Oracle9i Label Security

An Oracle White Paper
January 2002

EXECUTIVE OVERVIEW

Oracle9i Label Security addresses a variety of access control needs ranging from simple company by company data separation in a hosted environment to complex data sharing requirements between a parent company and its subsidiaries. Current users of Oracle9i Label Security include companies specializing in healthcare, energy and national security. Oracle9i Label Security simplifies Oracle Virtual Private Database. Oracle9i Label Security allows the security officer or application administrator to focus on managing user security clearances and not worry about the underlying enforcement mechanism. Data can be automatically assigned a sensitivity label or labeled using an application specific business rule. Oracle9i Label Security is a commercial off the shelf (COTS) product which decreases application development costs and increases data security.

INTRODUCTION

This paper outlines a methodology applicable to both Oracle9i Label Security and Virtual Private Database.

This paper outlines a series of steps which can be used standalone or integrated into an existing software design methodology to help ensure Oracle9i Label Security integrates seamlessly with an existing application or new software development design. The focus of this paper is Oracle9i Label Security, however, the general methodology is also applicable to security policies written using the Oracle Virtual Private Database fine grained access control feature.

UNDERSTANDING THE TECHNOLOGY

Database object privileges, roles, views, and stored procedures are typically sufficient to address most application access control requirements.

Neither Oracle9i Label Security or Oracle Virtual Private Database fine grained access control are designed to replace the traditional access control mechanisms found in Oracle. Database object privileges, roles, views, and stored procedures are typically sufficient to address most application access control requirements. Oracle9i Label Security provides an out-of-the-box row level security solution for situations where data stored in one or more tables must be restricted based on its sensitivity or ownership. In other words, the decision to allow access to a row is determined at the row level and not the table level. This type of access control is difficult to implement programmatically and increases application complexity.

Historical Approaches to Row Level Security

Historically, the three approaches used to provide row level security have involved one or more of the following:

- Application views
- Programming logic embedded in the application
- Physical separation using one or more databases

Application Views

Database views do provide the ability to filter data. However, the number of views required is sometimes large and directing application users to the correct view becomes a management burden.

Application views can be found in nearly all software applications. The number of views usually depends on the age of the application. Application complexity tends to increase due to unforeseen security requirements or poor application design. Database views do provide the ability to filter data.

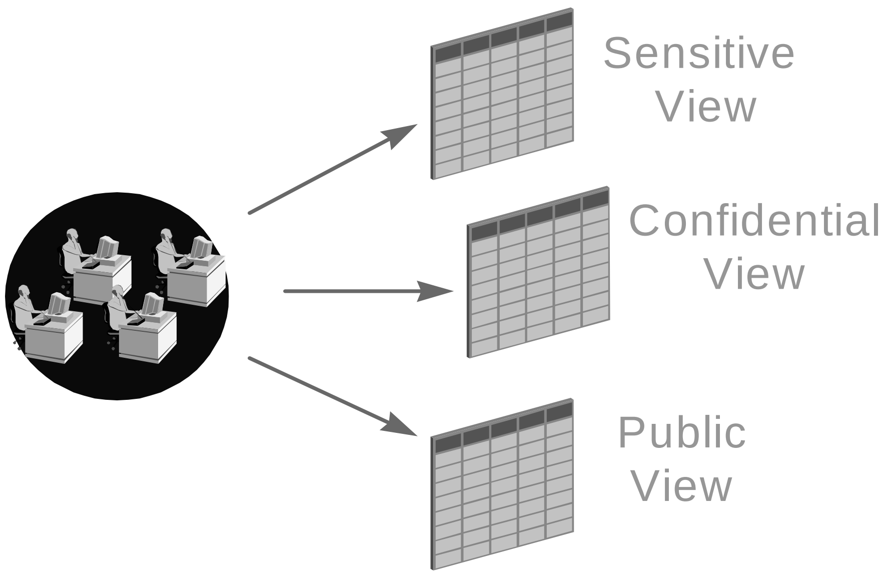


Figure 1. Restricting Access Using Database Views

However the number of views required is sometimes large and directing application users to the correct view becomes a management burden. The following is an example of using views to restrict access to three types of data: sensitive data, confidential data and public data.

Example Static View Definitions

```
Create view sensitive_employee as

select * from employee where sensitivity = 'Sensitive'
or 'Confidential' or 'Public';

Create view confidential_employee as

select * from employee where sensitivity =
'Confidential' or 'Public';

Create view public_employee as

select * from employee where sensitivity = 'Public';
```

Application Programmatic Logic Approach

Typically numerous additional tables must be built to maintain an application user's label authorizations.

Applications can control SQL statements submitted through the application. However, SQL statements submitted outside the application using a utility such as SQL*Plus can't be controlled. If an application is rewriting SQL statements to restrict access based on data sensitivity then typically numerous additional tables must be built and maintained to manage information related to the authorizations of the application user to view sensitive information.

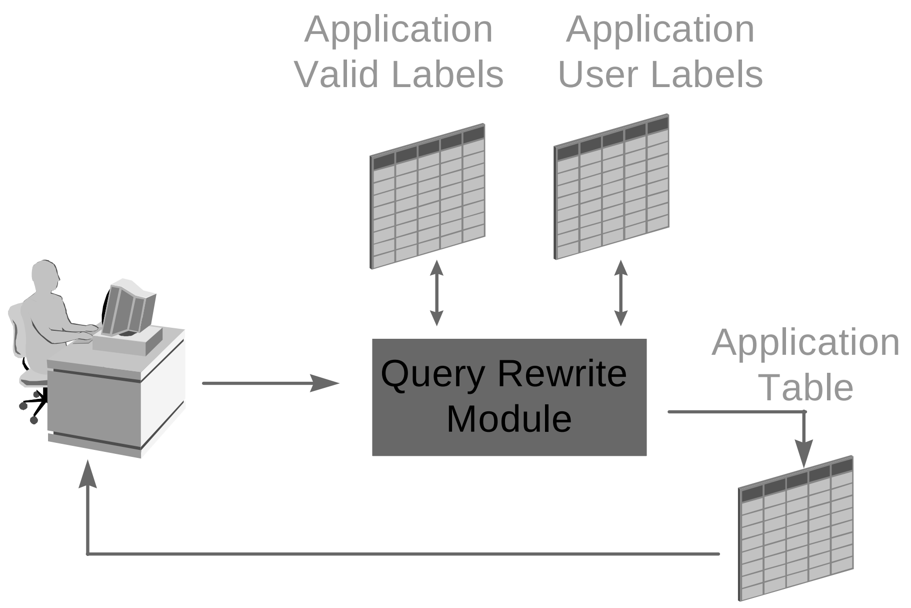


Figure 2. Restricting Access Using Application Logic

Multiple Database Approach

The number of databases required is equal to the number of data sensitivities

Sensitive data can also be protected by using dedicated databases to manage each sensitivity. However, creating a separate database for each sensitivity has several problems. First, the number of databases required is equal to the number of data sensitivities. Secondly, the overhead created by running multiple databases in terms of memory, processing power and backup storage is substantially increased. Thirdly, the costs associated with managing a single database are multiplied by the number of new databases. Finally, viewing information across multiple databases requires distributed queries and application logic.

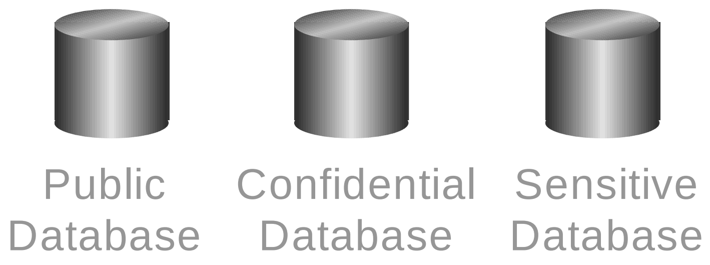


Figure 3. Restricting Access Using Separate Databases

ORACLE9I LABEL SECURITY

Oracle9i Label Security provides an integrated database solution for controlling access to data based on sensitivity labels

Oracle9i Label Security simplifies the three designs outlined in the previous section. Oracle9i Label Security provides an integrated database solution for controlling access to data based on sensitivity labels. Introducing Oracle9i Label Security into an existing application has the following benefits:

- Simplifies the application
- Increases security by moving access control into the database
- Creates a competitive advantage over competing applications
- Introduces new and powerful application functionality

Application Table

Project	Location	Sensitivity Label
AX703	Chicago	Public
B789C	Dallas	Sensitive: Executive Only
JFS845	Chicago	Secret: National Security
SF78SD	Miami	Internal : Global Warming

Figure 4. Example Application Table

Oracle9i Label Security Access Mediation

Oracle9i Label Security access mediation works by comparing a sensitivity label assigned to a data row with label authorizations assigned to the user.

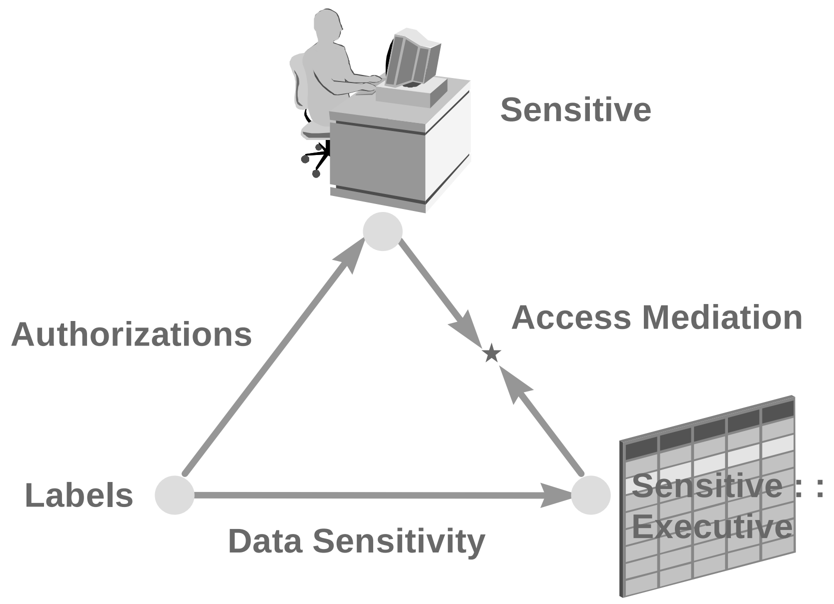


Figure 5. Oracle9i Label Security Access Mediation

A Closer Look at Sensitivity Labels

Sensitivity labels are central to Oracle9i Label Security. Sensitivity labels are what determine an application user's ability to view and update application data. Sensitivity labels provide sophisticated controls which are not possible with traditional object level privileges. For example, suppose an order entry application has a security policy which states that the application must be capable of limiting access to purchase orders labeled company sensitive? By default, giving an application user the SELECT privilege on the purchase orders table will allow the user to view all information. One approach to solving this requirement is to create two database views. The first view will exclude all the purchase orders deemed company sensitive and the second will include all the purchase orders. This approach is problematic because the security policy may change to include new levels of sensitivity. In addition, application users will need to be assigned the correct enterprise role depending on their authorization to view company sensitive information. Sensitivity labels solve this security requirement and eliminate the need for additional views. Oracle9i Label Security sensitivity labels contain three components: a single hierarchical level or classification, one or more horizontal compartments or categories and one or more groups.

Label Components

Level -- The level is a hierarchical component which denotes the sensitivity of the data. A typical government organization might define levels confidential, sensitive and highly sensitive. However, there is no requirement to define more than one level. For example, a commercial organization might define a single level for company confidential data or application hosting requirements

Compartment - The compartment component is sometimes referred to as a category and is non hierarchical. Typically one or more compartments are defined to segregate data. For example, a compartment might be defined for an ongoing strategic initiative or map to a hosted application subscriber. Data related to the initiative can be labeled with the newly defined compartment. Oracle Label Security supports up to 9999 unique compartments.

Group - The group component is used to record ownership and can be used hierarchically. For example, two groups called Senior VP and Manager can be created and subsequently assigned as children of the CEO group, creating an ownership tree. Labels can be composed of a standalone level component or a level component can be combined with compartments, groups or both.

External Representation

The external representation of a label is composed of the three label components, separated by a semicolon. The label "Confidential : Acquisitions : Asia" is composed of the following three label components:

Level = Confidential

Compartment = Acquisitions

Group = Asia

Releasabilities

Releasabilities add even more flexibility to the Oracle9i Label Security access control capabilities.

Release 2 of Oracle9i Label Security supports releasabilities, adding even more flexibility to the Oracle9i Label Security access control capabilities. Releasabilities have historically been used in government organizations to control the dissemination of data. Release 2 of Oracle9i Label Security makes this technology available to commercial and government organizations on widely used, commercial operating systems. In the past this technology has only been available on highly specialized operating systems.

Oracle9i Label Security uses inverse groups to indicate releasability of information: they are used to mark the dissemination of data. When you add an inverse group to a data label, the data becomes less classified. For example, a user with inverse groups UK, US cannot access data which only has inverse group UK. Adding US to that data makes it accessible to all users with the inverse groups UK, US. When you assign releasabilities to a user, you mark the communication channel to the user. For data to flow across the communication channel, the data releasabilities must dominate the releasabilities assigned to the user. In other words, releasabilities assigned to a data record must contain all the releasabilities assigned to a user. The advantage of releasabilities lies in their power to broadly disseminate information. Releasing data to the entire marketing organization becomes as simple as adding the marketing releasability to the data record. The term inverse group is used because an administrator can now create an Oracle Label Security policy which uses the access control logic provided by standard groups or decide to create the policy using inverse group access control logic.

Comparing Standard Groups and Inverse Groups

Groups in Oracle Label Security identify organizations which own or access data. Like standard groups, inverse groups control the dissemination of information. However, the behavior of inverse groups differs from Oracle Label Security standard group behavior. By default, all policies created in Oracle Label Security use the standard group behavior. When you include inverse groups in a data label, the effect is similar to assigning label compartment authorizations to a user. When Oracle Label Security evaluates whether a user can view a row of data assigned a label with inverse groups, it checks to see whether the data, not the user, has the appropriate group authorizations: does the data have all the inverse groups assigned to the user? With standard groups, by contrast, Oracle Label Security checks to see whether a user is authorized for at least one of the groups assigned to a row of data. Consider a policy which contains 3 standard groups: Eastern, Western, and Southern. User1's label authorizations include the groups Eastern and Western. Assuming User1 has been assigned the appropriate level and compartment authorizations in the policy, then:

- With standard Oracle Label Security groups, User1 can view all data records that have the group Eastern, or the group Western, or both Eastern and Western.
- With inverse groups, User1 can only view data records that have, at a minimum, all the groups assigned to the user: that is, both Eastern and Western. She cannot view records that have only the Eastern group, only the Western group, or that have no groups at all.

When using standard groups, a hierarchical relationship can be created by designating a parent for each group. However, designating hierarchical relationships between inverse groups is not practical because of the access control logic associated with the concept of releasabilities. Therefore when a policy is created and the inverse group option is specified, the ability to designate a parent for a particular group has been disabled. A policy can't use both standard groups and inverse groups. The decision to use standard groups versus inverse groups needs to be decided based on business needs and before the policy is created. The examples contained in this whitepaper use standard groups.

Oracle Virtual Private Database

Oracle Virtual Private Database is a term used for several powerful Oracle9i features: fine grained access control, application context and global application context. Fine grained access control allows an application developer to write his own security policy and apply it to application table or view. SQL statements or reports accessing the application table are transparently modified based on the security policy. Typically, the security policy utilizes the application context feature of Oracle. Application context is basically a named area of computer memory which is assigned a name by the application and allocated by Oracle. Depending on the complexity of the security policy required, the programming and design work associated with fine grained access control and application context may be significant.

