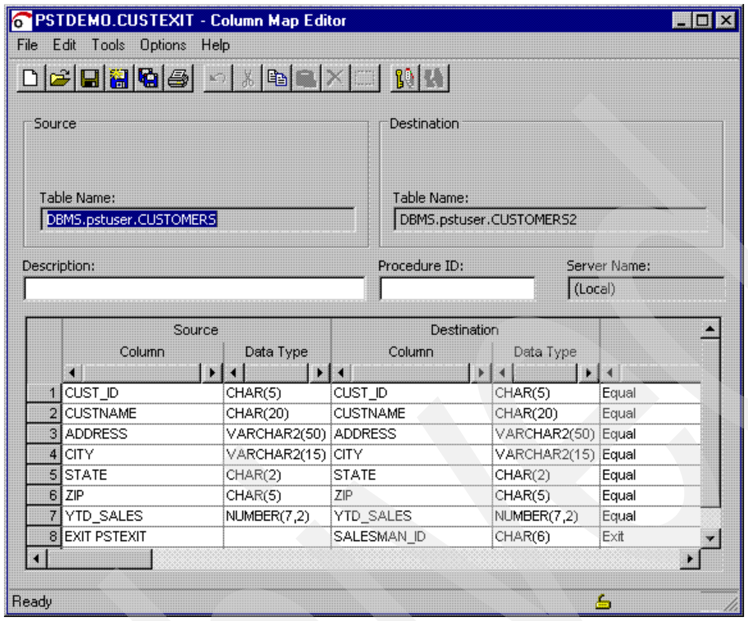
Source: [✓ File]
Name: C:\SOFTECH\RT\BIN\SAMPDATA.XF
Table: AUDB_LOCAL.ROBERTS.CUSTOMERS
Database Table
Name:

Destination: File
Name:
Extract Table:
[✓ Database Table]
Name: DBMS.PSTUSER.CUSTOMERS
Use Column Attributes from Source Table if Table Does Not Exist

Figure 8-29 New Column Map

3. For Validation Rules, select **Move/Archive**.
4. Specify the name of the source table containing the columns that you want to map:
   – If you select the File option, supply the file name and specify the name of a table in that file.
   – If you select the Database option, supply the name of the database table.
5. Specify the name of the destination table containing the columns that you want to map.

6. Click **OK** to open the column map editor. See Figure 8-30. Before displaying a column map, Optim verifies that the source and destination tables exist.



PSTDEMO.CUSTEXIT - Column Map Editor

Source — Table Name: DBMS.pstuser.CUSTOMERS

Destination — Table Name: DBMS.pstuser.CUSTOMERS2

Description:

Procedure ID:

Server Name: (Local)

	Source		Destination		
	Column	Data Type	Column	Data Type	
1	CUST_ID	CHAR(5)	CUST_ID	CHAR(5)	Equal
2	CUSTNAME	CHAR(20)	CUSTNAME	CHAR(20)	Equal
3	ADDRESS	VARCHAR2(50)	ADDRESS	VARCHAR2(50)	Equal
4	CITY	VARCHAR2(15)	CITY	VARCHAR2(15)	Equal
5	STATE	CHAR(2)	STATE	CHAR(2)	Equal
6	ZIP	CHAR(5)	ZIP	CHAR(5)	Equal
7	YTD_SALES	NUMBER(7,2)	YTD_SALES	NUMBER(7,2)	Equal
8	EXIT PSTEXIT		SALESMAN_ID	CHAR(6)	Exit

Ready

Figure 8-30 Column Map Editor

7. Modify the source column names, as needed. When you create a new column map, Optim populates the grid automatically on the basis of your selections in the New Column Map dialog. Edit the values in the first grid column to map columns.

   Data masking functions provide a variety of methods for transforming sensitive data. Access to these functions requires an IBM Optim Data Privacy license. The following are some examples of the different types of rules that can be used when creating and masking test data:

   – Static rule: specifies a constant value for a specific column.

   – Sourcecol rule: defines a column in the source data that is used for the target value.

   – Lookup rule: retrieves values for the source, based on a lookup table column search on the current value.

   – Mask rule: modifies values by replacing positions within a value with a pattern value or a static value.

   – Expression® rule: defines a formula for masking to dynamically calculate the value of the column.

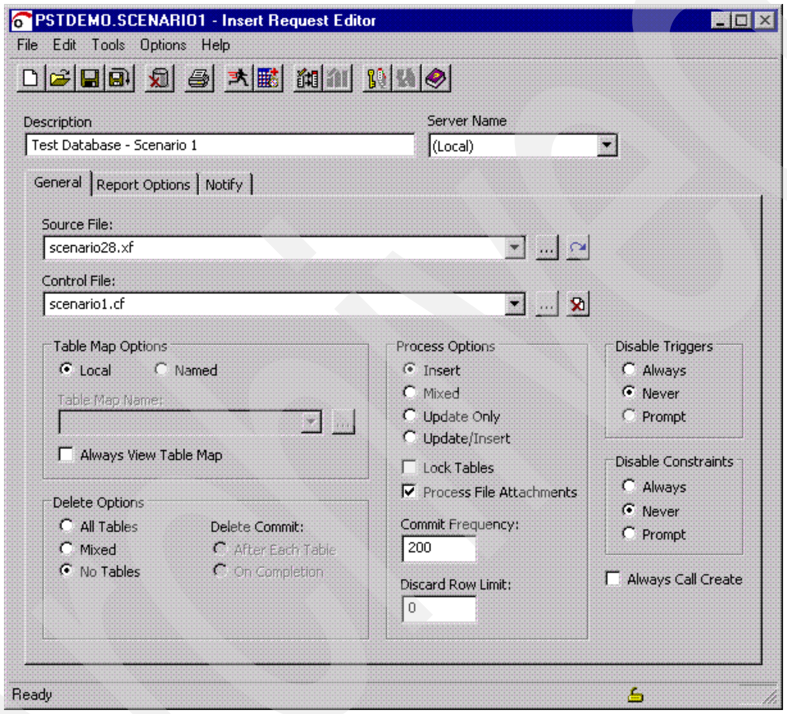
   - Random rule: generates random data of a specified type within certain bounds.
   - Pattern rule: specifies a pattern to be used to generate a masked value. Patterns can be used for character generation or string selector.
   - User-defined rule: defines your own data masking rules by extending abstract Java class com.ibm.db2.gri.server.userFunction.UserFunction.

   Each masking rule is associated with one column in the target. A masking rule defines how the value of a column is generated. Multiple rules can be applied to a column. When multiple generation rules are defined for one column, they are evaluated in the order in which they were defined. Masking rules are validated at the time when the target is used to populate the database. If the result is not appropriate for the target column, an error is generated. For example, attempting to put a CHAR value into an INTEGER column results in an error at database population time. When a rule is evaluated, the result either replaces, appends, or prefaces the existing value, depending on the action defined by the rule.

8. Select **Save** from the File menu to open the Save the Column Map dialog.
9. In the Pattern box, specify a unique column map name and then click **Save**.

8.3.4 Insert test data

The insert process copies data from the extract file to one or more destination databases. To begin the insert process, select **Open** from the File menu in the main window, and then select **Insert** from the Actions submenu to display the insert request editor (see Figure 8-31). Use the insert request editor to create and edit requests for inserting data into a database. You should enter a description to indicate the purpose or function of the insert request. Use the General tab to specify parameters for the Insert Process.


