

# Case Study: Oracle Label Security



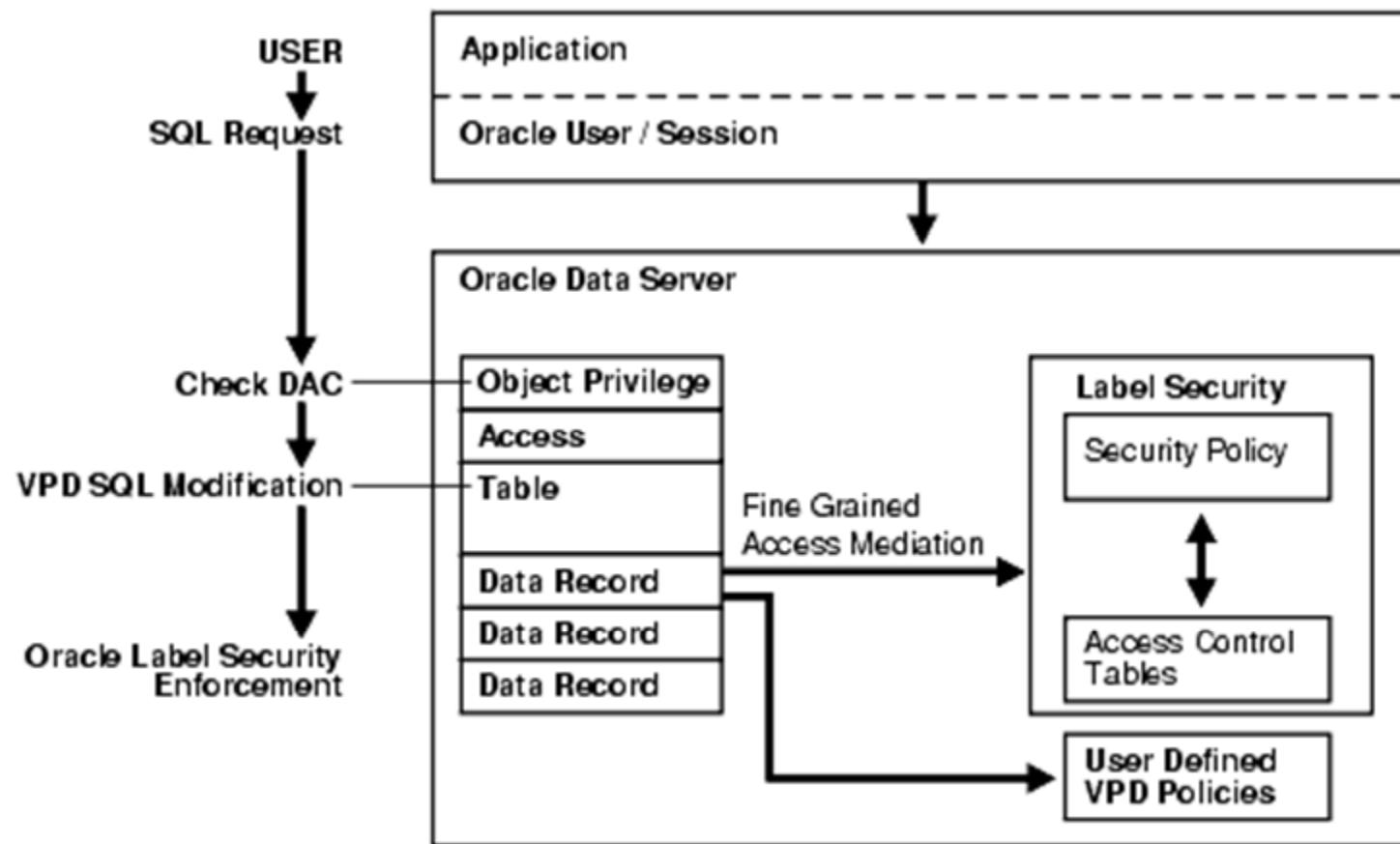
# Oracle Label Essential Concepts

- Oracle Label Security enables row-level access control, based on the virtual private database technology of Oracle Enterprise Edition
- It controls access to the contents of a row by comparing that row's label with a user's label and privileges
- Administrators can add selective row-restrictive policies to existing databases
- Developers can add label-based access control to their Oracle applications

# Oracle Label-Based Security

- Oracle label security
  - Enables row-level access control
  - Every table or view has an associated security policy
- Virtual private database (VPD) technology
  - Feature that adds predicates to user statements to limit their access in a transparent manner to the user and the application
  - Based on policies