Oracle9i Label Security uses the Oracle Virtual Private Database features to provide a wealth of security functionality to the administrator.

Oracle9i Label Security is designed for both government and commercial organizations

Oracle9i Label Security Benefits:

- No programming is required, security package is provided out-of-the-box
- Oracle9i Label Security policy options give administrator's the ability to customize enforcement based on their organizations specific needs
- Designed for both government and commercial organizations
- SQL predicates can be added to the access mediation process, extending access control beyond sensitivity labels

Incorporating Oracle9i Label Security

Oracle9i Label Security provides a fast forward solution for the development and deployment of applications

Oracle9i Label Security provides a fast forward solution for the development and deployment of applications. No design and development work is required. However, one important piece of work which Oracle9i Label Security does not eliminate is the process of looking at the application and determining how and where to apply Oracle9i Label Security. Most, if not all, applications on the market today weren't designed to work with sensitivity labels or row level security. In some cases, hundreds of views may exist in the application which restrict access based on sensitivity labels assigned to data. While incorporating Oracle9i Label Security during design phase of an application is easiest, the analysis steps are the same for applications which are already on the market.

Step 1: Analyzing the Application Scheme

In most cases, a small percentage of the tables in an application will require an Oracle9i Label Security policy

Analyzing the scheme means identifying the tables which need an Oracle9i Label Security policy applied to them. This is best accomplished with the assistance of an application administrator or developer who has intimate knowledge of the application scheme. In most cases, a small percentage of the tables in an application will require an Oracle9i Label Security policy. For example, tables which store lookup values or constants usually don't need to be protected with Oracle9i Label Security.

Projects
Sales

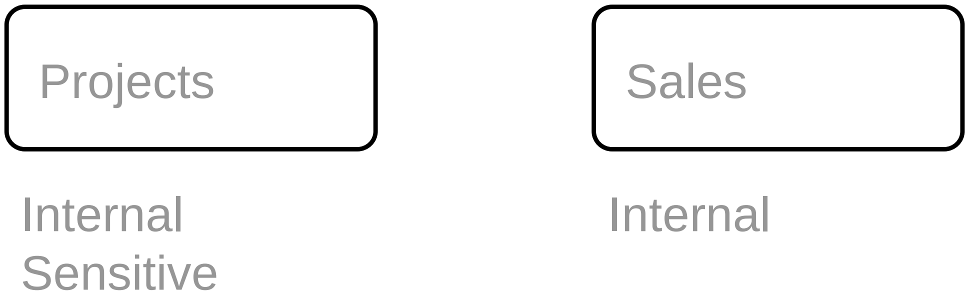
Step 2: Analyzing the Data Levels

Once the candidate tables have been identified, the data contained in the tables needs to be evaluated. The assistance of someone other than the application administrator or developer may be required. A manager who generates reports which access the table or someone who has broad familiarity with business operations can provide valuable assistance with this stage of the analysis. Data levels refer to the sensitivity level of the data. *Sensitive* is an example of a data level. Additional examples include *unclassified, public,* and *highly sensitive.* However, analysis may determine that an application table has only one data level. It's recommended that application data which may be stored in the table in the future be considered as well. This will create a robust set of label definitions in the Oracle9i Label Security access control data dictionary.

Data Label = Internal : Alpha, Beta : US, UK

User Label = Sensitive : Alpha, Beta : UK

If a data record is assigned a sensitivity label whose level component is at or below the user's session label sensitivity level, then a user attempting to read the record will be allowed to view the record. Note that this access rule is true if the user also has the appropriate compartments and groups which are discussed in the next two sections and has not been assigned any special Oracle9i Label Security privileges.



Step 3: Analyzing the Data Groups

As with data levels, the assistance of someone with broad familiarity with business operations can provide valuable assistance with this stage of the analysis

Groups are a component of the label which are handy for controlling access to data by organization, region or data ownership. As with data levels, the assistance of someone with broad familiarity with business operations can provide valuable assistance with this stage of the analysis. For example, if the application is a sales application, access to the sales data may be controlled on a country by country basis or on a regional basis. Examples of groups include *United States, Europe, Asia, Manager, Human Resources,* and *Legal.* If a data record is assigned a sensitivity label with multiple compartments and multiple groups, a user attempting to read the record must have all of the compartments found in the data sensitivity label in their session label and at least one of the groups. Note that since groups can be hierarchical in nature, a user could have the parent of one of the groups in the sensitivity label assigned to the data record and still be able to

Data Label = Sensitive : Alpha, Beta : US, UK

User Label = Sensitive : Alpha, Beta : UK

Note that this access rule is true if a user also has the appropriate compartments and his sensitivity level dominates the sensitivity level of the data record and he has not been assigned any special Oracle9i Label Security privileges.

Step 4: Analyzing the Data Compartments

A commercial application may find that data groups are sufficient to provide the necessary level of access control

Data compartments are primarily used in government and defense environments. A commercial application may find that data groups are sufficient to provide the necessary level of access control. The most significant difference between data compartments and data groups involves Oracle9i Label Security access mediation. If a data record is assigned a sensitivity label with multiple compartments and multiple groups, a user attempting to read the record must have all of the compartments found in the data sensitivity label in their session label and at least one of the groups.

Data Label = Internal : Alpha, Beta : US, UK

User Label = Sensitive : Alpha, Beta : UK

Note that this access rule is true if a user also has the appropriate data groups and his sensitivity level dominates the sensitivity level of the data record and he has not been assigned any special Oracle9i Label Security privileges.

Step 5: Analyzing the user population

Analyzing the user population requires separating the users into one or more designated user types. For example, a user might be designated as a regular user, highly privileged user, or administrative user. This process may require the assistance of managers and security administrators. After the user population has been separated into one or more user types, a comparison needs to be performed between the data levels identified in step 2 and the authorized sensitivity levels of the user population.. These need to correspond correctly for each table identified during scheme analysis in step 1. In addition, the organizational structure of the user population needs to be compared with the data groups identified in step 3. These do not need to correspond exactly on the first comparison. Adjustments may be necessary as access requirements are better understood.

Step 6: Analyzing special authorizations

Oracle9i Label Security has several special authorizations which can be assigned to users. In this step, examine the highly privileged and administrative users and determine what if any of the Oracle9i Label Security special authorizations should be assigned to the user. In general, regular users do not require any special authorizations. Please refer to appendix A for a complete list of the Oracle9i Label Security special authorizations.

Step 7: Review and Document

This step is crucial for continuity across the enterprise and the resulting document should become part of the enterprise security policy

The final step before starting the physical implementation is to review and document the information gathered. This step is crucial for continuity across the enterprise and the resulting document should become part of the enterprise security policy. Include tables, graphs and a narrative discussion to fully explain the results of the analysis. For example, the document should include a list of scheme tables which need to be protected and corresponding justification.

Analysis Methodology

This technique can help identify operational issues prior to the actual implementation of Oracle9i Label Security.

During the analysis it may be helpful to create a matrix which compares the information collected in step 5 with information collected about the actual data. This technique can help identify operational issues prior to the actual implementation of Oracle9i Label Security. For example, this type of analysis can identify the authorizations required for a specific job function.

Table	User Data	I	S	S:A:US	S:A,B:US,UK
Sales	I::UK	No Acess	No Acess	No Acess	Acess
	I::US	No Acess	No Acess	Acess	Acess
Projects	I	Acess	Acess	Acess	Acess
	S	No Acess	Acess	Acess	Acess
	S:A:US	No Acess	No Acess	Acess	Acess
	S:B:UK	No Acess	No Acess	No Acess	Acess
	S:A,B,US	No Acess	No Acess	No Acess	Acess

Implementation

The API can be used in place of Oracle9i Policy Manager to perform all Oracle9i Label Security operations

The implementation can be performed using Oracle9i Policy Manager or the Oracle9i Label Security API. The API can be used in place of Oracle9i Policy Manager to perform all Oracle9i Label Security operations.

- First, perform the analysis steps outlined above
- Second, create the Oracle9i Label Security Policy
- Third, define data label components including levels, compartments and groups using information gathered during analysis
- Fourth, create valid data labels for the policy using the components defined in step three
- Fifth, assign user population label authorizations and appropriate Oracle9i Label Security special authorizations using data gathered during analysis
- Sixth, apply the policy to the application table. If legacy data exists in the table, it may be necessary to apply the policy to the table with the no control enforcement option initially. After the policy is applied to the table, the sensitivity label will be empty and no data will be visible. Please refer to the section on labeling legacy data
- Seventh, if necessary update legacy data with appropriate data labels
- Eight, alter the Oracle9i Label Security policy and set appropriate policy enforcement options

Oracle9i Label Security Releasabilities

Releasabilities add even more flexibility to the Oracle9i Label Security access control capabilities.

Release 2 of Oracle9i Label Security supports releasabilities, adding even more flexibility to the Oracle9i Label Security access control capabilities. Releasabilities have historically been used in government organizations to control the dissemination of data. Release 2 of Oracle9i Label Security makes this technology available to commercial and government organizations on widely used, commercial operating systems. Historically, this technology has only been available on highly specialized operating systems. Oracle9i Label Security uses inverse groups to indicate releasability of information: they are used to mark the dissemination of data. When you add an inverse group to a data label, the data becomes less classified. For example, a user with inverse groups UK, US cannot access data which only has inverse group UK. Adding US to that data makes it accessible to all users with the inverse groups UK, US. When you assign releasabilities to a user, you mark the communication channel to the user. For data to flow across the communication channel, the data releasabilities must dominate the releasabilities assigned to the user. In other words, releasabilities assigned to a data record must contain all the releasabilities assigned to a user. The advantage of releasabilities lies in their power to broadly disseminate information. Releasing data to the entire marketing organization becomes as simple as adding the Marketing releasability to the data record. The term inverse group is used because an administrator can now create an Oracle Label Security policy which uses the access control logic provided by standard groups or decide to create the policy using inverse group access control logic.

Comparing Standard Groups and Inverse Groups

Groups in Oracle Label Security identify organizations which own or access data. Like standard groups, inverse groups control the dissemination of information. However, the behavior of inverse groups differs from Oracle Label Security standard group behavior. By default, all policies created in Oracle Label Security use the standard group behavior. When you include inverse groups in a data label, the effect is similar to assigning label compartment authorizations to a user. When Oracle Label Security evaluates whether a user can view a row of data assigned a label with inverse groups, it checks to see whether the data, not the user, has the appropriate group authorizations: does the data have all the inverse groups assigned to the user? With standard groups, by contrast, Oracle Label Security checks to see whether a user is authorized for at least one of the groups assigned to a row of data. Consider a policy which contains 3 standard groups: Eastern, Western, and Southern. User1's label authorizations include the groups Eastern and Western. Assuming User1 has been assigned the appropriate level and compartment authorizations in the policy, then:

- With standard Oracle Label Security groups, User1 can view all data records that have the group Eastern, or the group Western, or both Eastern and Western.
- With inverse groups, User1 can only view data records that have, at a minimum, all the groups assigned to the user: that is, both Eastern and Western. She cannot view records that have only the Eastern group, only the Western group, or that have no groups at all.

When using standard groups, a hierarchical relationship can be created by designating a parent for each group. However, designating hierarchical relationships between inverse groups is not practical because of the access control logic associated with the concept of releasabilities. Therefore when a policy is created and the inverse group option is specified, the ability to designate a parent for a particular group has been disabled. A policy can't use both standard groups and inverse groups. The decision to use standard groups versus inverse groups needs to be decided based on business needs and before the policy is created. The examples contained in this whitepaper use standard groups.

Labeling Legacy Data

Once an Oracle9i Label Security policy is applied to an application table with read control, no rows will be visible until valid data labels have been assigned to each data row. Several methods exist for labeling data legacy data.

The first method for labeling legacy data is to simply issue an update statement against the base table.

```
UPDATE SALES SET SECLAB = char_to_label:('FINANCE','S')
WHERE REGION_ID = 104;
```

This statement updates the SALES table and sets the policy label column SECLAB equal to the internal label tag defined for SENSITIVE in the FINANCE policy and SALES column REGION_ID is equal to 104.

The second method for labeling legacy data is to switch database connections during the data load.

```
CONNECT US_SALES_MGR
INSERT  INTO SALES (Col1, Col2, Col3) VALUES ('ACME',......);
CONNECT EU_SALES_MGR
INSERT INTO SALES (Col1, Col2, Col3) VALUES ('WIDGET',......);
```

If the policy applied to the SALES table includes the LABEL_DEFAULT enforcement option, the users default ROWLABEL value will be used to set the SECLAB column.

The third method is to write a labeling function using PL/SQL. Oracle9i Label Security label functions are written in PL/SQL. An example of a labeling function can be found in the Oracle9i Label Security administrator's guide as well as in the demo file shipped with the Oracle Enterprise Edition. On UNIX operating systems the demo file is called olsdemo.sql and can be found in the directory $ORACLE_HOME/rdbms/demo. ORACLE_HOME is an environment variable which points to the install location of the Oracle9i Enterprise Edition. The labeling function can be added to a policy using an Oracle9i Label Security API.

The label function could also be written to return an Oracle9i Label Security label tag. This type of function should not be added to an Oracle9i Label Security policy because it doesn't the correct data type. However, the function could be called in the context of an update statement on the target application table.

```
UPDATE SALES SET SECLAB = My_Function(sales.region_id);
```

The fourth method for labeling legacy data is to write a stored procedure which updates the application table column associated with the Oracle9i Label Security policy.

Hiding the Oracle9i Label Security policy column

Oracle9i Label Security has an option to hide the policy column when the policy is added to an application table.

Oracle9i Label Security requires a column name to be specified when a policy is created. When the policy is applied to an application table, the column name specified during policy creation is appended to the application table. Oracle9i Label Security has an option to hide the policy column when the policy is added to an application table. Hiding the column causes the column to not be displayed during a SQL Plus table describe operation. Hiding the column also allows SQL statements which do not specify column names to continue functioning even though a new column has been added to the application table specified in the SQL statement.

It's important to note that the Oracle9i Label Security policy column can pre-exist in an application table prior to application of an Oracle9i Label Security policy. To take advantage of this the application table column type must be number(10). Applications can be designed with an Oracle9i Label Security column built-in.

Installing Oracle9i Label Security with Oracle9i

Oracle9i Label Security is installed by choosing the custom installation option during the Oracle9i Enterprise Edition installation.

Oracle9i Label Security is available for all Oracle9i Enterprise Edition platforms.

Oracle9i Label Security is a security option for the Oracle9i Enterprise Edition and ships on the Oracle9i Enterprise Edition CD. Oracle9i Label Security is installed by choosing the custom installation option during the Oracle Enterprise Edition installation. After installing the software, the database configuration tool must be run to create the necessary Oracle9i Label Security data dictionary objects. Oracle9i Label Security is available for all Oracle9i Enterprise Edition platforms.

The initial database administration account for Oracle9i Label Security is an account called *LBACSYS*. This account is automatically locked during the installation of Oracle9i Label Security. The account can be unlocked by a database administrator.

Note, Oracle9i Label Security 8.1.7 is available for Sun Sparc Solaris (32-bit) only and ships on a separate CD with the Oracle8i Enterprise CD.

Oracle9i Policy Manager

Oracle9i Label Security policies can be managed using Oracle9i Policy Manager by connecting as the user LBACSYS or another user with appropriate privileges.