PSTDEMO.SCENARIO1 - Insert Request Editor

File Edit Tools Options Help

Description: Test Database - Scenario 1

Server Name: (Local)

General | Report Options | Notify

Source File: scenario28.xf

Control File: scenario1.cf

Table Map Options: [✓ Local] Named

Table Map Name:

Always View Table Map: ☐

Delete Options: All Tables, Mixed, [✓ No Tables]

Delete Commit: After Each Table, On Completion

Process Options: [✓ Insert] Mixed, Update Only, Update/Insert

Lock Tables: ☐

Process File Attachments: ☑

Commit Frequency: 200

Discard Row Limit: 0

Disable Triggers: Always, [✓ Never] Prompt

Disable Constraints: Always, [✓ Never] Prompt

Always Call Create: ☐

Ready

Figure 8-31 Insert Request Editor

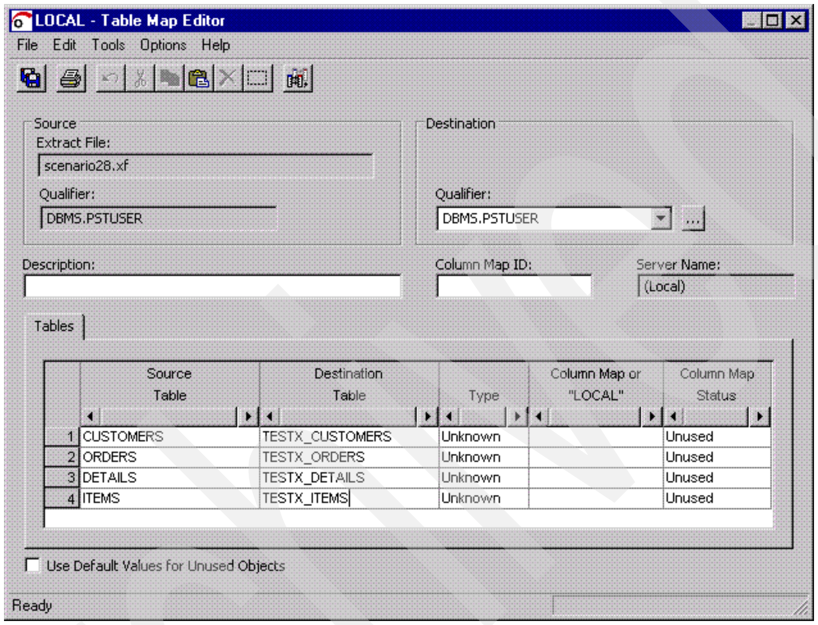
The options are:

- The source file is the extract file that contains your test data from the extract process. You can use the Browse button to locate the file.
- A control file is generated during the insert process to track the success or failure of each row in the extract file. Control files have a .cf extension by default.
- A table map directs the placement of data in the insert process by identifying and matching tables in the source, or extract file, with those in the database. In addition, you can use a table map to exclude tables from processing. Table

map options allow you to use a named table map that is saved and can be used with other process requests, or a local table map that is saved as part of the insert request.

- The Always View Table Map option provides an opportunity to review the table map specifications before you insert data.
- Delete options lets you delete rows from all or specified destination tables or retain all rows in destination tables (no delete).
- Process option allows you to select the type of insert processing to be performed and specify parameters to be used. You can insert new rows only, insert new rows and update existing rows, or update existing rows only. You can apply these options globally or on a table-by-table basis.
- Select **Lock Tables** (if authorized) to ensure that other database activity does not interfere during processing.
- Select **Process File Attachments** to process file attachments stored in the extract file.
- Specify the number of rows to process before committing the changes to the database.
- Specify the maximum number of rows that can be discarded. When the specified limit is reached and all rows in the array have been processed, the insert process is cancelled.
- Specify options to disable database triggers during the insert process.
- Specify options for disabling referential integrity constraints.
- Select the **Always Call Create** option to start the create utility to create or drop objects in the destination database before inserting the data. If you do not select the option, the create utility starts only when necessary to create desired objects in the destination database.

If the Always View Table Map option is selected in the insert request editor, the table map editor automatically displays when you run the process. A table map provides specifications on how to map source and destination tables in the extract file. The table map contains a column map that specifies how to map data from the source and destination columns. To display the table map editor, select **Edit Table Map** from the Tools menu in the Insert Request Editor. See Figure 8-32.



Source
Extract File: scenario28.xf
Qualifier: DBMS.PSTUSER

Destination
Qualifier: DBMS.PSTUSER

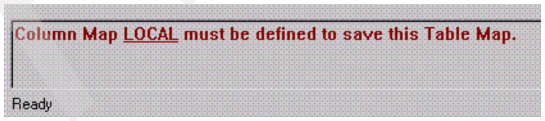
Description:
Column Map ID:
Server Name: (Local)

	Source Table	Destination Table	Type	Column Map or "LOCAL"	Column Map Status
1	CUSTOMERS	TESTX_CUSTOMERS	Unknown		Unused
2	ORDERS	TESTX_ORDERS	Unknown		Unused
3	DETAILS	TESTX_DETAILS	Unknown		Unused
4	ITEMS	TESTX_ITEMS	Unknown		Unused

☐ Use Default Values for Unused Objects

Figure 8-32 Table Map Editor

If you already have a column map defined, then type the name into the grid in the Column Map or LOCAL column. If you wish to generate a specific map for this run then you can type `LOCAL` into the relevant field then click outside the grid cell. The Status column changes from Unused to Unknown and a warning message (see Figure 8-33) displays at the bottom of the table map editor. The status is Unknown until you define the column map in the next step.



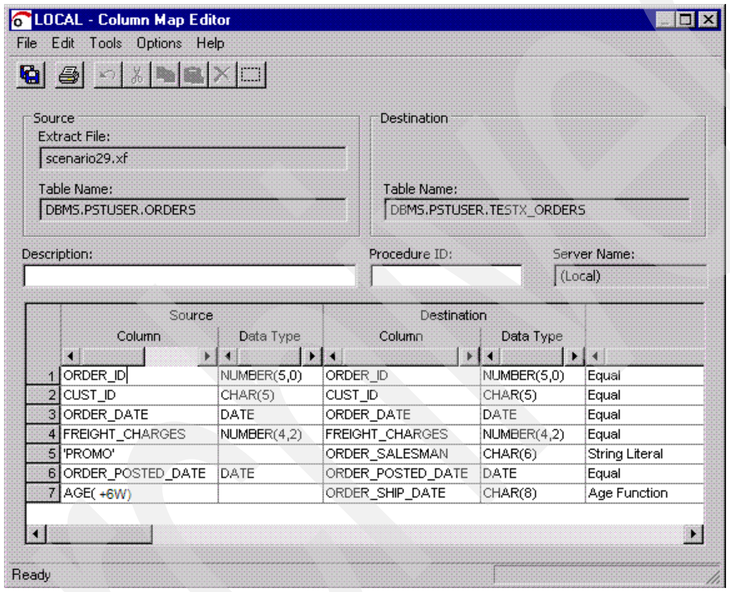
Column Map LOCAL must be defined to save this Table Map.

Ready

Figure 8-33 Error message for local column map

Type over source column values or right-click to select commands to Clear, Find, or Replace source column values. (If you clear a Source Column grid cell, the corresponding destination column value is not affected.) The masking capabilities are listed in 8.3.3, “Convert process” on page 217, and greater details are in the Optim Common Elements manual.

In this example, we use a LOCAL definition to make simple changes to two columns. We replace ORDER_SALESMAN in the Source column listed by typing the literal ‘PROMO’ in single quotation marks. (This literal value replaces the original value for the column. Replace ORDER_SHIP_DATE in the Source column list with an age function to add six weeks to the date. See Figure 8-34. The status of each destination column is displayed and the statuses for the two affected columns now show the function used to change the column.



Source
Extract File: scenario29.xf
Table Name: DBMS.PSTUSER.ORDERS

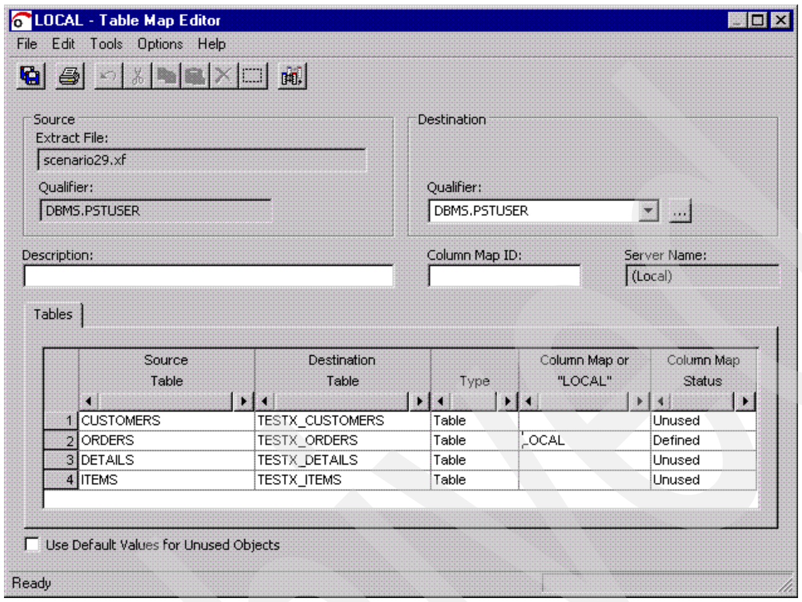
Destination
Table Name: DBMS.PSTUSER.TESTX_ORDERS

Description:
Procedure ID:
Server Name: (Local)

	Source		Destination		
	Column	Data Type	Column	Data Type	
1	ORDER_ID	NUMBER(5,0)	ORDER_ID	NUMBER(5,0)	Equal
2	CUST_ID	CHAR(5)	CUST_ID	CHAR(5)	Equal
3	ORDER_DATE	DATE	ORDER_DATE	DATE	Equal
4	FREIGHT_CHARGES	NUMBER(4,2)	FREIGHT_CHARGES	NUMBER(4,2)	Equal
5	'PROMO'		ORDER_SALESMAN	CHAR(6)	String Literal
6	ORDER_POSTED_DATE	DATE	ORDER_POSTED_DATE	DATE	Equal
7	AGE(+6W)		ORDER_SHIP_DATE	CHAR(8)	Age Function

Figure 8-34 Edited Column Map

Select **Update and Return** from the File menu in the column map editor to return to the table map editor. Note that the Status column for the ORDERS table on the table map editor has changed to Defined. See Figure 8-35.



Source
Extract File: scenario29.xf
Qualifier: DBMS.PSTUSER

Destination
Qualifier: DBMS.PSTUSER

Description:
Column Map ID:
Server Name: (Local)

	Source Table	Destination Table	Type	Column Map or "LOCAL"	Column Map Status
1	CUSTOMERS	TESTX_CUSTOMERS	Table		Unused
2	ORDERS	TESTX_ORDERS	Table	LOCAL	Defined
3	DETAILS	TESTX_DETAILS	Table		Unused
4	ITEMS	TESTX_ITEMS	Table		Unused

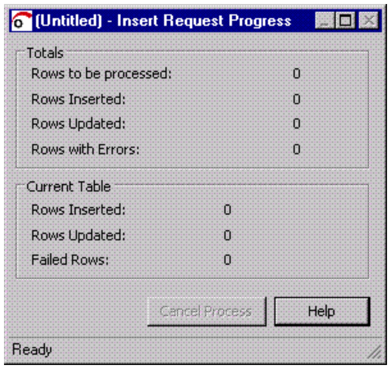
☐ Use Default Values for Unused Objects

Figure 8-35 Revised Table Map Editor

Select **Update and Return** from the File menu in the table map editor to return to the insert request editor.

Although you need not save an insert request to process it, we recommend saving an insert request. In the insert request editor, select **Save** from the File menu to display the Save an Insert Request dialog, then type a descriptive name into the dialog box and click **Save**.

In the insert request editor, select **Run** from the File menu to begin processing the insert request. The Insert Request Progress dialog is displayed as the insert request is processed (Figure 8-36).



Totals

Rows to be processed:	0
Rows Inserted:	0
Rows Updated:	0
Rows with Errors:	0

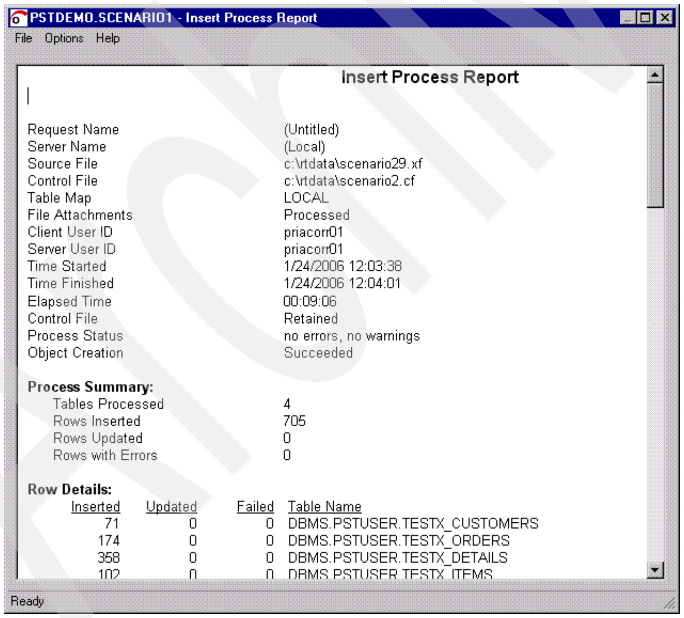
Current Table

Rows Inserted:	0
Rows Updated:	0
Failed Rows:	0

Ready

Figure 8-36 Insert request progress report

When the insert process has completed, the insert process report is displayed. See Figure 8-37.



```
                         Insert Process Report

Request Name                    (Untitled)
Server Name                     (Local)
Source File                     c:\rtdata\scenario29.xf
Control File                    c:\rtdata\scenario2.cf
Table Map                       LOCAL
File Attachments                Processed
Client User ID                  priacorr01
Server User ID                  priacorr01
Time Started                    1/24/2006 12:03:38
Time Finished                   1/24/2006 12:04:01
Elapsed Time                    00:09:06
Control File                    Retained
Process Status                  no errors, no warnings
Object Creation                 Succeeded

Process Summary:
    Tables Processed            4
    Rows Inserted               705
    Rows Updated                0
    Rows with Errors            0

Row Details:
       Inserted   Updated    Failed  Table Name
             71         0         0  DBMS.PSTUSER.TESTX_CUSTOMERS
            174         0         0  DBMS.PSTUSER.TESTX_ORDERS
            358         0         0  DBMS.PSTUSER.TESTX_DETAILS
            102         0         0  DBMS.PSTUSER.TESTX_ITEMS
```

Ready

Figure 8-37 Insert Process Report

Browse utility

You can use the browse utility to review the contents of an archive, compare, extract, or control file. You can browse the contents of a file to obtain information or to determine that the data is as expected. Default extensions for the files are as follows:

- Archive file .af
- Extract file .xf
- Compare file .cmp
- Control file .cf

When searching extracted data you can start with data in any table and join to related data in other tables. To browse data, in the main window, select **Browse** from the Utilities menu to display the Browse dialog. Then select the extract file from the File menu to populate the Browse File dialog. The Tables tab displays the fully qualified name and total number of extracted rows for each table in the file. See Figure 8-38.

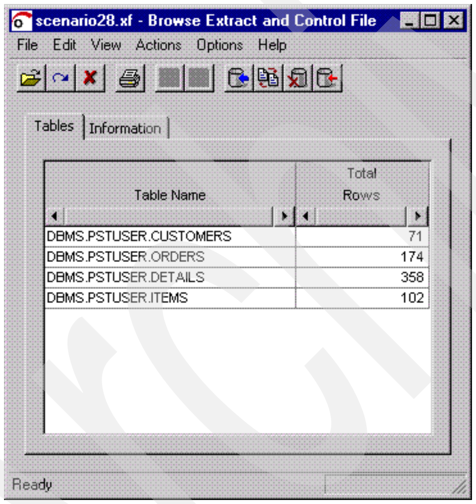

scenario28.xf - Browse Extract and Control File
File Edit View Actions Options Help
Tables
Information
Ready

Table Name	Total Rows
DBMS.PSTUSER.CUSTOMERS	71
DBMS.PSTUSER.ORDERS	174
DBMS.PSTUSER.DETAILS	358
DBMS.PSTUSER.ITEMS	102

Figure 8-38 Browse file

You can browse data in any table in a file. To display rows in a table, double-click the table name and a browse window displays the data (Figure 8-39). The browse window contains a toolbar that lets you select display options and menu choices that pertain to the display. For example, you can display or hide column attribute information, join and unjoin tables to view related data, and switch the display format.