Oracle9i Policy Manager is the new GUI administration tool for both Oracle9i Label Security and the Oracle Virtual Private Database features. On UNIX systems, you can type the command 'oemapp opm'. This command will launch the Oracle9i Policy Manager tool.

Oracle9i Label Security policies can be managed using Oracle9i Policy Manager by connecting as the user LBACSYS or another user with appropriate privileges.

Oracle9i Label Security Delegated Administration

Users can be given authorizations to create and manage Oracle9i Label Security policies.

The account LBACSYS is the primary administration account for Oracle9i Label Security. However, other users can be given authorizations to create and manage Oracle9i Label Security policies. When an Oracle9i Label Security policy is created, a new database role is created. The name of the database role is *policyname*_DBA. In the following code, the user LBACSYS creates a policy and gives the user HR_INFOSEC authorizations to manage policy label components and label authorizations.

```
CONNECT LBACSYS

EXECUTE SA_SYSDBA.CREATE_POLICY('HR_SECURITY', 'HR_LABEL');

GRANT HR_DBA TO HR_INFOSEC;
GRANT EXECUTE ON sa_components TO sub_admin;
GRANT EXECUTE ON sa_user_admin TO sub_admin;
GRANT EXECUTE ON sa_label_admin TO sub_admin;
GRANT EXECUTE ON sa_policy_admin TO sub_admin;
GRANT EXECUTE ON sa_audit_admin TO sub_admin;
```

Enforcement Exemptions

Oracle virtual private database fine grained access control policies and Oracle9i Label Security policies are not enforced during DIRECT path export.

Oracle virtual private database fine grained access control policies and Oracle9i Label Security policies are not enforced during DIRECT path export. Also, Virtual private database policies and Oracle9i Label Security policies cannot be applied to objects in schema SYS. As a consequence, the SYS user and users making a DBA-privileged connection to the database (for example, CONNECT/AS SYSDBA) do not have VPD or Oracle9i Label Security policies applied to their actions. Database administrators need to be able to administer the database. It would not suffice to export part of a table due to a VPD policy being applied. Database users who are granted the Oracle9i EXEMPT ACCESS POLICY privilege, directly or through a database role, are exempt from Virtual Private Database and Oracle9i Label Security enforcement. The users are exempt from Virtual Private Database and Oracle9i Label Security enforcement regardless of the export mode, application, or utility used to extract data from the database. EXEMPT ACCESS POLICY privilege is a powerful privilege and should be carefully managed. The EXEMPT ACCESS POLICY privilege does not affect the enforcement of object privileges such as SELECT, INSERT, UPDATE, and DELETE. These privileges are enforced even if a user has been granted the EXEMPT ACCESS POLICY privilege.

Performance Tuning

The percentage of unique labels compared to the number of data rows in an application table is usually very low.

Oracle9i Label Security is highly optimized. Oracle recommends creating a bitmap index on the Oracle9i Label Security policy column. The percentage of unique labels compared to the number of data rows in an application table is usually very low. This fact makes the policy label column an ideal candidate for bitmap indexes.

Partitioning

The Oracle Partitioning option can be used in conjunction with Oracle9i Label Security to partition data based on sensitivity.

The Oracle Partitioning option can be used in conjunction with Oracle9i Label Security to partition data based on sensitivity.

```
CREATE TABLE EMPLOYEE
    EMPNO NUMBER(10) CONSTRAINT PK_EMPLOYEE PRIMARY KEY,
    ENAME VARCHAR2(10),
    JOB VARCHAR2(9),
    MGR NUMBER(4),
    HIREDATE DATE,
    SAL NUMBER(7,2),
    COMM NUMBER(7,2),
    DEPTNO NUMBER(4),
    HR_LABEL NUMBER(10))
    TABLESPACE PERF_DATA
    STORAGE (initial 2M
    NEXT 1M
    MINEXTENTS 1
    MAXEXTENTS unlimited)
    PARTITION BY RANGE (hr_label)
    (partition sx1 VALUES LESS THAN (2000) NOLOGGING,
     partition sx2 VALUES LESS THAN (3000),
     partition sx3 VALUES LESS THAN (4000) );
```

Oracle Advanced Security Enterprise Users

Directory users mapped to a common database schema will, by default, assume the label authorization profile established for the common database schema name.

Oracle9i Label Security stores policy definitions, valid labels and user label authorizations in the Oracle9i Enterprise Edition database. Oracle9i Label Security provides label authorization granularity to the shared schema. Directory users mapped to a common database schema will, by default, assume the label authorization profile established for the common database schema name.

When assigning Oracle9i Label Security policy label authorizations to a user, the user name specified doesn't have to be a database user. For example, the user ACME_MGR specified in the following Oracle9i Label Security command doesn't have to be a valid database user.

```
SQL> execute sa_user_admin.set_user_labels
('PRIVACY','ACME_MGR','HIGHLY_SENSITIVE::USWEST');
```

SET_ACCESS_PROFILE Command

Oracle9i Label Security provides the ability for an authorized user to assume an Oracle9i Label Security authorization profile. The following code uses the Oracle9i Label Security SET_ACCESS_PROFILE command to assume the Oracle9i Label Security profile called SCOTT for the Oracle9i Label Security policy PRIVACY.

```
SQL*Plus  username

SQL>  execute sa_session.set_access_profile (‘PRIVACY’,’ACME_MGR’);

SQL>  select * from emp;
```

The above example assumes that the user connecting to the database has the appropriate Oracle9i Label Security authorizations necessary to execute the SET_ACCESS_PROFILE command. By executing the SET_ACCESS_PROFILE command, he or she assumes the Oracle9i Label Security profile called SCOTT. When the SQL statement

```
SQL>  SELECT * FROM EMP;
```

is executed, Oracle will first verify the database user has the appropriate object privileges on the table EMP. Second, for each row in the EMP table returned by the query, Oracle will further mediate access based on the Oracle9i Label Security label authorizations defined for the profile SCOTT and the sensitivity label assigned to the row. This technique is used to assume the Oracle9i Label Security profile of users who are not defined in the database or in the directory. As noted earlier, Oracle9i Label Security does not enforce a mapping between database users and the user name specified when establishing label authorizations. For example, if an application uses a proxy account architecture, the application can utilize one of the many Oracle SYS_CONTEXT variables to determine which Oracle9i Label Security profile should be specified in the SET_ACCESS_PROFILE command. For enterprise users the EXTERNAL_NAME SYS_CONTEXT value could be passed to the SET_ACCESS_PROFILE command. The Oracle9i Label Security development team is working on enhancements which will automatically set the Oracle9i Label Security profile based on the user defined in the directory and not the shared scheme name.

Additional Resources

Part 1 of the Oracle9i Label Security Administrator’s Guide is essential reading for the developer or administrator who’s new to Oracle9i Label Security. It provides an overview of label based security, policies, label components, label syntax, access mediation and user authorizations. In addition to the Oracle9i Label Security Administrator’s Guide, an on-demand training class is available on the Oracle Learning Network. The Oracle Technology Network (OTN) has material on Oracle9i Label Security, Oracle9i Policy Manager and other Oracle9i Security features.

APPENDIX A - ORACLE9I LABEL SECURITY SPECIAL USER AUTHORIZATIONS

READ — The READ authorization allows a user to access all data protected by Oracle9i Label Security, however, access mediation is still enforced on UPDATE, INSERT and DELETE operations. Oracle9i Label Security makes no mediation check on SELECT operations.

FULL — The FULL authorization turns off all Oracle9i Label Security access mediation. A user with the FULL authorization can perform SELECT, UPATE, INSERT and DELETE operations with no label authorizations. Note that Oracle SYSTEM and OBJECT authorizations are still enforced. For example, a user must still have SELECT on the application table. The FULL authorization turns off the access mediation check at the individual row level.

WRITEDOWN — The WRITEDOWN authorization allows a user to modify the level component of a label and lower the sensitivity of the label. For example, application data which is labeled *Top Secret: Alpha, Beta* could be changed to *Secret: Alpha, Beta*. This authorization is only applicable to policies which use the label update enforcement option.

WRITEUP — The WRITEUP authorization allows a user to modify the level component of a label and raise the sensitivity of the label. For example, application data which is labeled *Secret: Alpha, Beta* could be changed to *Top Secret: Alpha, Beta*. Note that the *Maximum Level* label authorization assigned to the user would limit modification. This authorization is only applicable to policies which use the label update enforcement option.

WRITEACROSS — The WRITEACROSS authorization allows a user to modify the compartments and groups in a label to any valid compartment and group defined in Oracle9i Label Security for the policy. For example, if application data which is labeled *Secret: Alpha, Beta* could be modified to *Secret: Alpha, Beta, Delta* even though the user was not authorized for the *Delta* compartment. This authorization is only applicable to policies which use the label update enforcement option.

PROFILE ACCESS — The PROFILE ACCESS authorization allows a user to assume the Oracle9i Label Security authorizations of another user. For example, user *Scott* who has access to compartments *A,B, and C* could assume the profile of user *Joe* who has access to compartments *A,B, C and D*. This functionality might be useful in an environment where an application uses a single application account for all application users. The application account could use the PROFILE ACCESS authorization to immediately assume a designated label security profile when an application users connects to the system.

APPENDIX B - ORACLE9I LABEL SECURITY POLICY ENFORCEMENT OPTIONS

Oracle9i Label Security policy enforcement options can be customized for each policy. For example, a Human Resources policy and a Defense policy can exist in the same Oracle database and provide different degrees of protection. The Human Resources application might use the READ CONTROL option and the Defense policy might use the READ CONTROL and WRITE CONTROL options.

READ CONTROL — Applies policy enforcement to all queries; only authorized rows are accessible for SELECT, UPDATE, and DELETE operations.

INSERT CONTROL — Applies policy enforcement to INSERT operations, according to the Oracle9i Label Security algorithm for write access.

UPDATE CONTROL — Applies policy enforcement to UPDATE operations on the data columns within a row, according to the Oracle9i Label Security algorithm for write access.

DELETE CONTROL — Applies policy enforcement to DELETE operations, according to the Oracle9i Label Security algorithm for write access.

WRITE CONTROL — Determines the ability to INSERT, UPDATE, and DELETE data in a row. If this option is set, it enforces INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL.

LABEL DEFAULT — If the user does not explicitly specify a label on INSERT, the users default *row label* value is used. By default, the *row label* value is computed internally by Oracle9i Label Security using the label authorization values specified for the user. A user can set the row label independently, but only to:

A level which is less than or equal to the level of the session label, and greater than or equal to the user's minimum level.

Include a subset of the compartments and groups from the session label, for which the user is authorized to have write access.

LABEL UPDATE — Applies policy enforcement to UPDATE operations that set or change the value of a label attached to a row. The WRITEUP, WRITEDOWN, and WRITEACROSS privileges are only enforced if the LABEL_UPDATE option is set.

LABEL CHECK — Applies READ_CONTROL policy enforcement to INSERT and UPDATE statements to assure that the new row label is read-accessible by the user after and INSERT or UPDATE statement.

NO CONTROL — Applies no enforcement options. A labeling function or a SQL predicate can nonetheless be applied.

APPENDIX C - ORACLE9I LABEL SECURITY USER AUTHORIZATIONS

Oracle9i Label Security user authorizations must be established by a security administrator before an application user can access an application table protected by Oracle9i Label Security. Oracle9i Label Security user label authorizations are defined as follows:

Maximum Level — The maximum sensitivity level a user is authorized to access. In a hosting environment only single level may exist. In government and defense environments four or five levels might be defined.

Minimum Level — The minimum sensitivity level a user is authorized to write data. For example, an administrator can prevent users from labeling data as *Public* or *Internet* by assigning a minimum level of *Company Confidential.*

Default Level — The level used by default when a user connects to the database. For example, a user can set his or her default level to Secret. When he or she connects to the system, the default level will be initialized to Secret.

Row Level — The level used to label data inserted into the database by the user through the application or directly through a tool such as SQL*Plus.

Read Compartments — The set of compartments assigned to the user and used during READ access mediation. For example, if a user has compartments *A,B and C*, he could view data which has compartments *A and B* but not data which has compartments *A,B,C and D.* The read algorithm will be defined in the next section.

Write Compartments — The set of compartments assigned to the user and used during WRITE access mediation. For example, a user could be given READ and WRITE access to compartments *A and B* but READ-ONLY access to compartment *C.* If an application record was labeled with compartments *A,B and C,* the user would not be allowed to update the record because he or she does not have WRITE access on compartment *C.*

Read Groups — The set of groups assigned to the user and used during READ access mediation. For example, if a user had the group *Manager*, he could view data which has the *Manager* group but not data which had the *Senior VP* group. The read algorithm will be defined in the next section.

Write Groups — The set of groups assigned to the user and used during WRITE access mediation. For example, a user could be given READ and WRITE access to group *Senior VP* but READ-ONLY access to group *Manager.* If an application record was labeled with a single group, *Manager*, the user would not be allowed to update the record because he or she does not have WRITE access on the *Manager* group.

APPENDIX D - APPLICATION ANALYSIS CASE STUDY

Introduction

Key to the success of any enterprise security architecture is identifying the components which need added security protection. Row level security is no exception. Typically, only a small percentage of tables in an application will require sophisticated row level security. The following outlines the steps in the implementation of a new security policy using Oracle9i Label Security.

Business Scenario and Analysis

ACME Corporation, a leader in energy exploration, has decided to centralize twenty corporate databases distributed throughout the world. The twenty databases will be replaced by two databases located in North America, one of which will serve as the hot backup. This consolidation will save ACME enormous sums of money and promises to increase margins dramatically. However, ACME currently relies on physical separation as a key element in its security policy. The existing twenty databases represent individual energy exploration territories. The twenty exploration databases are subdivided into four global regions, each containing five exploration territories. Finally, the global enterprise region encompasses the four global regions, producing a worldwide view.

The ACME IT security department is very concerned about the central consolidation and its ability to provide the same granular access control provided by the distributed model. The ACME IT department has decided to use Oracle for the two new centralized servers. In addition, the ACME IT department has just started the analysis process to determine the impact of consolidation on the current application. The ACME IT department has three goals:

1) Increased security
2) Application modification must be minimized
3) Access controls which exist in the current distributed model must be maintained in the new centralized design

Scheme Analysis

The first step is to analyze the application and determine which tables require added security protection. After analyzing the new policies, the ACME IT department determines that the FACILITY table is the principal table needing row level security. Other tables can be protected using traditional Oracle system and object privileges.

Data Level Analysis

The second step is to examine the data in the two tables and determine the sensitivity levels of the data. After completing this task, the security administrator identifies three sensitivity levels.

INTERNAL

SENSITIVE

HIGHLY SENSITIVE

Data Group Analysis

The third step is to analyze the data and determine what access control groups exist as well as any access control group hierarchies. The security administrator identifies the following groups.

Territories 1 through 20

Compartment Analysis

The fourth step is to analyze the data and determine if compartments are necessary to implement the policy. ACME has decided not to use compartments for this security implementation because the level and group functionality is sufficient.

User Community Analysis

The fifth step is to examine the ACME Corporation user community and define groups. The following user community groups are identified.

Territory Managers

Regional Managers

Corporation Managers

Special Access Requirements

The sixth step is to examine the application and determine whether any special access authorizations are required for business reports or special processing tasks. Oracle9i Label Security has an array of privileges which allow data to be accessed outside the authorized range of access. The privileges can be granted to users or stored program units. ACME corporate managers have asked the IT department to build a special mechanism in the centralized database which will allow corporate managers to easily verify the access abilities of subordinate employees. The Oracle9i Label Security PROFILE_ACCESS privilege will provide this capability.