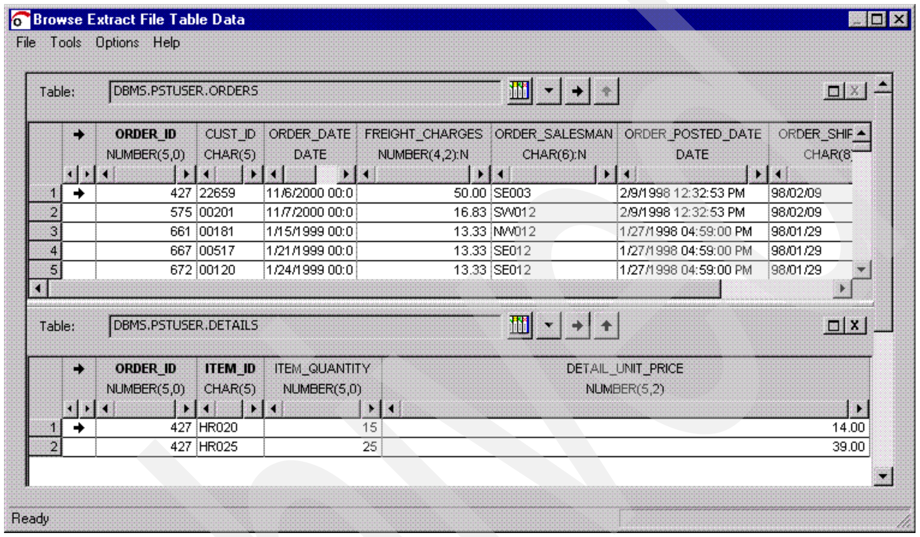


Table: DBMS.PSTUSER.ORDERS

		ORDER_ID	CUST_ID	ORDER_DATE	FREIGHT_CHARGES	ORDER_SALESMAN	ORDER_POSTED_DATE	ORDER_SHIF
		NUMBER(5,0)	CHAR(5)	DATE	NUMBER(4,2):N	CHAR(6):N	DATE	CHAR(8
1	→	427	22659	11/6/2000 00:0	50.00	SE003	2/9/1998 12:32:53 PM	98/02/09
2		575	00201	11/7/2000 00:0	16.83	SW012	2/9/1998 12:32:53 PM	98/02/09
3		661	00181	1/15/1999 00:0	13.33	NW012	1/27/1998 04:59:00 PM	98/01/29
4		667	00517	1/21/1999 00:0	13.33	SE012	1/27/1998 04:59:00 PM	98/01/29
5		672	00120	1/24/1999 00:0	13.33	SE012	1/27/1998 04:59:00 PM	98/01/29

Table: DBMS.PSTUSER.DETAILS

		ORDER_ID	ITEM_ID	ITEM_QUANTITY	DETAIL_UNIT_PRICE
		NUMBER(5,0)	CHAR(5)	NUMBER(5,0)	NUMBER(5,2)
1	→	427	HR020	15	14.00
2		427	HR025	25	39.00

Ready

Figure 8-39 View data from table in the extract file

When finished, return to the Browse File dialog by selecting Close File from the File menu on the Browse File Table Data dialog. Then select **Close** from the File menu to exit the Browse Utility.

A

Sample applications and scripts

This appendix provides the following applications and scripts:

- Trusted context applications in C++ and Java
- Scripts for creating or migrating DB2 audit tables

Sample trusted context application

This appendix provides the trusted context applications in C++ and Java.

C++ application

Example A-1 demonstrates how to switch users in trusted context using a C++ application.

Example: A-1 Trusted context application in C++

```
#include "TCUtil.h"

int main(int argc, char *argv[])
{
  SQLRETURN cliRC = SQL_SUCCESS;
   SQLHANDLE henv = 0, hdbc1, hstmt1, hstmt2;
   char* dbName = argv[1];
   char* origUserid = argv[2];
  char* origPassword = argv[3];
   char* switchUserid = argv[4];

  if (argc != 5) return printf("Usage:\n%s remoteDatabase userId
password switchUserId\n", argv[0]), -1;

  // Allocate the handles
  cliRC = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
  CHECK_RC_HENV(SQLAllocHandle, cliRC, henv);
  cliRC = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
  CHECK_RC_HENV(SQLAllocHandle, cliRC, henv);

  // Set the trusted connection attribute
  cliRC = SQLSetConnectAttr(hdbc1, SQL_ATTR_USE_TRUSTED_CONTEXT,
(SQLPOINTER)SQL_TRUE, SQL_IS_INTEGER);
  CHECK_RC_HDBC(SQLSetConnectAttr, cliRC, hdbc1);

  // Establish a trusted connection
  cliRC = SQLConnect(hdbc1, dbName, SQL_NTS, origUserid, SQL_NTS,
origPassword, SQL_NTS);
  CHECK_RC_HDBC(SQLConnect, cliRC,  hdbc1);

  // perform some work
  cliRC = SQLAllocStmt(hdbc1, &hstmt1);
  CHECK_RC_HDBC(SQLAllocStmt, cliRC, hdbc1);
  cliRC = SQLExecDirect(hstmt1, (SQLCHAR*) "SELECT * FROM
DB2INST1.LE_EMPLOYEE WHERE EMPNO='012214'", SQL_NTS);
  CHECK_RC_HSTMT(SQLExecDirect, cliRC, hstmt1);
  FetchAll(hstmt1);
  cliRC = SQLFreeStmt(hstmt1, SQL_CLOSE);
  CHECK_RC_HSTMT(SQLFreeStmt, cliRC, hstmt1);
  cliRC = SQLExecDirect(hstmt1, (SQLCHAR*) "UPDATE DB2INST1.LE_EMPLOYEE
SET JOB='SeniorSR' WHERE EMPNO='012214'", SQL_NTS);
  CHECK_RC_HSTMT(SQLExecDirect, cliRC, hstmt1);

   // Commit the work
  cliRC = SQLEndTran(SQL_HANDLE_DBC, hdbc1, SQL_COMMIT);
  CHECK_RC_HDBC(SQLEndTran, cliRC, hdbc1);


  // Switch the user ID on the trusted connection
  cliRC = SQLSetConnectAttr(hdbc1, SQL_ATTR_TRUSTED_CONTEXT_USERID,
switchUserid,  SQL_IS_POINTER);
  CHECK_RC_HDBC(SQLSetConnectAttr, cliRC, hdbc1);
//  cliRC = SQLSetConnectAttr(hdbc1, SQL_ATTR_TRUSTED_CONTEXT_PASSWORD,
switchPassword,  SQL_IS_POINTER);

  // Perform some work using the new userid
  cliRC = SQLAllocStmt(hdbc1, &hstmt2);
  CHECK_RC_HDBC(SQLAllocStmt, cliRC, hdbc1);
  cliRC = SQLExecDirect(hstmt2, (SQLCHAR*) "SELECT * FROM
DB2INST1.LE_EMPLOYEE WHERE empno='012214'", SQL_NTS);
  CHECK_RC_HSTMT(SQLExecDirect, cliRC, hstmt2);
  FetchAll(hstmt2);
  cliRC = SQLFreeStmt(hstmt2, SQL_CLOSE);
  CHECK_RC_HSTMT(SQLFreeStmt, cliRC, hstmt2);
  cliRC = SQLExecDirect(hstmt2, (SQLCHAR*) "UPDATE DB2INST1.LE_PENSIONS
SET BASE=BASE + 2500 WHERE EMPNO='012214'", SQL_NTS);
  CHECK_RC_HSTMT(SQLExecDirect, cliRC, hstmt2);

  //Commit the work
  cliRC = SQLEndTran(SQL_HANDLE_DBC, hdbc1, SQL_COMMIT);
  CHECK_RC_HDBC(SQLEndTran, cliRC, hdbc1);
  // Disconnect from database
  cliRC = SQLDisconnect(hdbc1);
  CHECK_RC_HDBC(SQLDisconnect, cliRC, hdbc1);

   return 0;
}
```

Example A-2 shows the C++ program include file TCUtil.h.

Example: A-2 TCUtil.h

```
#include <stdio.h>
#include "sqlcli1.h"

// error macros
#define ERR(CLIError) if (CLIError == cliRC) return #CLIError
#define CHECK_RC_HENV(fn, cliRC, henv) CHECK_RC(fn, cliRC, henv, 0, 0)
#define CHECK_RC_HDBC(fn, cliRC, hdbc) CHECK_RC(fn, cliRC, 0, hdbc, 0)
#define CHECK_RC_HSTMT(fn, cliRC, hstmt) CHECK_RC(fn, cliRC, 0, 0,
hstmt)

#define CHECK_RC(fn, cliRC, henv, hdbc, hstmt) printf("%s: rc = %d
(%s)\n", #fn, cliRC, GetErrorString(cliRC));
\
 if (cliRC < SQL_SUCCESS)
\
    return printf("  ERROR %d (%s) in %s at line %d\n", cliRC,
GetErrorString(cliRC), __FILE__, __LINE__), GetErrorDetails(henv, hdbc,
hstmt), cliRC; \
 else if (cliRC > SQL_SUCCESS)
\
    printf("  WARNING %d (%s) in %s at line %d\n", cliRC,
GetErrorString(cliRC), __FILE__, __LINE__), GetErrorDetails(henv, hdbc,
hstmt);
```

Example A-3 lists the TCUTIL.c source.