Review and Document

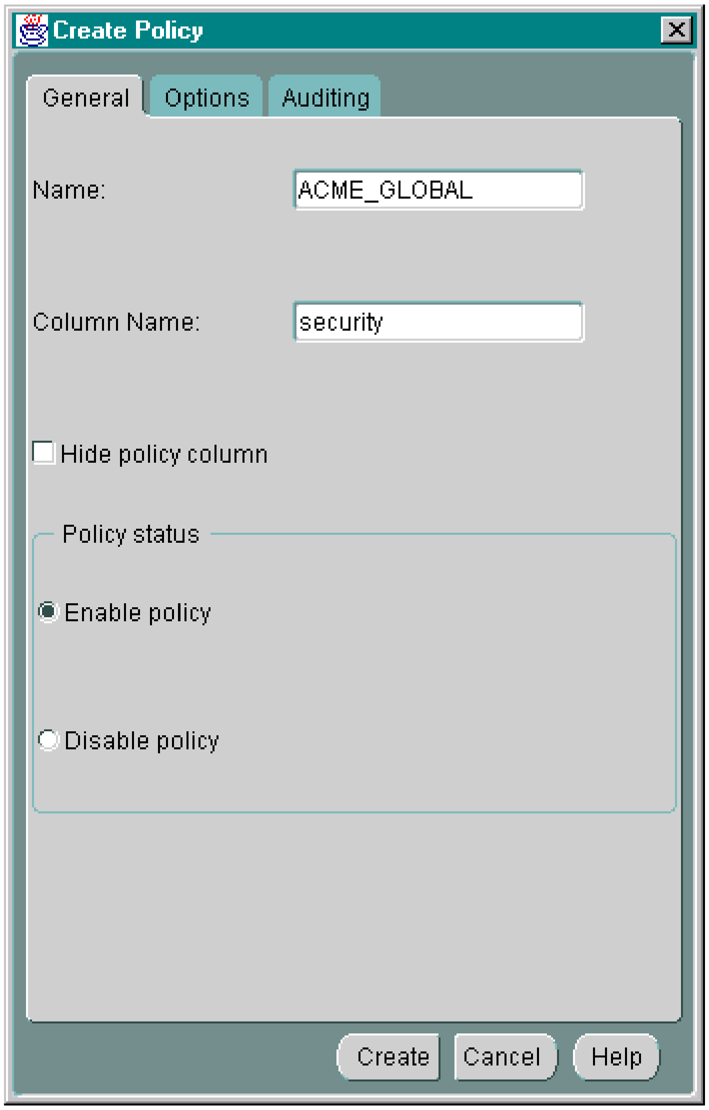
The seventh step is to review and verify the information gathered. This may involve repeating the above process and reviewing the information gathered with various departments. In our scenario, as in others, it is important to compare the information collected during data group analysis with the information collected during user community analysis. In addition, the information collected during data level analysis should be compared with the information gathered during user community analysis. Why is this important? If not done correctly, the user community authorized to access the data may be prevented from doing so. Three potential scenarios exist which can cause this problem. First, the user community has not been given the appropriate authorizations. Second, data has been assigned an incorrect sensitivity label or over classified. Third, application users or stored program units have not been given the appropriate Oracle9i Label Security privilege authorizations identified during analysis.

Finally, the analysis and data gathered during the analysis should be documented and incorporated into the enterprise security policy.

IMPLEMENTATION

Step 1 - Define the policy

The first step in implementing the new policy is to define a policy using Oracle9i Policy Manager. Creating the policy with Oracle9i Policy Manager is as simple as specifying a descriptive name for the policy as well as a column name for the policy to use. In this case, a policy called *ACME_GLOBAL* is created. Note that the column specified can be specified as hidden.



Name: ACME_GLOBAL

Column Name: security

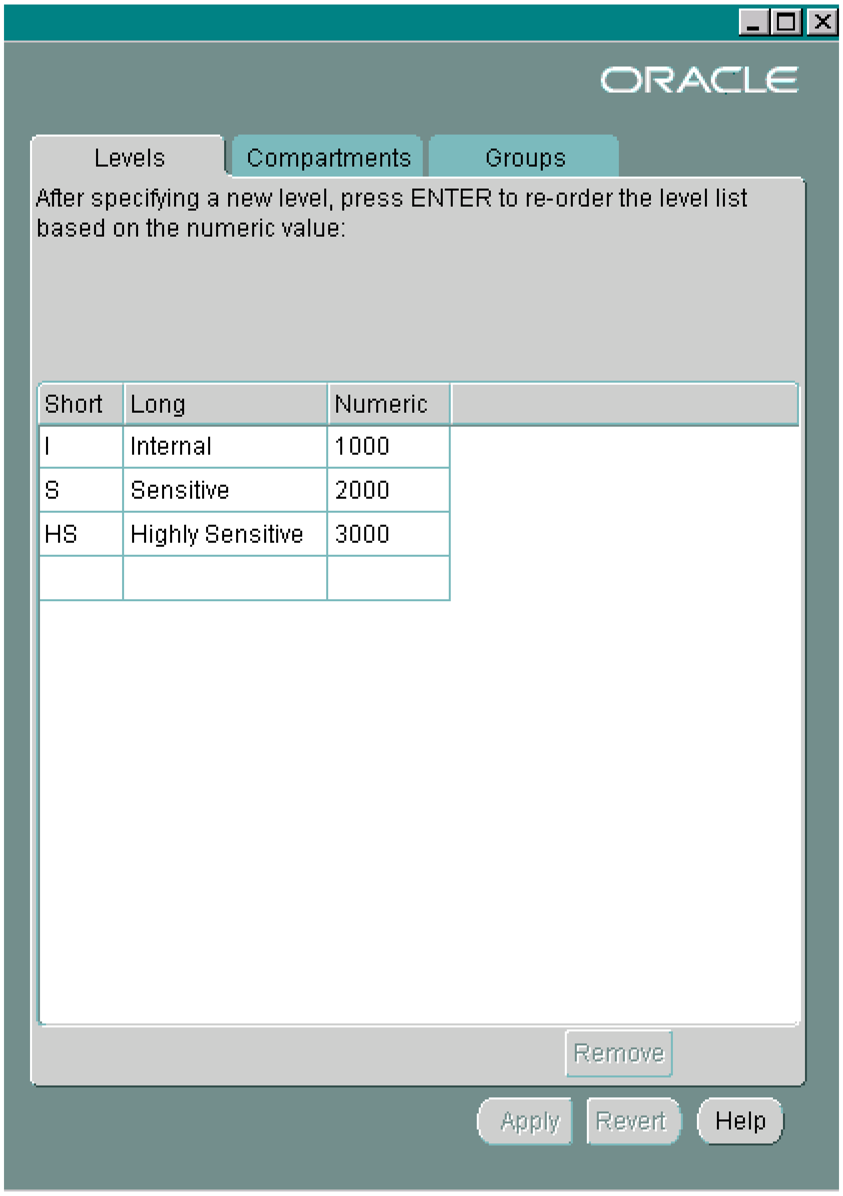
☐ Hide policy column

Policy status

● Enable policy

○ Disable policy

Step 2 - Define Level components

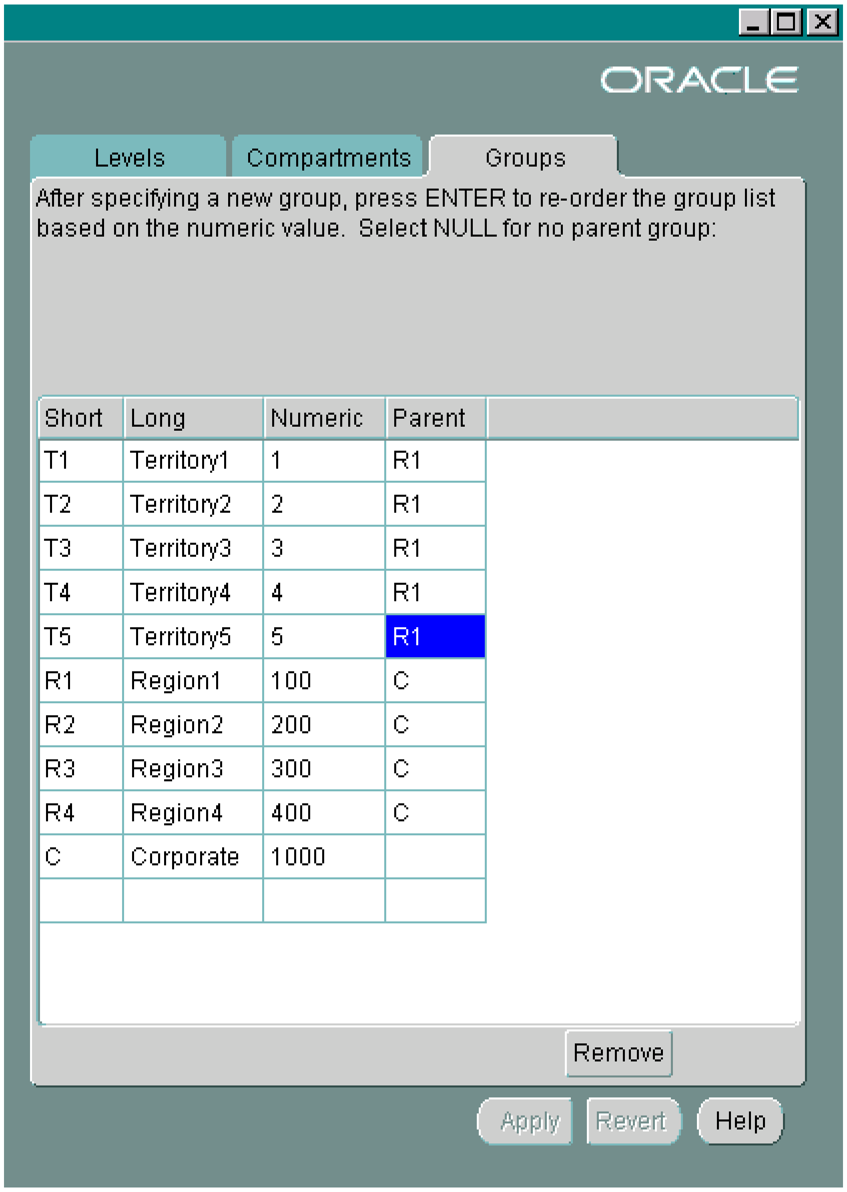


After specifying a new level, press ENTER to re-order the level list based on the numeric value:

Short	Long	Numeric
I	Internal	1000
S	Sensitive	2000
HS	Highly Sensitive	3000

Define the data levels identified during analysis.

Step 3 - Define Group Components



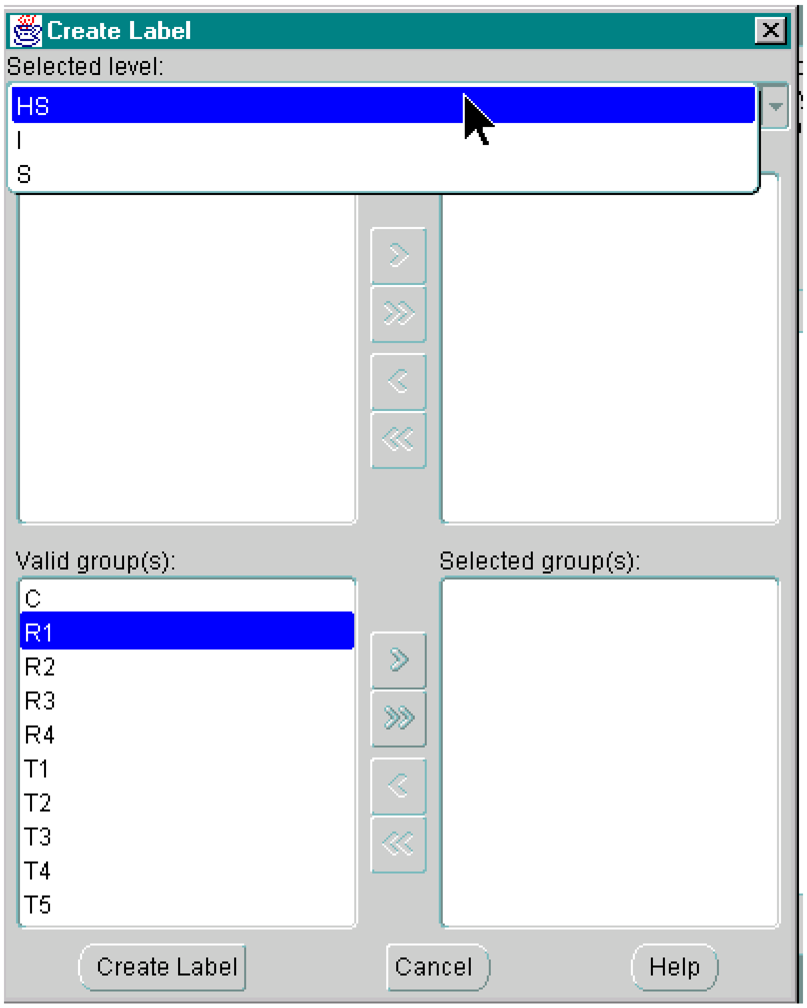
After specifying a new group, press ENTER to re-order the group list based on the numeric value. Select NULL for no parent group:

Short	Long	Numeric	Parent
T1	Territory1	1	R1
T2	Territory2	2	R1
T3	Territory3	3	R1
T4	Territory4	4	R1
T5	Territory5	5	R1
R1	Region1	100	C
R2	Region2	200	C
R3	Region3	300	C
R4	Region4	400	C
C	Corporate	1000	

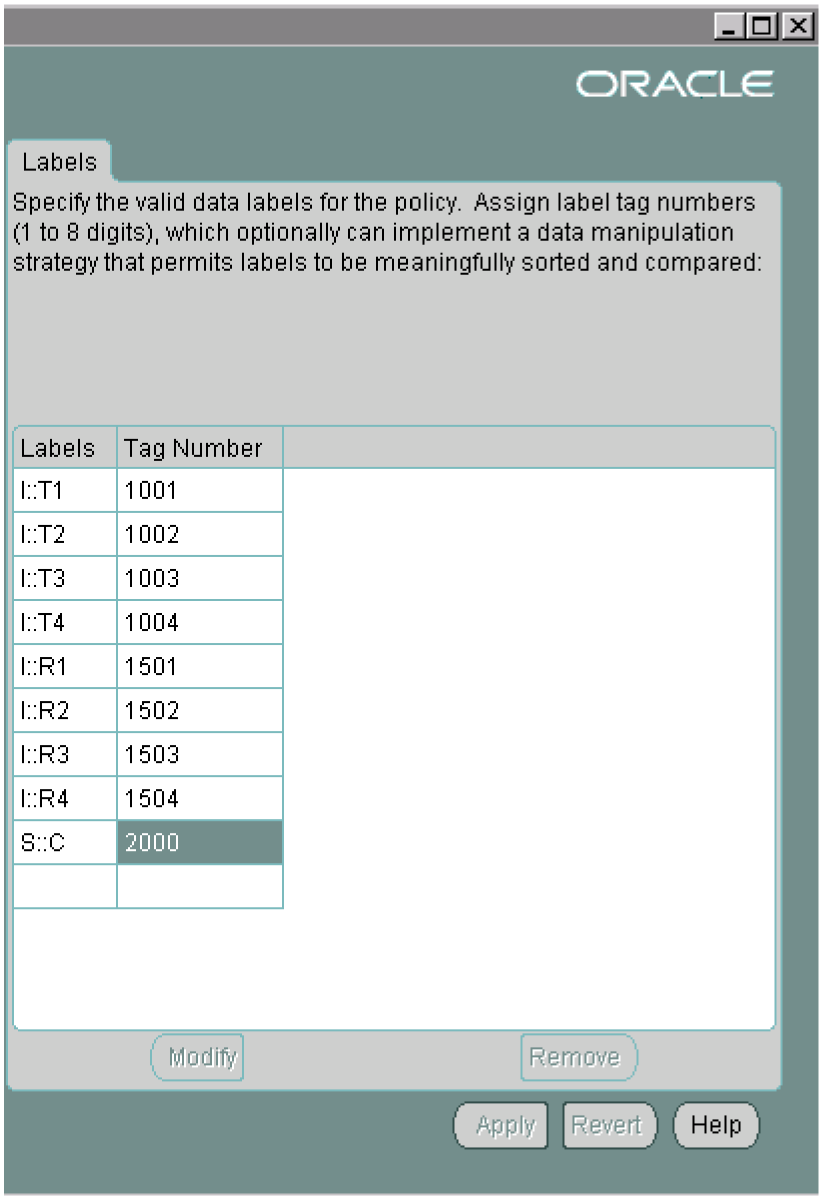
A good practice is to define all groups prior to setting the parent value

Define the groups identified during user community and data analysis. Notice the hierarchy which established by specifying parent values. A good practice is to define all groups prior to setting the parent value.

Step 4 - Define valid sensitivity labels



Sensitivity labels are created by selecting from the defined components and assigning a label tag value.

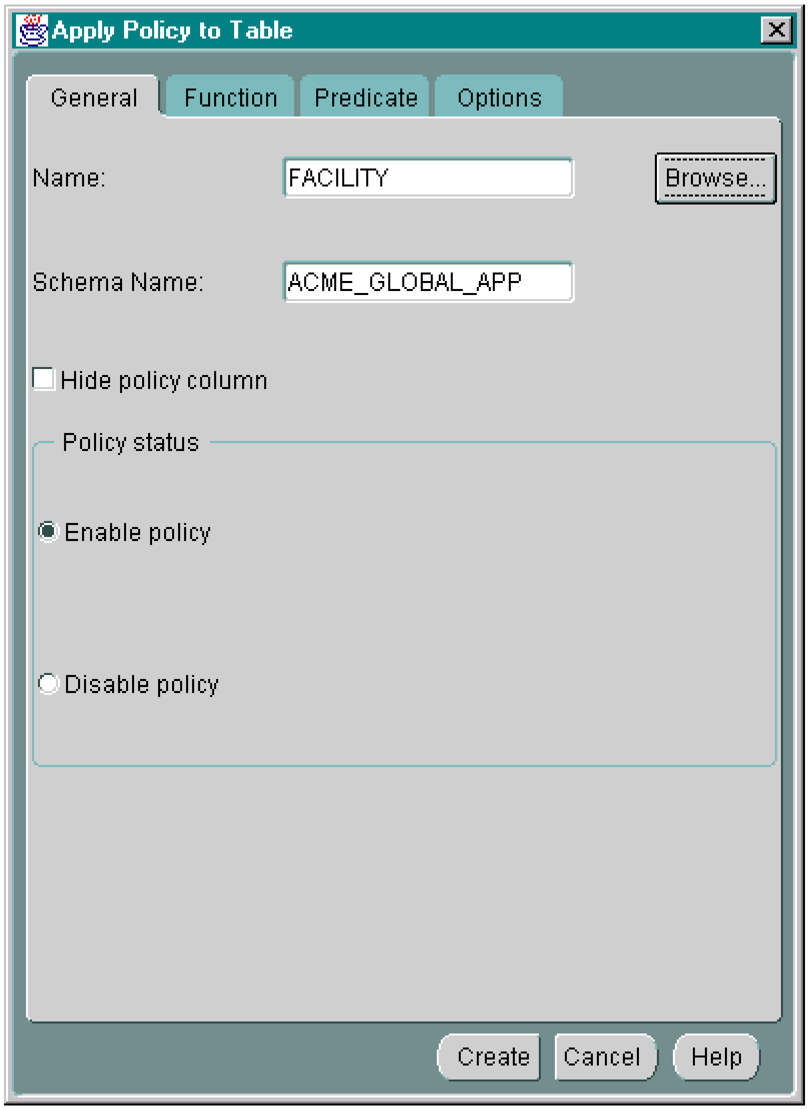

Labels

Specify the valid data labels for the policy. Assign label tag numbers (1 to 8 digits), which optionally can implement a data manipulation strategy that permits labels to be meaningfully sorted and compared:

Labels	Tag Number
I::T1	1001
I::T2	1002
I::T3	1003
I::T4	1004
I::R1	1501
I::R2	1502
I::R3	1503
I::R4	1504
S::C	2000

Notice that labels in the level Internal are in the 1000-1999 range and labels in the Sensitive range are in the 2000-2999 range.

Step 5 - Apply ACME_GLOBAL policy to tables



Name: FACILITY

Schema Name: ACME_GLOBAL_APP

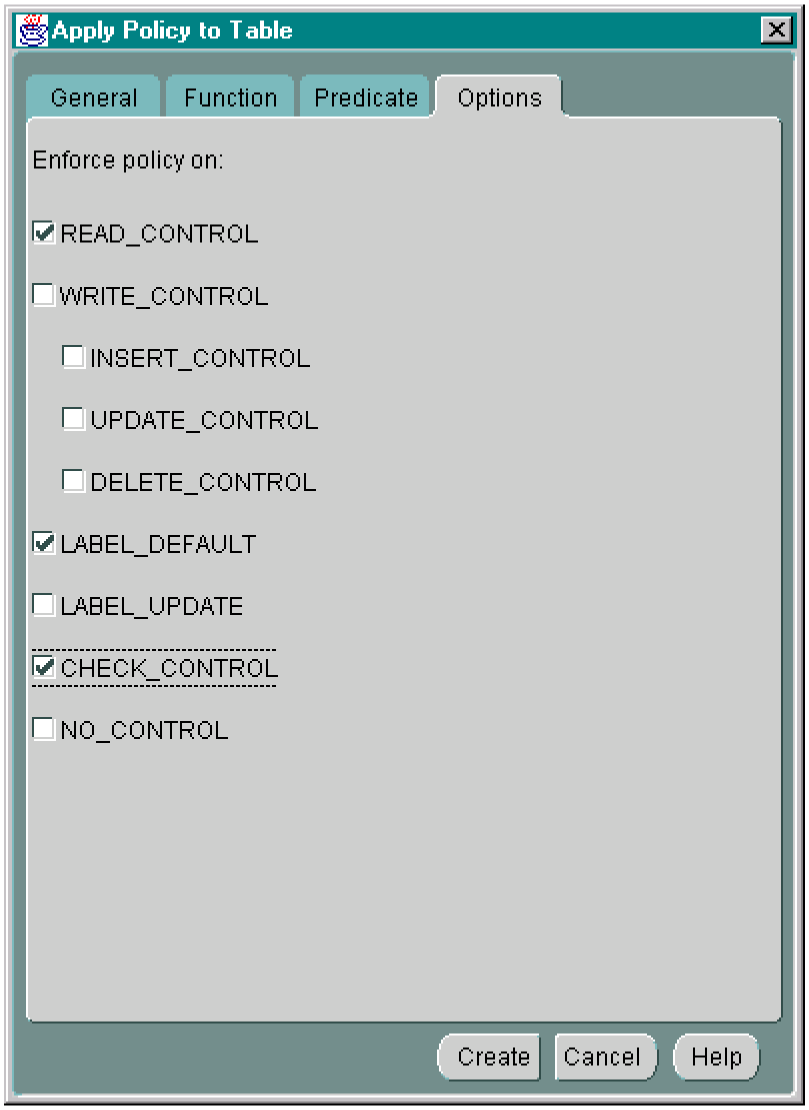
☐ Hide policy column

Policy status

[✓ Enable policy] Disable policy

At this point the policy can be applied to the tables identified during the analysis stage. In this scenario, the facility table is protected.

Step 6 - Set ACME_GLOBAL policy enforcement options



Apply Policy to Table

Enforce policy on:

[✓] READ_CONTROL

[ ] WRITE_CONTROL

[ ] INSERT_CONTROL

[ ] UPDATE_CONTROL

[ ] DELETE_CONTROL

[✓] LABEL_DEFAULT

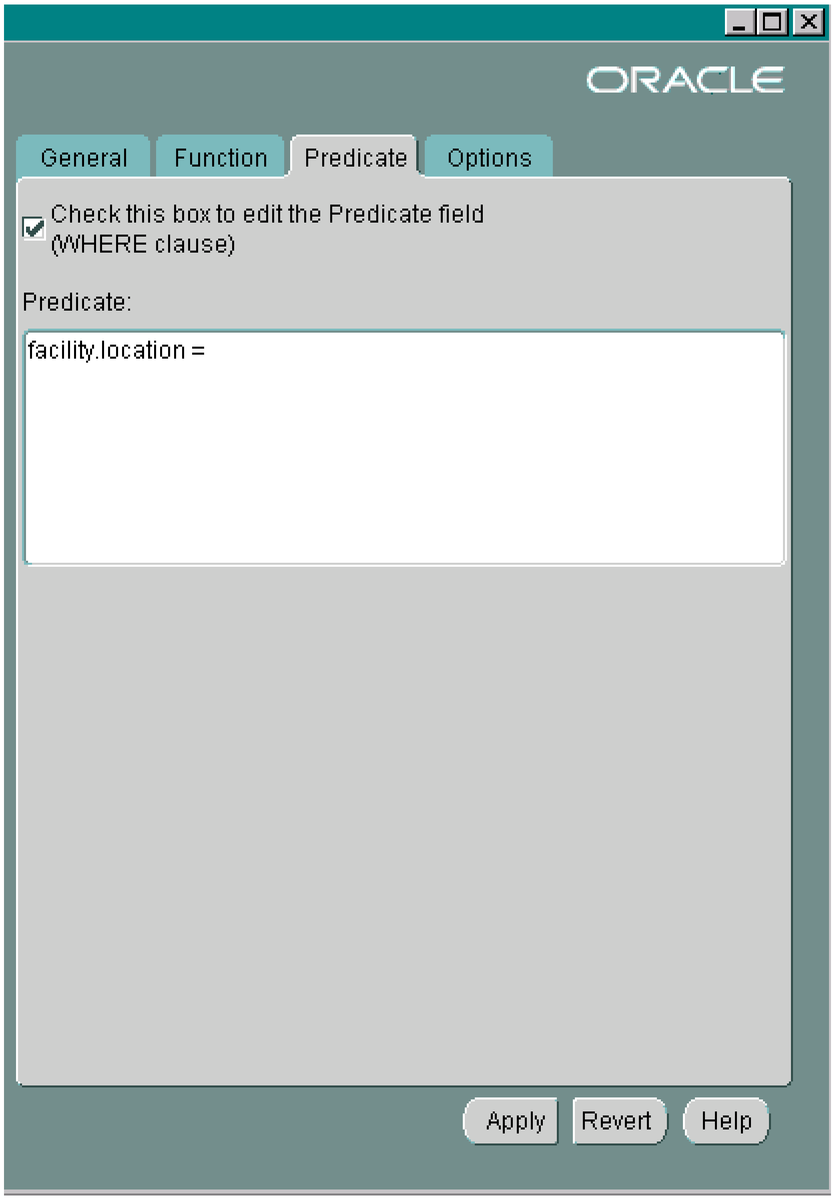
[ ] LABEL_UPDATE

[✓] CHECK_CONTROL

[ ] NO_CONTROL

When a policy is applied to a table, the degree of enforcement can be customized to the individual needs of the enterprise. Here the policy options selected are READ_CONTROL, LABEL_DEFAULT and CHECK_CONTROL. Please refer to the Oracle9i Label Security feature overview and administrator's guide for a complete description of the various enforcement options.

Step 7 - Set ACME_GLOBAL policy optional predicate



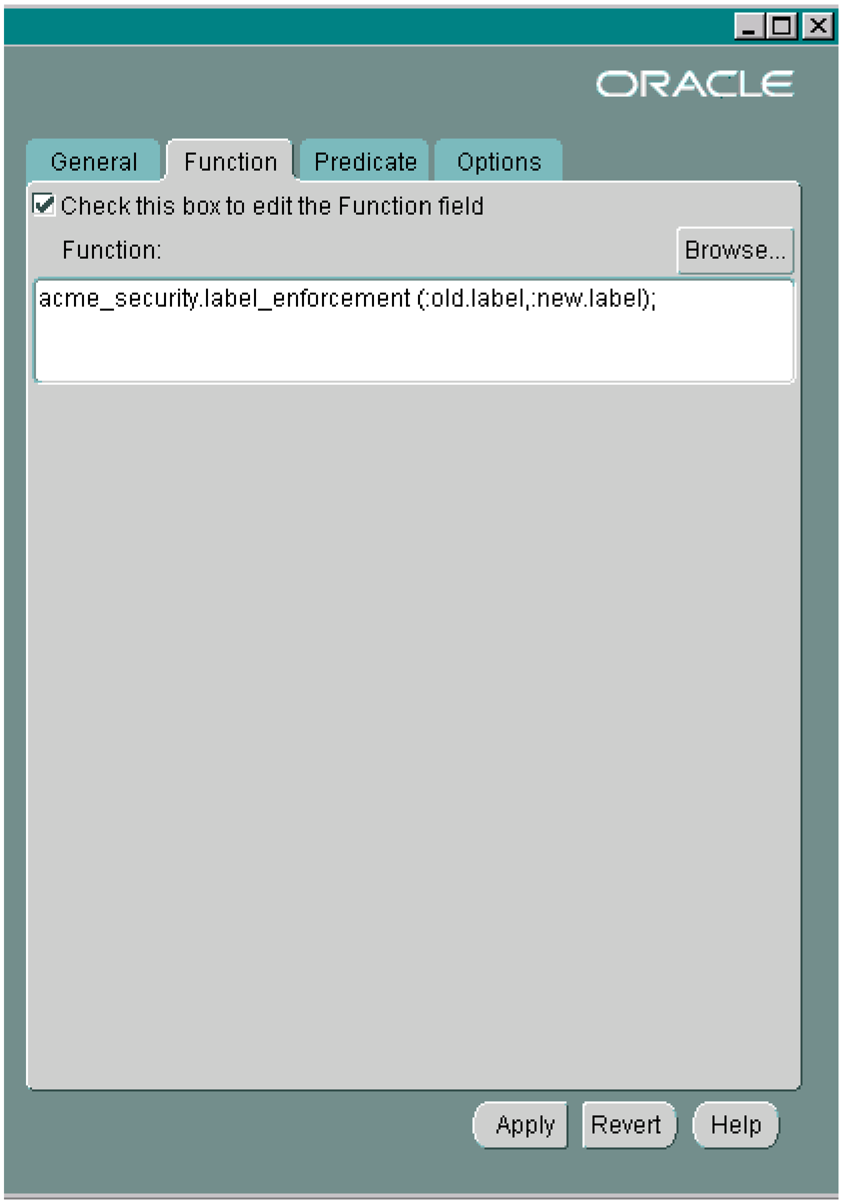
[✓] Check this box to edit the Predicate field (WHERE clause)

Predicate:

```
facility.location =
```

Oracle9i Label Security allows an additional *where conditional* to be added and evaluated in addition to the label based access control. For example, an Oracle SYS_CONTEXT variable could be referenced.