Example: A-3 TCUTIL.c

```
#include "TCUtil.h"
#define MAX(a,b) (a>b?a:b)

// function to retrieve details of an SQL error/warning
void GetErrorDetails(SQLHANDLE hEnv, SQLHANDLE hDbc, SQLHANDLE hStmt)
{
  SQLCHAR sMessage[SQL_MAX_MESSAGE_LENGTH + 1],
sState[SQL_SQLSTATE_SIZE + 1];
  SQLINTEGER sCode;
  SQLSMALLINT nLength, nCnt = 1;

  if (!hEnv && !hDbc && !hStmt) return;
  while (SQLGetDiagRec((SQLSMALLINT)(hStmt ? SQL_HANDLE_STMT : (hDbc ?
SQL_HANDLE_DBC : SQL_HANDLE_ENV)), hStmt ? hStmt : (hDbc ? hDbc :
hEnv), nCnt++, sState, &sCode, sMessage, sizeof(sMessage), &nLength) ==
SQL_SUCCESS)
    printf("  SQLState          : %s\n  Native Error Code : %d\n  Error
message     : %s\n", sState, sCode, sMessage);
}

// function to format friendly error name
char* GetErrorString(int cliRC)
{
  ERR(SQL_SUCCESS); ERR(SQL_SUCCESS_WITH_INFO); ERR(SQL_NEED_DATA);
ERR(SQL_NO_DATA); ERR(SQL_STILL_EXECUTING); ERR(SQL_ERROR);
ERR(SQL_INVALID_HANDLE); return "?";
}

// function to fetch and display all remaining rows from a result set,
displays column names as well
int FetchAll(SQLHANDLE hstmt)
{
  #define MAX_COLUMNS 255
  SQLRETURN cliRC = SQL_SUCCESS;
  SQLSMALLINT i, j, nFetched = 0, nResultCols, colNameLen, colType,
colScale;
  SQLCHAR colName[32];
  SQLUINTEGER colSize;
  SQLINTEGER colDataDisplaySize, colDisplaySize[MAX_COLUMNS];

  struct
  {
    SQLCHAR *buff;
    SQLINTEGER len;
    SQLINTEGER buffLen;
  }
  outData[MAX_COLUMNS]; /* variable to read the results */


  cliRC = SQLNumResultCols(hstmt, &nResultCols);
  printf("FetchAll: Columns: %d\n", nResultCols);
  for (i = 0; i < nResultCols; i++)
  {
    cliRC = SQLDescribeCol(hstmt, (SQLSMALLINT)(i + 1), colName,
sizeof(colName), &colNameLen, &colType, &colSize, &colScale, NULL);
    if (cliRC) return 0;
    cliRC = SQLColAttribute(hstmt, (SQLSMALLINT)(i + 1),
SQL_DESC_DISPLAY_SIZE, NULL, 0, NULL, &colDataDisplaySize);
    if (cliRC) return 0;

   colDisplaySize[i] = MAX(colDataDisplaySize, colNameLen) + 1;
// set "column display size" to the larger of "column data display
size" and "column name length" and add one space between columns.
    printf("%-*.*s", (int)colDisplaySize[i], (int)colDisplaySize[i],
colName); // print the column name
   outData[i].buffLen = colDataDisplaySize + 1;
// set "output data buffer length" to "column data display size and add
one byte for null the terminator
   outData[i].buff = (SQLCHAR *)malloc((int)outData[i].buffLen);
// allocate memory to bind a column

    cliRC = SQLBindCol(hstmt, (SQLSMALLINT)(i + 1), SQL_C_CHAR,
outData[i].buff, outData[i].buffLen, &outData[i].len);
    if (cliRC) return 0;
  }

  printf("\n");
  for (i = 0; i < nResultCols; i++)
  {
    for (j = 1; j < (int)colDisplaySize[i]; j++)
      printf("-");
    printf(" ");
  }
  printf("\n");

  cliRC = SQLFetch(hstmt);
  while (cliRC != SQL_NO_DATA_FOUND)
  {
    for (i = 0; i < nResultCols; i++) // for all columns in this row
    {
      if (outData[i].len == SQL_NULL_DATA)
        printf("%-*.*s", (int)colDisplaySize[i],
(int)colDisplaySize[i], "NULL");
      else
        printf("%-*.*s", (int)colDisplaySize[i],
(int)colDisplaySize[i], outData[i].buff);
```

```
    }
    printf("\n");
    nFetched++;

    cliRC = SQLFetch(hstmt); // fetch next row
  }
  printf("FetchAll: %d rows fetched.\n", nFetched);

  for (i = 0; i < nResultCols; i++) free(outData[i].buff);
  return 1;
}
```

Java application

Example A-4 demonstrates how to switch users in trusted context using a Java application.

Example: A-4 Trusted context Java application

```
import java.sql.*;
import java.io.*;
import com.ibm.db2.jcc.*;

class TC
{
  public static void callCustomCode1(Connection con, PreparedStatement
stmt, ResultSet rs)  throws Exception
  {
    stmt = con.prepareStatement("SELECT * FROM DB2INST1.LE_EMPLOYEE
WHERE EMPNO='012214'");
    rs = stmt.executeQuery();
    FetchAll(rs);
    stmt = con.prepareStatement("UPDATE DB2INST1.LE_EMPLOYEE SET
JOB='SeniorSR' WHERE EMPNO='012214'");
    stmt.executeUpdate();

  }

  public static void callCustomCode2(Connection con, PreparedStatement
stmt, ResultSet rs)  throws Exception
  {
    stmt = con.prepareStatement("SELECT * FROM DB2INST1.LE_EMPLOYEE
WHERE EMPNO='012214'");
    rs = stmt.executeQuery();
    FetchAll(rs);
    stmt = con.prepareStatement("UPDATE DB2INST1.LE_PENSIONS SET 
BASE=BASE + 2500 WHERE EMPNO='012214'");
    stmt.executeUpdate();
  }

  public static void main(String argv[])  throws Exception
  {
    if (argv.length != 6)
      throw new Exception("\n\nUsage: TC database server port userId 
password switchUserId\n");

    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    com.ibm.db2.jcc.DB2ConnectionPoolDataSource ds = null;

    try
    {
      String dbname = argv[0];
      String server = argv[1];
      String port = argv[2];
      String userId = argv[3];
      String password = argv[4];
      String switchUserId = argv[5];

      ds = 
(com.ibm.db2.jcc.DB2ConnectionPoolDataSource)Class.forName("com.ibm.db2
.jcc.DB2ConnectionPoolDataSource").newInstance();
      ds.setDatabaseName(dbname);
    ds.setDriverType(4);
      ds.setPortNumber(Integer.parseInt(port));
      ds.setServerName(server);

      java.util.Properties props = new java.util.Properties();
      Object[] objects = ds.getDB2TrustedPooledConnection(userId, 
password, props);

      // The first item obtained from the getDB2TrustedPooledConnection 
call is a connection object. Cast it to a PooledConnection object.
    javax.sql.PooledConnection pooledCon = 
(javax.sql.PooledConnection)objects[0];
```

```
      // The second item obtained from the
getDB2TrustedPooledConnection call is the cookie for the connection.
Cast it as a byte array.
      byte[] cookie = (byte[])(objects[1]);

      // Get a trusted connection
      con =
((com.ibm.db2.jcc.DB2PooledConnection)pooledCon).getConnection();
    callCustomCode1(con, stmt, rs);

      // Switch the user ID on the trusted connection
      con =
((com.ibm.db2.jcc.DB2PooledConnection)pooledCon).getDB2Connection(cooki
e, switchUserId, null/*switchPassword*/, null, null, null, props);
    callCustomCode2(con, stmt, rs);
    }
    catch (SQLException e)
    {
      System.out.println("Execute exception: " + e + ".");
      for (SQLException cur = e; cur != null; cur =
cur.getNextException())
      {
          cur.printStackTrace();
      }
    }
    finally
    {
      try
      {
        if (rs != null)
          rs.close();
        if (stmt != null)
          stmt.close();
        if (con != null)
          con.close();
      }
      catch (SQLException e)
      {
        System.out.println("Cleanup exception: ");
        for (SQLException cur = e; cur != null; cur =
cur.getNextException())
        {
            cur.printStackTrace();
        }
      }
    }
  }

  public static void FetchAll(ResultSet rs)  throws Exception
  {
    int i, j, col = (rs.getMetaData()).getColumnCount();
    System.out.println("Columns: " + col);


   int colDataDisplaySize, colNameLen, colDisplaySize[] = new int[255]
;
    String colName;

    for (i = 0; i < col; i++)
    {
      colDataDisplaySize = (rs.getMetaData()).getColumnDisplaySize(i +
1);
      colName = (rs.getMetaData()).getColumnName(i + 1);
      colNameLen = colName.length();

     colDisplaySize[i] = Math.max(colDataDisplaySize, colNameLen);  //
set "column display size" to the larger of "column data display size"
and "column name length" and add one space between columns.
      for ( j = colNameLen; j <= (int)colDisplaySize[i]; j++)
        colName += " ";
      System.out.print(colName); // print the column name
    }

    System.out.println();
    for (i = 0; i < col; i++)
    {
      for (j = 1; j <= (int)colDisplaySize[i]; j++)
        System.out.print("-");
      System.out.print(" ");
    }
    System.out.println();

    String query_results;
    int row = 0;

    while (rs.next())
    {
      for (i = 0; i < col; i++)
      {
        query_results = rs.getString(i + 1);
        for (j = query_results.length(); j <= (int)colDisplaySize[i];
j++)
          query_results += " ";
        System.out.print(query_results); // print the column name
      }
      System.out.println();
      row++;
    }
    System.out.println("Rows: " + row);
  }
}
```

DB2 audit tables migration scripts

This section provides the scripts for migrating DB2 audit tables.

db2audit_80.ddl

Example A-5 shows the script for creating required audit tables for DB2 8.