Step 8 - Set ACME_GLOBAL policy optional label function



[✓] Check this box to edit the Function field

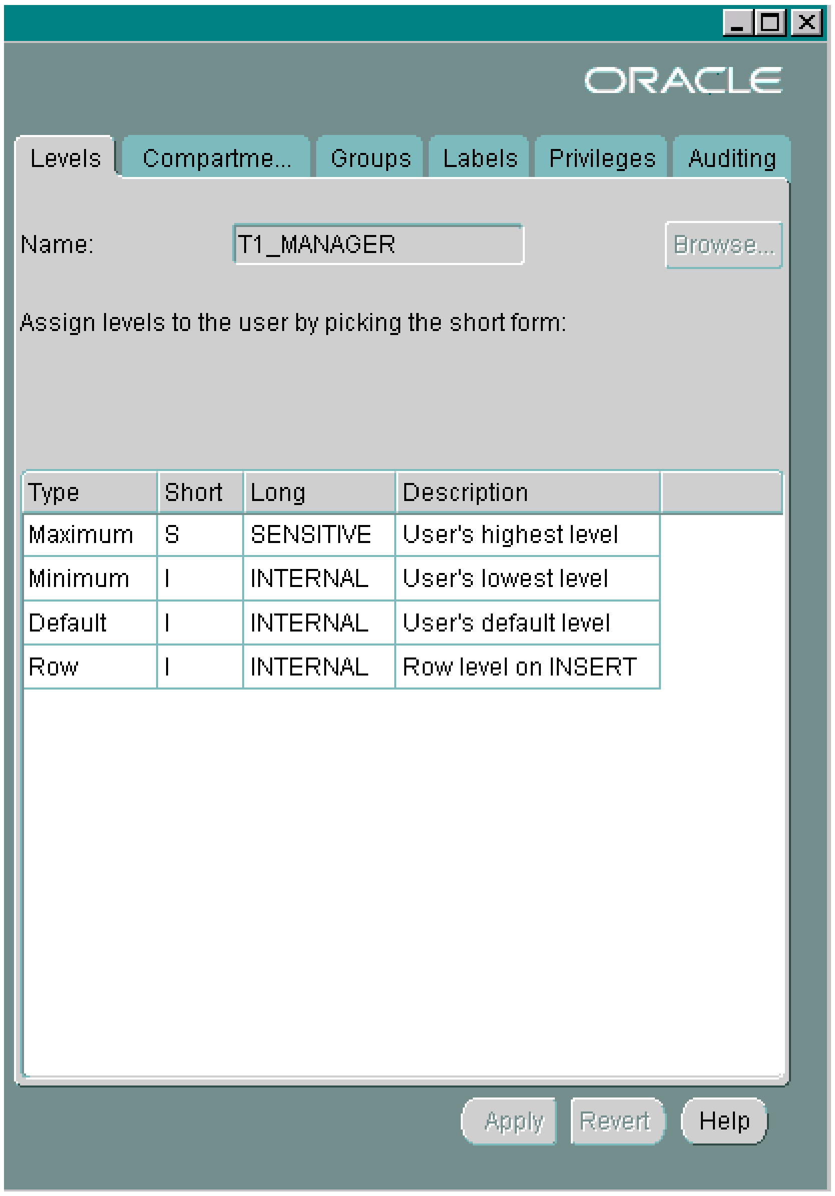
Function:

```
acme_security.label_enforcement (:old.label,:new.label);
```

Oracle9i Label Security allows an optional labeling function to be referenced within the policy. The labeling function will override any label otherwise specified and set the label equal to a value determined by the function. Labeling functions are called in the context of a special before row trigger.

Step 9 - Territory Manager Label Authorizations

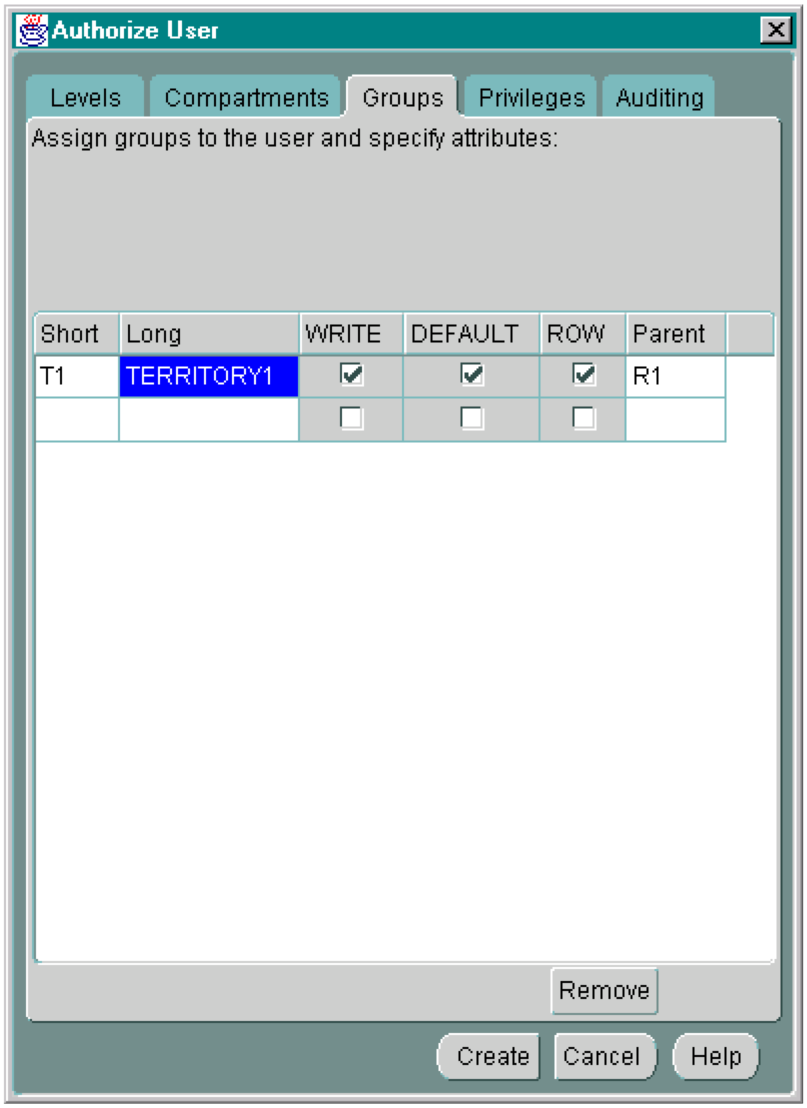
For each of the various levels of management, the appropriate label authorizations need to be set up. First the territory managers are assigned appropriate levels and groups. Second the regional managers are assigned appropriate levels and groups. Finally, the corporate managers are assigned appropriate levels and groups.



Name: T1_MANAGER

Assign levels to the user by picking the short form:

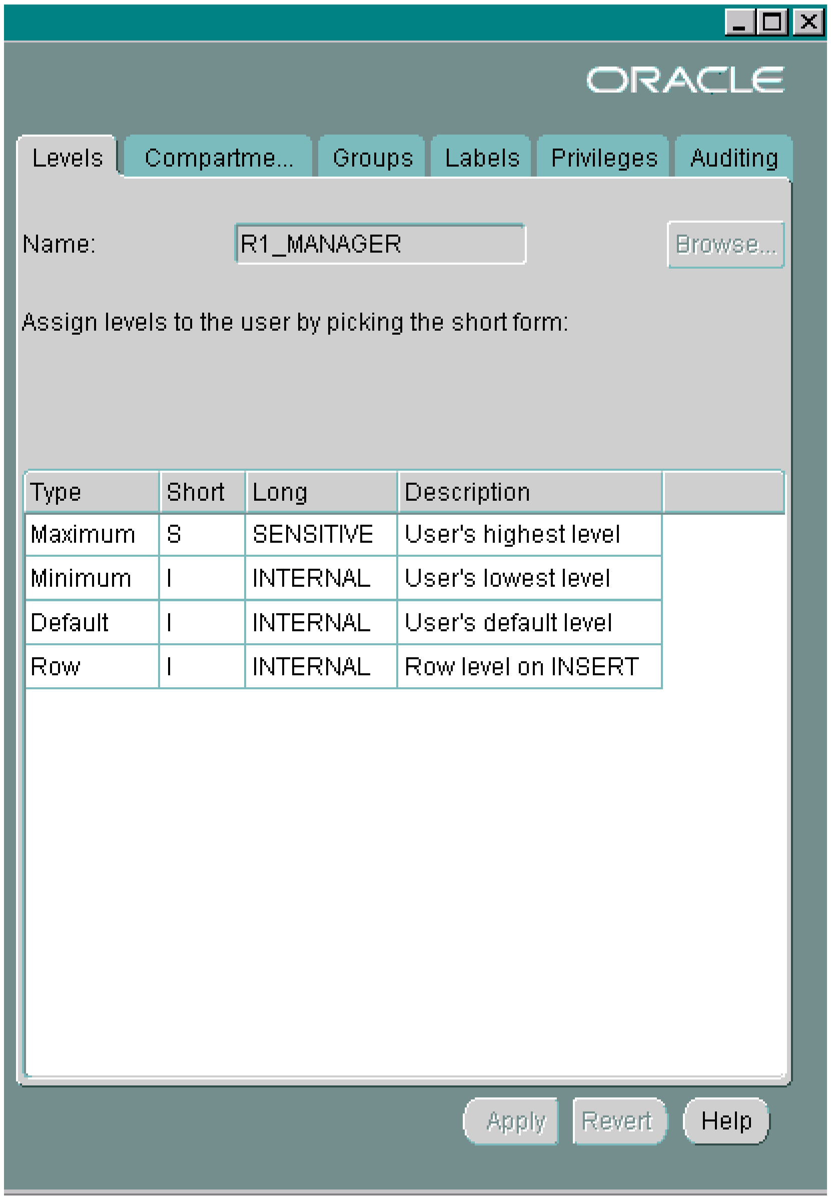
Type	Short	Long	Description
Maximum	S	SENSITIVE	User's highest level
Minimum	I	INTERNAL	User's lowest level
Default	I	INTERNAL	User's default level
Row	I	INTERNAL	Row level on INSERT



Assign groups to the user and specify attributes:

Short	Long	WRITE	DEFAULT	ROW	Parent
T1	TERRITORY1	✓	✓	✓	R1
		☐	☐	☐	

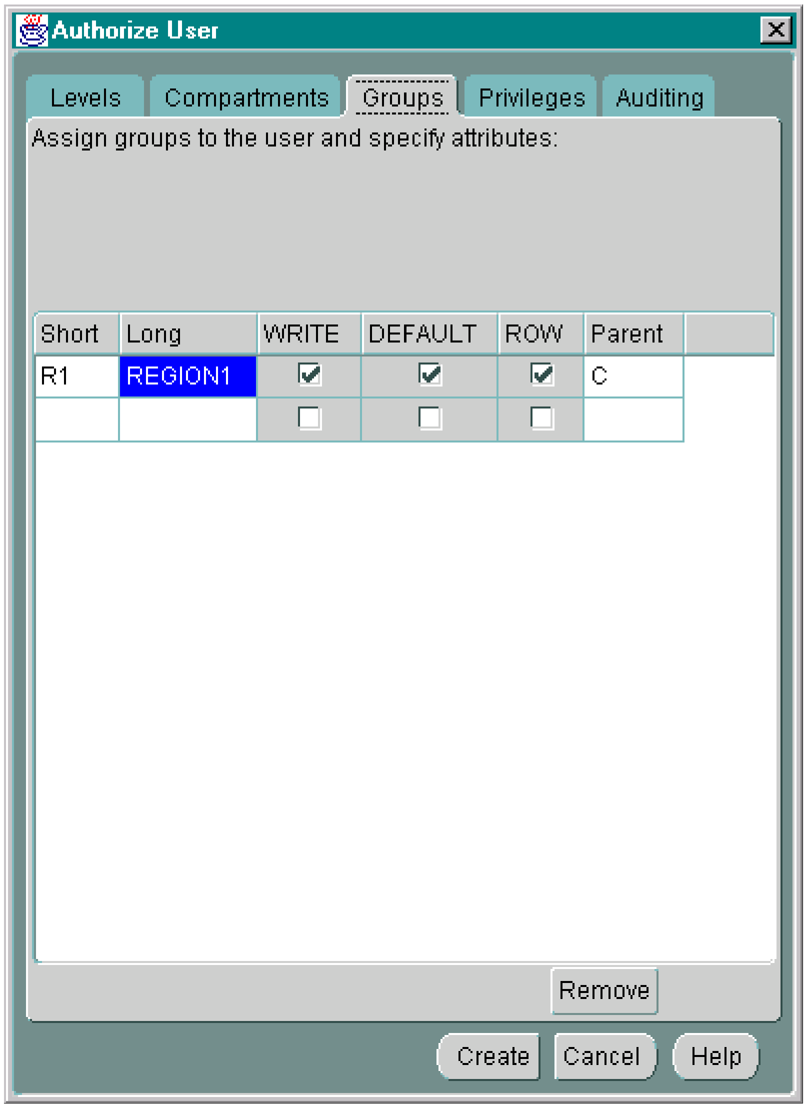
Step 10 - Regional Manager Label Authorizations



Name: R1_MANAGER

Assign levels to the user by picking the short form:

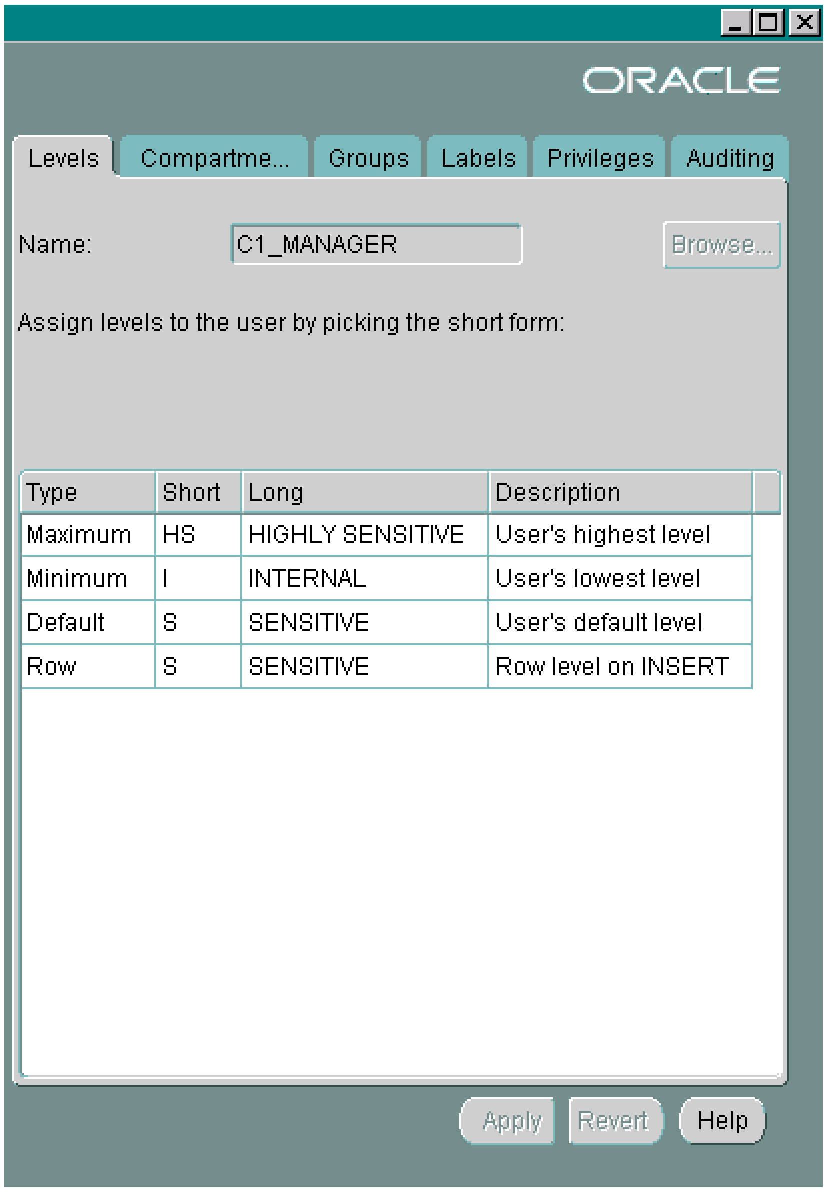
Type	Short	Long	Description
Maximum	S	SENSITIVE	User's highest level
Minimum	I	INTERNAL	User's lowest level
Default	I	INTERNAL	User's default level
Row	I	INTERNAL	Row level on INSERT



Assign groups to the user and specify attributes:

Short	Long	WRITE	DEFAULT	ROW	Parent
R1	REGION1	✓	✓	✓	C
		☐	☐	☐	

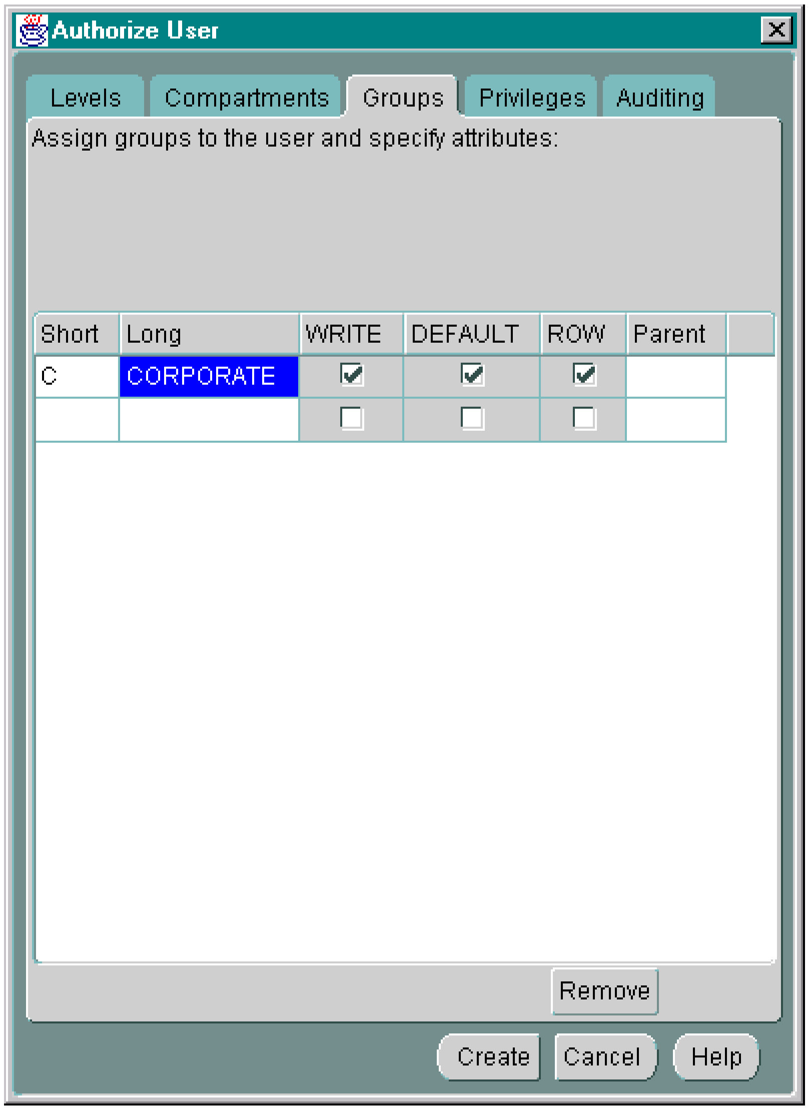
Step 11 - Corporate Manager Label Authorizations



Name: C1_MANAGER

Assign levels to the user by picking the short form:

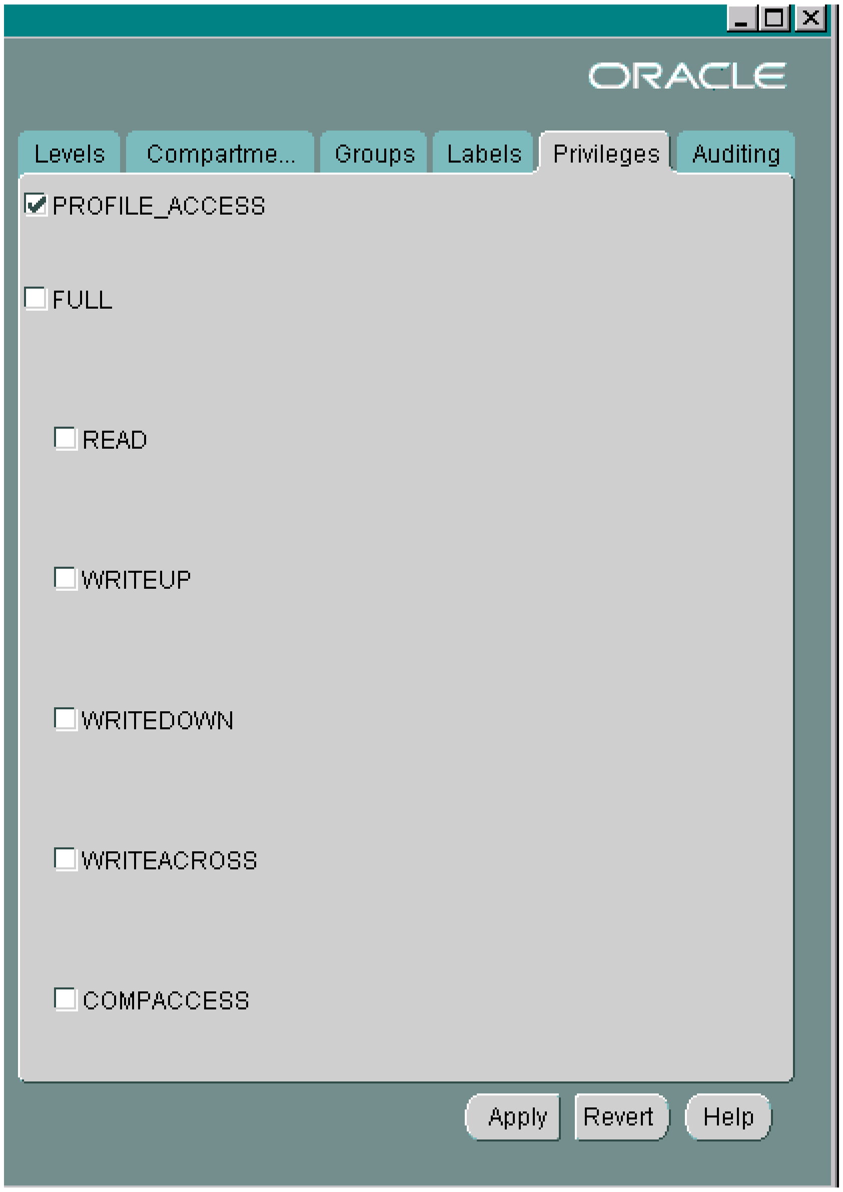
Type	Short	Long	Description
Maximum	HS	HIGHLY SENSITIVE	User's highest level
Minimum	I	INTERNAL	User's lowest level
Default	S	SENSITIVE	User's default level
Row	S	SENSITIVE	Row level on INSERT



Assign groups to the user and specify attributes:

Short	Long	WRITE	DEFAULT	ROW	Parent
C	CORPORATE	✓	✓	✓	
		☐	☐	☐	

Step 12 - Authorize Corporate Managers for special privileges



[✓] PROFILE_ACCESS

[ ] FULL

[ ] READ

[ ] WRITEUP

[ ] WRITEDOWN

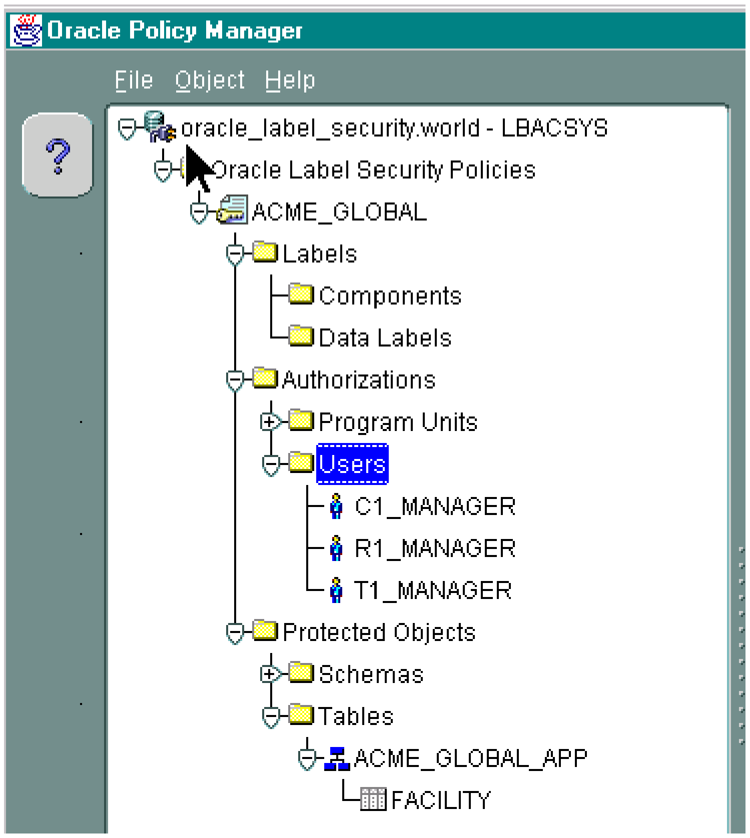
[ ] WRITEACROSS

[ ] COMPACCESS

During analysis it was determined that corporate managers should be able to assume the profile of any subordinate. To accommodate this requirements, the PROFILE_ACCESS privilege is authorized for all corporate level managers.

Oracle9i Policy Manager ACME_GLOBAL Policy Overview

The expanded view of the ACME_GLOBAL policy shows the protected objects and authorized users. In addition the component and data label tabs are shown. At this point, authorizations have been completed for ACME territory one managers, regional managers and corporate managers. Users in territory one will have the level and group label component authorizations specified in the previous pictures when connecting to the new ACME centralized application.



```
oracle_label_security.world - LBACSYS
  Oracle Label Security Policies
    ACME_GLOBAL
      Labels
        Components
        Data Labels
      Authorizations
        Program Units
        Users
          C1_MANAGER
          R1_MANAGER
          T1_MANAGER
      Protected Objects
        Schemas
        Tables
          ACME_GLOBAL_APP
            FACILITY
```

Oracle9i Label Security API

In addition to Oracle9i Policy Manager,. Oracle9i Label Security also provides a comprehensive API. The API can be used to establish authorizations for a large number of users in batch mode. The API is fully documented in the Oracle9i Label Security administrator’s guide.

APPENDIX E - HOSTING AND PRIVACY DEMONSTRATION USING THE API

Overview

This demonstration shows how Oracle9i Label Security can be used to implement simple company by company data separation as well as how Oracle9i Label Security can be used to achieve a more sophisticated granular data separation within a specific company. This type of technology is important because database consolidation and the Internet have made granular access control extremely important due to the increase in number of users who can access applications.

In this demonstration, a policy called HOSTING is created using Oracle9i Label Security. A single level called 'company' is created and a group is created corresponding to each company. For the purposes of this demonstration, two companies exist: ACME Corporation and Widget Corporation. For the basic HOSTING policy multiple levels were not necessary and therefore only a single level called 'company' was created.

In addition to the HOSTING policy, a policy called PRIVACY is created to demonstrate how a greater degree of granular access control can be achieved by using Oracle9i Label Security levels, compartments and groups. In this demonstration, only levels and groups are used. Levels and groups are usually sufficient for commercial organizations. Levels, compartments and groups are called label components. Label components must be defined before actual labels can be created and assigned to data. For example, the label components 'secret' and 'NATO' need to be created before the label 'SECRET : : NATO' can be created and assigned to data. Two Oracle9i Label Security policies have been created for this demo. They are

- HOSTING
- PRIVACY

Key Demonstration Points

1. Multiple policies are assigned to application table SALES_DATA
2. Policy column for the privacy policy is hidden. Issuing a describe command in SQL*Plus does not show the privacy policy column SECLAB. The benefits of using the hidden column include the ability to continue using SQL statements which do not specify a column list during insert.
3. The column HOSTLAB pre-exists in the table and is reused by Oracle9i Label Security. In order for Oracle9i Label Security to use an existing column, the column must be defined as number (10).
4. Groups are used to provide added granularity for privacy and other business needs. In this case, the WIDGET corporation has groups defined for countries in EUROPE and the ACME Corporation has groups defined for various parts of the USA. The groups roll up to a parent group. For ACME the parent group is US and for WIDGET the parent group is EUROPE.
5. This demonstration also shows how the Oracle9i Label Security SQL*predicate capability can be used to enforce additional access controls. When an Oracle9i Label Security policy is assigned to a table, a SQL*predicate can be supplied which will be appended to SQL statements accessing the table. For this demo, a SQL*predicate is added which restricts access to data outside Monday through Friday.
6. VPD could be used to implement the hosting policy and achieve simple company by company data separation. Oracle9i Label Security eliminates the coding and provides the foundation for more sophisticated granular access controls.
7. Industries such as Healthcare, Retail (franchises), Energy, Government, Law Enforcement, National Security, and Finance can use Oracle9i Label Security to solve sophisticated security requirements and enhance their individual product offerings.

Demonstration Accounts

The application owner account is SALESAPP and the password is SALESAPP. A single table called *sales_data* is used in this demonstration..

Two accounts exist which show how the *hosting* policy can be combined with a second Oracle9i Label Security policy called *privacy* to achieve additional access control granularity. They are:

ACME_MGR

WIDGET_MGR

Logging in as either of these users will retrieve a subset of information within their company.   For example, user WIDGET_MGR will only see information for the country GERMANY while the user ACME_MGR will only see information for the USWEST territory of ACME corporation.