Example: A-5 db2audit_80.ddl

```
--
-- Sample DDL to create audit tables for Version DB2 8.0
--
--   -> assumes db2start issued
--   -> assumes connection to a database exists
--   -> assumes an 8K tablespace is available for use
--   -> assumes called by "db2 -tf db2audit_80.ddl"
--


--
-- AUDIT CATEGORY
--
CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                    CATEGORY CHAR(8),
                    EVENT VARCHAR(32),
                    CORRELATOR INTEGER,
                    STATUS INTEGER,
                    USERID VARCHAR(1024),
                    AUTHID VARCHAR(128));
```

```
--
-- CHECKING CATEGORY
--
CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       ACCESSAPP CHAR(18),
                       ACCESSATT CHAR(18),
                       PKGVER VARCHAR(64));

--
-- OBJMAINT CATEGORY
--
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64));

--
-- SECMAINT CATEGORY
--
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
                       GRANTEETYPE VARCHAR(32),
                       PRIVAUTH CHAR(18),
                       PKGVER VARCHAR(64));


--
-- SYSADMIN CATEGORY
--
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64));

--
-- VALIDATE CATEGORY
--
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       PLUGINNAME VARCHAR(32));

--
-- CONTEXT CATEGORY
--
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
                      PKGNAME VARCHAR(128),
                      PKGSECNUM SMALLINT,
                      STMTTEXT CLOB(2M),
                      PKGVER VARCHAR(64));
```

db2audit_90.ddl

Example A-6 shows the script for creating required audit tables for DB2 9.

Example: A-6   db2audit_90.ddl

```
--
-- Sample DDL to create audit tables for DB2 9.0
--
--   -> assumes db2start issued
--   -> assumes connection to a database exists
--   -> assumes an 8K tablespace is available for use
--   -> assumes called by "db2 -tf db2audit_90.ddl"
--

--
-- AUDIT CATEGORY
--
CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                    CATEGORY CHAR(8),
                    EVENT VARCHAR(32),
                    CORRELATOR INTEGER,
                    STATUS INTEGER,
                    USERID VARCHAR(1024),
                    AUTHID VARCHAR(128));

--
-- CHECKING CATEGORY
--
CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       ACCESSAPP CHAR(18),
                       ACCESSATT CHAR(18),
                       PKGVER VARCHAR(64),
                       CHKAUTHID VARCHAR(128));

--
-- OBJMAINT CATEGORY
--
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64));

--
-- SECMAINT CATEGORY
--
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
                       GRANTEETYPE VARCHAR(32),
                       PRIVAUTH CHAR(18),
                       PKGVER VARCHAR(64));

--
-- SYSADMIN CATEGORY
--
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64));

--
-- VALIDATE CATEGORY
--
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       PLUGINNAME VARCHAR(32));

--
-- CONTEXT CATEGORY
--
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
                      PKGNAME VARCHAR(128),
                      PKGSECNUM SMALLINT,
                      STMTTEXT CLOB(2M),
                      PKGVER VARCHAR(64));
```

db2audit_95.ddl

Example A-7 shows the script for creating required audit tables for DB2 9.5.

Example: A-7 db2audit_95.ddl

```
--
-- Sample DDL to create audit tables for DB2 9.5
--
--   -> assumes db2start issued
--   -> assumes connection to a database exists
--   -> assumes an 8K tablespace is available for use
--   -> assumes called by "db2 -tf db2audit_95.ddl"
--

--
-- AUDIT CATEGORY
--
CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                    CATEGORY CHAR(8),
                    EVENT VARCHAR(32),
                    CORRELATOR INTEGER,
                    STATUS INTEGER,
                    USERID VARCHAR(1024),
                    AUTHID VARCHAR(128),
                    DATABASE CHAR(8),
                    NODENUM SMALLINT,
                    COORDNUM SMALLINT,
                    APPID VARCHAR(255),
                    APPNAME VARCHAR(1024),
                    PKGSCHEMA VARCHAR(128),
                    PKGNAME VARCHAR(128),
                    PKGSECNUM SMALLINT,
                    PKGVER VARCHAR(64),
                    LCLTRANSID VARCHAR(10) FOR BIT DATA,
                    GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                    CLNTUSERID VARCHAR(255),
                    CLNTWRKSTNAME VARCHAR(255),
                    CLNTAPPNAME VARCHAR(255),
                    CLNTACCSTRING VARCHAR(255),
                    TRSTCTXNAME VARCHAR(255),
                    CONTRSTTYPE INTEGER,
                    ROLEINHERITED VARCHAR(128),
                    POLNAME VARCHAR(128),
                    POLASSOCOBJTYPE CHAR(1),
                    POLASSOCSUBOBJTYPE CHAR(1),
                    POLASSOCNAME VARCHAR(128),
                    POLASSOCOBJSCHEMA VARCHAR(128),
                    AUDITSTATUS CHAR(1),
                    CHECKINGSTATUS CHAR(1),
                    CONTEXTSTATUS CHAR(1),
                    EXECUTESTATUS CHAR(1),
                    EXECUTEDATA CHAR(1),
                    OBJMAINTSTATUS CHAR(1),
                    SECMAINTSTATUS CHAR(1),
                    SYSADMINSTATUS CHAR(1),
                    VALIDATESTATUS CHAR(1),
                    ERRORTYPE CHAR(8),
                    DATAPATH VARCHAR(1024),
                    ARCHIVEPATH VARCHAR(1024));

--
-- CHECKING CATEGORY
--
CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       ACCESSAPP CHAR(18),
                       ACCESSATT CHAR(18),
                       PKGVER VARCHAR(64),
                       CHKAUTHID VARCHAR(128),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128));

--
-- OBJMAINT CATEGORY
--
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64),
                       SECPOLNAME VARCHAR(128),
                       ALTERACTION VARCHAR(32),
                       PROTCOLNAME VARCHAR(128),
                       COLSECLABEL VARCHAR(128),
                       SECCOLNAME VARCHAR(128),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128));

--
-- SECMAINT CATEGORY
--
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
                       GRANTEETYPE VARCHAR(32),
                       PRIVAUTH CHAR(18),
                       PKGVER VARCHAR(64),
                       ACCESSTYPE VARCHAR(32),
                       ASSUMEAUTHID VARCHAR(128),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       GRANTORTYPE VARCHAR(32),
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXUSER VARCHAR(128),
                       TRSTCTXUSERAUTH INTEGER,
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128));

--
-- SYSADMIN CATEGORY
--
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128));

--
-- VALIDATE CATEGORY
--
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       PLUGINNAME VARCHAR(32),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128));
```

```
--
-- CONTEXT CATEGORY
--
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
                      PKGNAME VARCHAR(128),
                      PKGSECNUM SMALLINT,
                      STMTTEXT CLOB(8M),
                      PKGVER VARCHAR(64),
                      LCLTRANSID VARCHAR(10) FOR BIT DATA,
                      GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                      CLNTUSERID VARCHAR(255),
                      CLNTWRKSTNAME VARCHAR(255),
                      CLNTAPPNAME VARCHAR(255),
                      CLNTACCSTRING VARCHAR(255),
                      TRSTCTXNAME VARCHAR(255),
                      CONTRSTTYPE INTEGER,
                      ROLEINHERITED VARCHAR(128));

--
-- EXECUTE CATEGORY
--
CREATE TABLE EXECUTE ( TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       SESSNAUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       UOWID BIGINT,
                       ACTIVITYID BIGINT,
                       STMTINVOCID BIGINT,
                       STMTNESTLVL BIGINT,
                       ACTIVITYTYPE VARCHAR(32),
                       STMTTEXT CLOB(8M),
                       STMTISOLATIONLVL CHAR(8),
                       COMPENVDESC BLOB(8K),
                       ROWSMODIFIED INTEGER,
                       ROWSRETURNED BIGINT,
                       SAVEPOINTID BIGINT,
                       STMTVALINDEX INTEGER,
                       STMTVALTYPE CHAR(16),
                       STMTVALDATA CLOB(128K));
```

db2audit_migrate_80_to_90.ddl

Example A-8 shows the script for migrating the DB2 audit tables from DB2 8 to DB2 9.