In addition, two more accounts were created to show how both policies can be in effect and the Oracle9i Label Security group functionality used to allow the CEO's of both companies to see information across all company territories or sales regions.   They are:

ACME_CEO

WIDGET_CEO

Demo Build Using Oracle9i Label Security API

The files associated with this demo will be available for download on the Oracle Technology Network.

```
Prompt
Prompt ************************************************************************
Prompt Connect as User SYSTEM or another DBA with permissions to
Prompt database accounts
Prompt ************************************************************************

CONNECT system@demo.world

Prompt ************************************************************************
Prompt Create Users SALESAPP
Prompt Create Users ACME_MGR
Prompt Create Users WIDGET_MGR
Prompt Create Users ACME_CEO
Prompt Create Users WIDGET_CEO
Prompt ************************************************************************

GRANT CONNECT, RESOURCE to SALESAPP IDENTIFIED BY SALESAPP;
GRANT CONNECT, RESOURCE to ACME_MGR IDENTIFIED BY ACME_MGR;
GRANT CONNECT, RESOURCE to WIDGET_MGR IDENTIFIED BY
WIDGET_MGR;
GRANT CONNECT, RESOURCE to ACME_CEO IDENTIFIED BY ACME_CEO;
GRANT CONNECT, RESOURCE to WIDGET_CEO IDENTIFIED BY WIDGET_CEO;

Prompt ************************************************************************
Prompt Grant User SALESAPP CREATE PUBLIC SYNONYM
Prompt ************************************************************************

GRANT CREATE PUBLIC SYNONYM to salesapp;
```

```
Prompt ********************************************************************
Prompt Connect as User LBACSYS
Prompt ********************************************************************

CONNECT lbacsys@demo.world

SPOOL ols_demo_setup.log

Prompt ********************************************************************
Prompt Grant User SALESAPP EXECUTE ON
Prompt
Prompt SA_SESSION
Prompt SA_SYSDBA
Prompt SA_USER_ADMIN
Prompt SA_COMPONENTS
Prompt SA_UTL
Prompt SA_LABEL_ADMIN
Prompt SA_POLICY_ADMIN
Prompt
Prompt ********************************************************************

GRANT EXECUTE ON sa_session    TO salesapp;
GRANT EXECUTE ON sa_sysdba     TO salesapp;
GRANT EXECUTE ON sa_user_admin TO salesapp;
GRANT EXECUTE ON sa_components TO salesapp;
GRANT EXECUTE ON sa_utl        TO salesapp;
GRANT EXECUTE ON sa_label_admin TO salesapp;
GRANT EXECUTE ON sa_policy_admin to salesapp;
```

```
Prompt **********************************************************************
Prompt Grant role LBAC_DBA to User SALESAPP
Prompt **********************************************************************

GRANT LBAC_DBA TO salesapp;

Prompt **********************************************************************
Prompt Connecting as user salesapp
Prompt **********************************************************************

CONNECT salesapp/salesapp@demo.world

Prompt **********************************************************************
Prompt Dropping HOSTING policy
Prompt **********************************************************************

EXECUTE SA_SYSDBA.DROP_POLICY('HOSTING');

Prompt **********************************************************************
Prompt Dropping PRIVACY policy
Prompt **********************************************************************

EXECUTE SA_SYSDBA.DROP_POLICY('PRIVACY');
```

```
Prompt **********************************************************************
Prompt Drop and create table sales_data
Prompt **********************************************************************

DROP TABLE sales_data;

CREATE TABLE salesapp.sales_data  (
                        ORG_ID          number(4),
                        ORG_NAME        varchar2(15),
                        DIV             varchar2(14),
                        MONTH           varchar2(3),
                        YEAR            number(4),
                        HOURS           number(4),
                        EXPENSES        number(6),
                        HOSTLAB         number(10));

Describe salesapp.sales_data

Prompt
Prompt **********************************************************************
Prompt Add primary key to table sales_data
Prompt **********************************************************************

Alter table salesapp.sales_data add constraint SALES_PK PRIMARY KEY
(ORG_ID,MONTH,YEAR,DIV,ORG_NAME,HOSTLAB);
```

```
Prompt **********************************************************************

Prompt Grant select,insert,update,delete on salesapp.sales_data to PUBLIC

Prompt **********************************************************************


GRANT SELECT, INSERT, UPDATE, DELETE ON SALES_DATA to PUBLIC;


Prompt

Prompt **********************************************************************

Prompt Create public synonym sales_data for table salesapp.sales_data

Prompt **********************************************************************


CREATE PUBLIC SYNONYM SALES_DATA FOR SALESAPP.SALES_DATA;


Prompt **********************************************************************

Prompt Creating HOSTING Policy

Prompt **********************************************************************


EXECUTE
SA_SYSDBA.CREATE_POLICY('HOSTING','HOSTLAB','READ_CONTROL,CHECK
_CONTROL,LABEL_DEFAULT');


Prompt **********************************************************************

Prompt Creating PRIVACY Policy

Prompt **********************************************************************


EXECUTE
SA_SYSDBA.CREATE_POLICY('PRIVACY','SECLAB','READ_CONTROL,CHECK_C
ONTROL,LABEL_DEFAULT,HIDE');
```

```
Prompt **********************************************************************

Prompt Creating PRIVACY policy LEVEL label components

Prompt **********************************************************************



EXECUTE
SA_COMPONENTS.CREATE_LEVEL('PRIVACY',1000,'CONFIDENTIAL','CONFIDE
NTIAL');

EXECUTE
SA_COMPONENTS.CREATE_LEVEL('PRIVACY',2000,'SENSITIVE','SENSITIVE');

EXECUTE
SA_COMPONENTS.CREATE_LEVEL('PRIVACY',3000,'HIGHLY_SENSITIVE','HIGH
LY_SENSITIVE');


Prompt

Prompt **********************************************************************

Prompt Creating PRIVACY policy GROUP label components

Prompt **********************************************************************


EXECUTE SA_COMPONENTS.CREATE_GROUP('PRIVACY',1000,'Global','Global');

EXECUTE SA_COMPONENTS.CREATE_GROUP('PRIVACY',100,'US','United
States','GLOBAL');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',101,'USWest','USWest','US');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',102,'USCentral','USCentral','US');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',103,'USEast','USEast','US');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',200,'Europe','Europe');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',201,'Sweden','Sweden','Europe');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',202,'Germany','Germany','Europe
');
```

```
EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',203,'France','France','Europe');
EXECUTE
SA_COMPONENTS.CREATE_GROUP('PRIVACY',204,'Italy','Italy','Europe');


Prompt
Prompt **********************************************************************
Prompt Creating PRIVACY policy LABELS
Prompt **********************************************************************


EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1101,'CONFIDENTIAL::USWest');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1102,'CONFIDENTIAL::USCentral')
;
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1103,'CONFIDENTIAL::USEast');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3101,'HIGHLY_SENSITIVE::USWe
st');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3102,'HIGHLY_SENSITIVE::USCe
ntral');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3103,'HIGHLY_SENSITIVE::USEa
st');


EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1201,'CONFIDENTIAL::Sweden');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1202,'CONFIDENTIAL::Germany');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1203,'CONFIDENTIAL::France');
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',1204,'CONFIDENTIAL::Italy');
```

```
EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3201,'HIGHLY_SENSITIVE::Swed
en');

EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3202,'HIGHLY_SENSITIVE::Germ
any');

EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3203,'HIGHLY_SENSITIVE::Franc
e');

EXECUTE
SA_LABEL_ADMIN.CREATE_LABEL('PRIVACY',3204,'HIGHLY_SENSITIVE::Italy');


Prompt

Prompt ***********************************************************************

Prompt Creating HOSTING policy LEVEL label components

Prompt ***********************************************************************


EXECUTE
SA_COMPONENTS.CREATE_LEVEL('HOSTING',1000,'COMPANY','COMPANY');


Prompt

Prompt ***********************************************************************

Prompt Creating HOSTING policy GROUP label components

Prompt ***********************************************************************


EXECUTE SA_COMPONENTS.CREATE_GROUP('HOSTING',100,'ACME','ACME
Corporation');

EXECUTE
SA_COMPONENTS.CREATE_GROUP('HOSTING',101,'WIDGET','WIDGET
Corporation');
```

```
Prompt ***********************************************************************

Prompt Creating HOSTING policy LABELS

Prompt ***********************************************************************


EXECUTE SA_LABEL_ADMIN.CREATE_LABEL
('HOSTING',1100,'COMPANY::ACME');

EXECUTE SA_LABEL_ADMIN.CREATE_LABEL
('HOSTING',1200,'COMPANY::WIDGET');


Prompt

Prompt ***********************************************************************

Prompt Setting User Authorizations for users:

Prompt SALESAPP

Prompt ACME_MGR

Prompt WIDGET_MGR

Prompt ACME_CEO

Prompt WIDGET_CEO

Prompt ***********************************************************************


Prompt ***********************************************************************

Prompt Authorize user SALESAPP privileges FULL and PROFILE_ACCESS on

Prompt HOSTING and PRIVACY policies.  Authorize user SALESAPP highest label

Prompt on HOSTING and PRIVACY policies.

Prompt ***********************************************************************


EXECUTE SA_USER_ADMIN.SET_USER_PRIVS
('PRIVACY','SALESAPP','FULL,PROFILE_ACCESS');

EXECUTE SA_USER_ADMIN.SET_USER_PRIVS
('HOSTING','SALESAPP','FULL,PROFILE_ACCESS');
```

```
Prompt ********************************************************************

Prompt Setting ACME_MGR user label authorizations

Prompt Setting WIDGET_MGR user label authorizations

Prompt ********************************************************************


EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’HOSTING’,’ACME_MGR’,’COMPANY::ACME’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’HOSTING’,’WIDGET_MGR’,’COMPANY::WIDGET’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’PRIVACY’,’ACME_MGR’,’HIGHLY_SENSITIVE::USWEST’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’PRIVACY’,’WIDGET_MGR’,’HIGHLY_SENSITIVE::GERMANY’);


Prompt ******************************************************************************

Prompt Setting ACME_CEO user label authorizations

Prompt Setting WIDGET_CEO user label authorizations

Prompt *******************************************************************************


EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’HOSTING’,’ACME_CEO’,’COMPANY::ACME’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’HOSTING’,’WIDGET_CEO’,’COMPANY::WIDGET’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’PRIVACY’,’ACME_CEO’,’HIGHLY_SENSITIVE::GLOBAL’);

EXECUTE SA_USER_ADMIN.SET_USER_LABELS
(’PRIVACY’,’WIDGET_CEO’,’HIGHLY_SENSITIVE::EUROPE’);
```

Because the predicate input parameter is a string and contains embedded quotes, the input value must be built using string concatenation. Concatenating four single quotes results in a single quote. The resulting string is to_char(sysdate,'d') in (2,3,4,5,6)

```
Prompt **********************************************************************
Prompt Applying HOSTING policy to salesapp.sales_data table.
Begin
   sa_policy_admin.apply_table_policy (
     POLICY_NAME => 'HOSTING',
     SCHEMA_NAME => 'salesapp',
     TABLE_NAME  => 'sales_data',
     TABLE_OPTIONS => NULL,
     LABEL_FUNCTION => NULL,
     PREDICATE =>  'to_char(sysdate,' || '''' || 'd' || '''' ||
                   ') in (2,3,4,5,6)' );
END;
/
Prompt **********************************************************************
Prompt Applying PRIVACY policy to salesapp.sales_data table.
Prompt **********************************************************************
BEGIN
   sa_policy_admin.apply_table_policy (
     POLICY_NAME => 'PRIVACY',
     SCHEMA_NAME => 'salesapp',
     TABLE_NAME  => 'sales_data',
     TABLE_OPTIONS => NULL,
     LABEL_FUNCTION => NULL,
     PREDICATE => NULL);
End;
/
Spool off
```

Demo Data Load

```
SPOOL ols_demo_dataload.log

Prompt **********************************************************************

Prompt Populating Data

Prompt **********************************************************************

connect salesapp/salesapp@demo.world


Prompt **********************************************************************

Prompt Loading Acme ORG 123 NORTH-WEST data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’NORTH-WEST’,’United States’,’FEB’,2000,1469,37124,1100,1101);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES(123,’NORTH-WEST’,’United States’,’MAR’,2000,897,19459,1100,1101);


Prompt **********************************************************************

Prompt Loading Acme ORG 123 NORTH-WEST data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’NORTH-WEST’,’United States’,’FEB’,2001 ,664,14882,1100,3101);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’NORTH-WEST’,’United States’,’MAR’,2001 ,1629,42668,1100,3101);
```

```
Prompt **********************************************************************

Prompt Loading Acme ORG 115 MID-WEST data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’MID-WEST’,’United States’,’FEB’,2000,739,16521,1100,1102);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’MID-WEST’,’United States’,’MAR’,2000,1345,17234,1100,1102);


Prompt **********************************************************************

Prompt Loading Acme ORG 115 MID-WEST data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’MID-WEST’,’United States’,’FEB’,2001 ,500,19521,1100,3102);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’MID-WEST’,’United States’,’MAR’,2001 ,1200,18234,1100,3102);


Prompt **********************************************************************

Prompt Loading Acme ORG 115 NORTH-EAST data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (117,’NORTH-EAST’,’United States’,’FEB’,2000,1405,35999,1100,1103);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (117,’NORTH-EAST’,’United States’,’MAR’,2000,902,18929,1100,1103);
```

```
Prompt **********************************************************************

Prompt Loading Acme ORG 115 NORTH-EAST data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (117,’NORTH-EAST’,’United States’,’FEB’,2001 ,411,11234,1100,3103);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (117,’NORTH-EAST’,’United States’,’MAR’,2001 ,1332,32123,1100,3103);


Prompt **********************************************************************

Prompt Loading Widget ORG 123 FRANCE data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’FRANCE’,’Europe’,’JAN’,2000,1242,43323,1200,1203);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’FRANCE’,’Europe’,’FEB’,2000,912,58293,1200,1203);


Prompt **********************************************************************

Prompt Loading Widget ORG 123 FRANCE data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’FRANCE’,’Europe’,’JAN’,2001 ,1000,57003,1200,3203);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (123,’FRANCE’,’Europe’,’FEB’,2001 ,989,43000,1200,3203);
```

```
Prompt **********************************************************************

Prompt Loading Widget ORG 115 SWEDEN data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’SWEDEN’,’Europe’,’JAN’,2000,234,21000,1200,1201);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’SWEDEN’,’Europe’,’FEB’,2000,553,34000,1200,1201);


Prompt **********************************************************************

Prompt Loading Widget ORG 115 SWEDEN data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’SWEDEN’,’Europe’,’JAN’,2001 ,239,19342,1200,3201);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (115,’SWEDEN’,’Europe’,’FEB’,2001 ,531,31343,1200,3201);


Prompt **********************************************************************

Prompt Loading Widget ORG 143 GERMANY data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (143,’GERMANY’,’Europe’,’JAN’,2000,1100,19402,1200,1202);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (143,’GERMANY’,’Europe’,’FEB’,2000,1400,33495,1200,1202);

Prompt **********************************************************************
Prompt Loading Widget ORG 143 GERMANY data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (143,’GERMANY’,’Europe’,’JAN’,2001 ,987,20121,1200,3202);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (143,’GERMANY’,’Europe’,’FEB’,2001 ,1443,34344,1200,3202);


Prompt **********************************************************************

Prompt Loading Widget ORG 323 ITALY data for 2000

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES

(323,’ITALY’,’Europe’,’JAN’,2000,994,18909,1200,1204);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (323,’ITALY’,’Europe’,’FEB’,2000,1375,33213,1200,1204);

Prompt **********************************************************************

Prompt Loading Widget ORG 323 ITALY data for 2001

Prompt **********************************************************************

INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (323,’ITALY’,’Europe’,’JAN’,2001 ,1030,2032,1200,3204);


INSERT INTO SALESAPP.SALES_DATA (ORG_ID, ORG_NAME,  DIV, MONTH,
YEAR, HOURS, EXPENSES, HOSTLAB, SECLAB)

VALUES (323,’ITALY’,’Europe’,’FEB’,2001 ,1503,35234,1200,3204);

COMMIT;
```

White Paper Title
January 2002
Author: Paul Needham
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com