Example: A-8 db2audit_migrate_80_to_90.ddl

```
--
-- Sample DDL to migrate audit tables from DB2 8.0 to DB2 9.0
--
--   -> assumes db2start issued
--   -> assumes connection to a database exists
--   -> assumes the database is migrated to the current release by
MIGRATE DATABASE command
--   -> assumes an 8K tablespace is available for use
--   -> assumes all audit tables from DB2 8.0 exist
--   -> assumes called by "db2 -tf db2audit_migrate_80_to_90.ddl"
--

--
-- CHECKING CATEGORY
--
ALTER TABLE CHECKING
                  ADD COLUMN CHKAUTHID VARCHAR(128);
```

db2audit_migrate_90_to_95.ddl

Example A-9 shows the script for migrating DB2 audit tables from DB2 9 to DB2 9.5.

Example: A-9 db2audit_migrate_90_to_95.ddl

```
--
-- Sample DDL to migrate audit tables from DB2 9.0 to DB2 9.5
--
--   -> assumes db2start issued
--   -> assumes connection to a database exists
--   -> assumes the database is migrated to the current release by
MIGRATE DATABASE command
--   -> assumes an 8K tablespace is available for use
--   -> assumes all audit tables from DB2 9.0 exist
--   -> assumes called by "db2 -tf db2audit_migrate_90_to_95.ddl"
--

--
-- AUDIT CATEGORY
--
ALTER TABLE AUDIT
                  ADD COLUMN DATABASE CHAR(8)
                  ADD COLUMN NODENUM SMALLINT
                  ADD COLUMN COORDNUM SMALLINT
                  ADD COLUMN APPID VARCHAR(255)
                  ADD COLUMN APPNAME VARCHAR(1024)
                  ADD COLUMN PKGSCHEMA VARCHAR(128)
                  ADD COLUMN PKGNAME VARCHAR(128)
                  ADD COLUMN PKGSECNUM SMALLINT
                  ADD COLUMN PKGVER VARCHAR(64)
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128)
                  ADD COLUMN POLNAME VARCHAR(128)
                  ADD COLUMN POLASSOCOBJTYPE CHAR(1)
                  ADD COLUMN POLASSOCSUBOBJTYPE CHAR(1)
                  ADD COLUMN POLASSOCNAME VARCHAR(128)
                  ADD COLUMN POLASSOCOBJSCHEMA VARCHAR(128)
                  ADD COLUMN AUDITSTATUS CHAR(1)
                  ADD COLUMN CHECKINGSTATUS CHAR(1)
                  ADD COLUMN CONTEXTSTATUS CHAR(1)
                  ADD COLUMN EXECUTESTATUS CHAR(1)
                  ADD COLUMN EXECUTEDATA CHAR(1)
                  ADD COLUMN OBJMAINTSTATUS CHAR(1)
                  ADD COLUMN SECMAINTSTATUS CHAR(1)
                  ADD COLUMN SYSADMINSTATUS CHAR(1)
                  ADD COLUMN VALIDATESTATUS CHAR(1)
                  ADD COLUMN ERRORTYPE CHAR(8)
                  ADD COLUMN DATAPATH VARCHAR(1024)
                  ADD COLUMN ARCHIVEPATH VARCHAR(1024);

--
-- CHECKING CATEGORY
--
ALTER TABLE CHECKING
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);


--
-- OBJMAINT CATEGORY
--
ALTER TABLE OBJMAINT
                  ADD COLUMN SECPOLNAME VARCHAR(128)
                  ADD COLUMN ALTERACTION VARCHAR(32)
                  ADD COLUMN PROTCOLNAME VARCHAR(128)
                  ADD COLUMN COLSECLABEL VARCHAR(128)
                  ADD COLUMN SECCOLNAME VARCHAR(128)
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);

--
-- SECMAINT CATEGORY
--
ALTER TABLE SECMAINT
                  ADD COLUMN ACCESSTYPE VARCHAR(32)
                  ADD COLUMN ASSUMEAUTHID VARCHAR(128)
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN GRANTORTYPE VARCHAR(32)
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXUSER VARCHAR(128)
                  ADD COLUMN TRSTCTXUSERAUTH INTEGER
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);

--
-- SYSADMIN CATEGORY
--
ALTER TABLE SYSADMIN
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);
```

```
--
-- VALIDATE CATEGORY
--
ALTER TABLE VALIDATE
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);

--
-- CONTEXT CATEGORY
--
ALTER TABLE CONTEXT
                  ALTER COLUMN STMTTEXT SET DATA TYPE CLOB(8M)
                  ADD COLUMN LCLTRANSID VARCHAR(10) FOR BIT DATA
                  ADD COLUMN GLBLTRANSID VARCHAR(30) FOR BIT DATA
                  ADD COLUMN CLNTUSERID VARCHAR(255)
                  ADD COLUMN CLNTWRKSTNAME VARCHAR(255)
                  ADD COLUMN CLNTAPPNAME VARCHAR(255)
                  ADD COLUMN CLNTACCSTRING VARCHAR(255)
                  ADD COLUMN TRSTCTXNAME VARCHAR(255)
                  ADD COLUMN CONTRSTTYPE INTEGER
                  ADD COLUMN ROLEINHERITED VARCHAR(128);

--
-- EXECUTE CATEGORY
--
CREATE TABLE EXECUTE ( TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       SESSNAUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       CLNTUSERID VARCHAR(255),
                       CLNTWRKSTNAME VARCHAR(255),
                       CLNTAPPNAME VARCHAR(255),
                       CLNTACCSTRING VARCHAR(255),
                       TRSTCTXNAME VARCHAR(255),
                       CONTRSTTYPE INTEGER,
                       ROLEINHERITED VARCHAR(128),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64),
                       LCLTRANSID VARCHAR(10) FOR BIT DATA,
                       GLBLTRANSID VARCHAR(30) FOR BIT DATA,
                       UOWID BIGINT,
                       ACTIVITYID BIGINT,
                       STMTINVOCID BIGINT,
                       STMTNESTLVL BIGINT,
                       ACTIVITYTYPE VARCHAR(32),
                       STMTTEXT CLOB(8M),
                       STMTISOLATIONLVL CHAR(8),
                       COMPENVDESC BLOB(8K),
                       ROWSMODIFIED INTEGER,
                       ROWSRETURNED BIGINT,
                       SAVEPOINTID BIGINT,
                       STMTVALINDEX INTEGER,
                       STMTVALTYPE CHAR(16),
                       STMTVALDATA CLOB(128K));
```

ssl.java

Example A-10 shows the Java code that establishes an SSL connection using the DB2DataSource interface and issues a SELECT statement to retrieve data.

Example: A-10 Encryption test application

```
import java.sql.*;
import java.io.*;
import com.ibm.db2.jcc.*;

class SSL
{
  public static void callCustomCode(Connection con, PreparedStatement
stmt, ResultSet rs)  throws Exception
  {
    stmt = con.prepareStatement("SELECT * FROM DB2INST1.LE_EMPLOYEE
WHERE EMPNO='012214'");
    rs = stmt.executeQuery();
    FetchAll(rs);
  }

  public static void main(String argv[])  throws Exception
  {
    if (argv.length != 5)
      throw new Exception("\n\nUsage: SSL database server port userId
password\n");

    Connection con = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    com.ibm.db2.jcc.DB2DataSource ds = null;

    try
    {
      String dbname = argv[0];
      String server = argv[1];
      String port = argv[2];
      String userId = argv[3];
      String password = argv[4];

      ds =
(com.ibm.db2.jcc.DB2DataSource)Class.forName("com.ibm.db2.jcc.DB2DataSo
urce").newInstance();
      ds.setDatabaseName(dbname);
      ds.setDriverType(4);
      ds.setPortNumber(Integer.parseInt(port));
      ds.setServerName(server);
      ds.setUser(userId);
      ds.setPassword(password);
      ds.setSslConnection(true);

      con = ds.getConnection();

      callCustomCode(con, stmt, rs);
    }
    catch (SQLException e)
    {
      System.out.println("Execute exception: " + e + ".");
      for (SQLException cur = e; cur != null; cur =
cur.getNextException())
```

```
      {
        cur.printStackTrace();
      }
    }
    finally
    {
      try
      {
        if (rs != null)
          rs.close();
        if (stmt != null)
          stmt.close();
        if (con != null)
          con.close();
      }
      catch (SQLException e)
      {
        System.out.println("Cleanup exception: ");
        for (SQLException cur = e; cur != null; cur =
cur.getNextException())
        {
          cur.printStackTrace();
        }
      }
    }
  }

  public static void FetchAll(ResultSet rs)  throws Exception
  {
    int i, j, col = (rs.getMetaData()).getColumnCount();
    System.out.println("Columns: " + col);


    int colDataDisplaySize, colNameLen, colDisplaySize[] = new int[255]
;
    String colName;

    for (i = 0; i < col; i++)
    {
      colDataDisplaySize = (rs.getMetaData()).getColumnDisplaySize(i +
1);
      colName = (rs.getMetaData()).getColumnName(i + 1);
      colNameLen = colName.length();
      colDisplaySize[i] = Math.max(colDataDisplaySize, colNameLen);  //
set "column display size" to the larger of "column data display size"
and "column name length" and add one space between columns.
      for ( j = colNameLen; j <= (int)colDisplaySize[i]; j++)
        colName += " ";
      System.out.print(colName); // print the column name
    }

    System.out.println();
    for (i = 0; i < col; i++)
    {
      for (j = 1; j <= (int)colDisplaySize[i]; j++)
        System.out.print("-");
      System.out.print(" ");
    }
    System.out.println();

    String query_results;
    int row = 0;

    while (rs.next())
    {
      for (i = 0; i < col; i++)
      {
        query_results = rs.getString(i + 1);
        for (j = query_results.length(); j <= (int)colDisplaySize[i];
j++)
          query_results += " ";
        System.out.print(query_results); // print the column name
      }
      System.out.println();
      row++;
    }
    System.out.println("Rows: " + row);
  }
}
```

B

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG247555`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com``/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247555.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
Roles.zip	Zipped code samples for creating example tables for role and trusted contexts
TC.zip	Zipped code samples for trusted contexts
Encryption.zip	Java sample code for testing encryption
auditMigration.zip	Zipped code samples for creating and migrating DB2 audit tables

System requirements for downloading the Web material

We recommend the following system configuration:

Hard disk space	500 MB minimum
Operating System	Windows
Processor	468 or later
Memory	512 MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 268. Note that some of the documents referenced here may be available in softcopy only.

- *DB2 UDB V8.2 on the Windows Environment*, SG24-7102

Other publications

These publications are also relevant as further information sources:

IBM - DB2 9.5

- *What's New*, SC23-5869-00
- *Administrative API Reference,* SC23-5842-00
- *Administrative Routines and Views,* SC23-5843-00
- *Call Level Interface Guide and Reference, Volume 1,* SC23-5844-00
- *Call Level Interface Guide and Reference, Volume 2,* SC23-5845-00
- *Command Reference*, SC23-5846-00
- *Data Movement Utilities Guide and Reference*, SC23-5847-00
- *Data Recovery and High Availability Guide and Reference*, SC23-5848-00
- *Data Servers, Databases, and Database Obj*ects Guide, SC23-5849-00
- *Database Security Guide*, SC23-5850-00
- *Developing ADO.NET and OLE DB Applications*, SC23-5851-00
- *Developing Embedded SQL Applications*, SC23-5852-00
- *Developing Java Applications*, SC23-5853-00
- *Developing Perl and PHP Applications*, SC23-5854-00

- *Developing User-defined Routines (SQL and External)*, SC23-5855-00
- *Getting Started with Database Application Development*, GC23-5856-00
- *Getting Started with DB2 installation and administration on Linux and Windows*, GC23-5857-00
- *Internationalization Guide*, SC23-5858-00
- *Message Reference, Volume 1*, GI11-7855-00
- *Message Reference, Volume 2*, GI11-7856-00
- *Migration Guide*, GC23-5859-00
- *Net Search Extender Administration and User's Guide*, SC23-8509-00
- *Partitioning and Clustering Guide*, SC23-5860-00
- *Query Patroller Administration and User's Guide*, SC23-8507-00
- *Quick Beginnings for IBM Data Server Clients*, GC23-5863-00
- *Quick Beginnings for DB2 Servers*, GC23-5864-00
- *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC23-8508-00
- *SQL Reference, Volume 1*, SC23-5861-00
- *SQL Reference, Volume 2*, SC23-5862-00
- *System Monitor Guide and Reference*, SC23-5865-00
- *Troubleshooting Guide*, GI11-7857-00
- *Tuning Database Performance*, SC23-5867-00
- *Visual Explain Tutorial*, SC23-5868-00
- *Workload Manager Guide and Reference*, SC23-5870-00
- *pureXML Guide*, SC23-5871-00
- *XQuery Reference*, SC23-5872-00

IBM - DB2 9

- *What's New*, SC10-4253
- *Administration Guide: Implementation*, SC10-4221
- *Administration Guide: Planning*, SC10-4223
- *Administrative API Reference*, SC10-4231
- *Administrative SQL Routines and Views*, SC10-4293
- *Administration Guide for Federated Systems,* SC19-1020
- *Call Level Interface Guide and Reference, Volume 1*, SC10-4224

- *Call Level Interface Guide and Reference, Volume 2*, SC10-4225
- *Command Reference,* SC10-4226
- *Data Movement Utilities Guide and Reference*, SC10-4227
- *Data Recovery and High Availability Guide and Reference*, SC10-4228
- *Developing ADO.NET and OLE DB Applications*, SC10-4230
- *Developing Embedded SQL Applications*, SC10-4232
- *Developing Java Applications*, SC10-4233
- *Developing Perl and PHP Applications*, SC10-4234
- *Developing SQL and External Routines*, SC10-4373
- *Getting Started with Database Application Development*, SC10-4252
- *Getting started with DB2 installation and administration on Linux and Windows*, GC10-4247
- *Message Reference Volume 1*, SC10-4238
- *Message Reference Volume 2*, SC10-4239
- *Migration Guide*, GC10-4237
- *Performance Guide*, SC10-4222
- *Query Patroller Administration and User's Guide*, GC10-4241
- *Quick Beginnings for DB2 Clients*, GC10-4242
- *Quick Beginnings for DB2 Servers*, GC10-4246
- *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC18-9749
- *SQL Guide*, SC10-4248
- *SQL Reference, Volume 1,* SC10-4249
- *SQL Reference, Volume 2,* SC10-4250
- *System Monitor Guide and Reference,* SC10-4251
- *Troubleshooting Guide*, GC10-4240
- *Visual Explain Tutorial*, SC10-4319
- *XML Extender Administration and Programming*, SC18-9750
- *XML Guide*, SC10-4254
- *XQuery Reference*, SC18-9796
- *DB2 Connect User's Guide*, SC10-4229
- *DB2 9 pureXML Guide*, SG24-7315

- *Quick Beginnings for DB2 Connect Personal Edition*, GC10-4244
- *Quick Beginnings for DB2 Connect Servers*, GC10-4243

Online resources

These Web sites are also relevant as further information sources:

- DB2 Information Center

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp
- Database and Information Management home page

  http://www.ibm.com/software/data/
- DB2 Universal Database home page

  http://www.ibm.com/software/data/db2/udb/
- DB2 Technical Support

  http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/report
- DB2 online library

  http://www.ibm.com/db2/library

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

DB2 Security and Compliance Solutions for Linux, UNIX, and Windows

IBM®

DB2 Security and Compliance Solutions for Linux, UNIX, and Windows

Redbooks®

Understand DB2 security concepts and technologies

Learn security implementation by examples

Protect your data with DB2 security features

In this IBM Redbooks publication we discuss the existing and new DB2 security features introduced in DB2 9.5 for Linux, UNIX, and Windows. These enriched DB2 security features provide you with the capability to protect your data and comply with regulatory requirements.

We describe how you can control data access through DB2 authentication and authorization functions. The role feature provides new options for tighter security, more granularity, and flexibility in administrating data access. Data encryption offers the capability to protect sensitive data in the database, critical database files, and data transferred over the network. Trusted contexts and trusted connections allow you to have more control over when a data access privilege becomes available to a user. Using label-based access control (LBAC), you can control read and write access of users to individual rows and columns at the table level. The enhanced audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events for analysis and identifying system misuse.

At the end, we introduce other DB2 data security solutions including IBM Database Encryption Expert, DB2 Audit Management Expert, and IBM Optim Enterprise Data Management.

SG24-7555-00 ISBN 0738485810