# Oracle Label Architecture

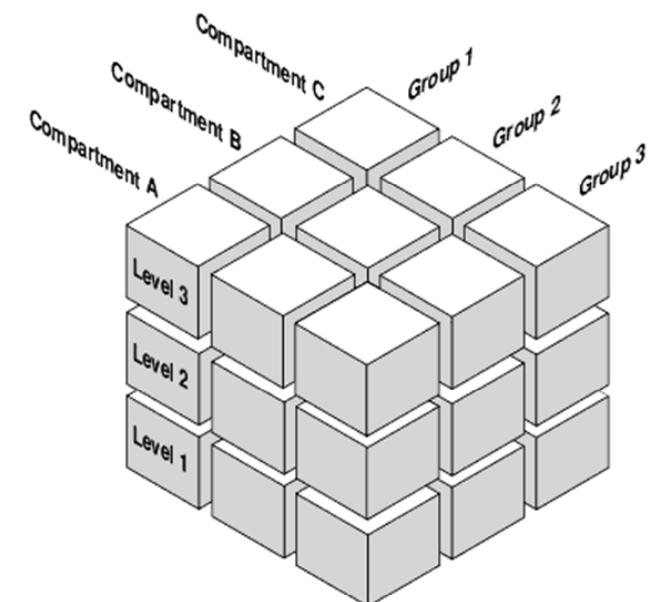
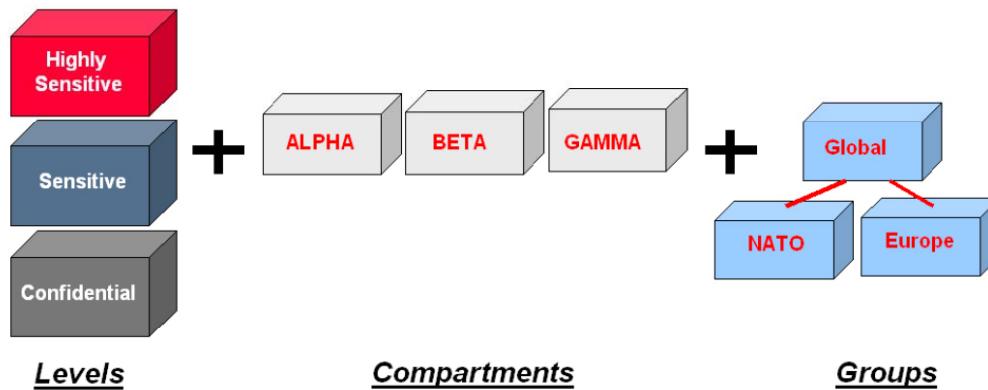


# Label policy features

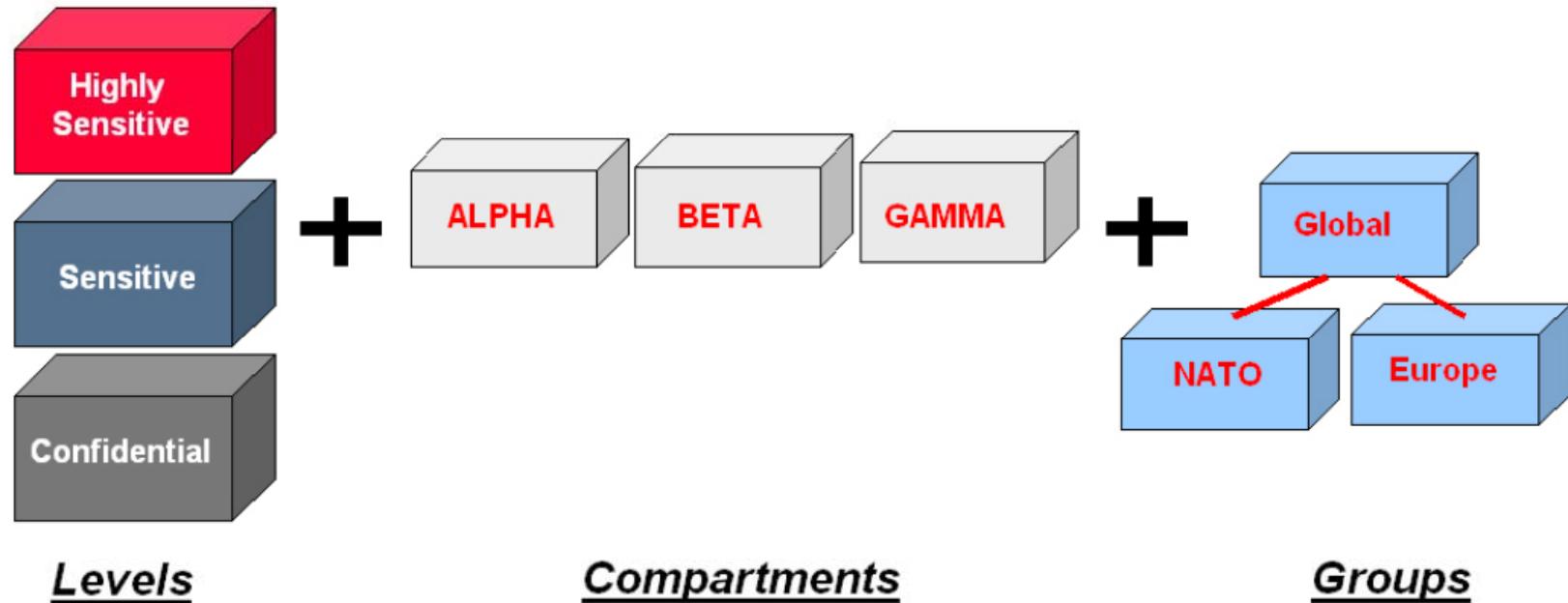
- Oracle label controls the access to data by using 3 factors:
  - The label of the data row to which access is requested
  - The label of the user session requesting access
  - The policy privileges for that user session

# Data Labels

- Every label contains three components:
  - a single level (sensitivity) ranking
  - zero or more horizontal compartments or categories
  - zero or more hierarchical groups



# Data Labels



## Example:

Confidential (10)

Highly Confidential (20)

Sensitive (30)

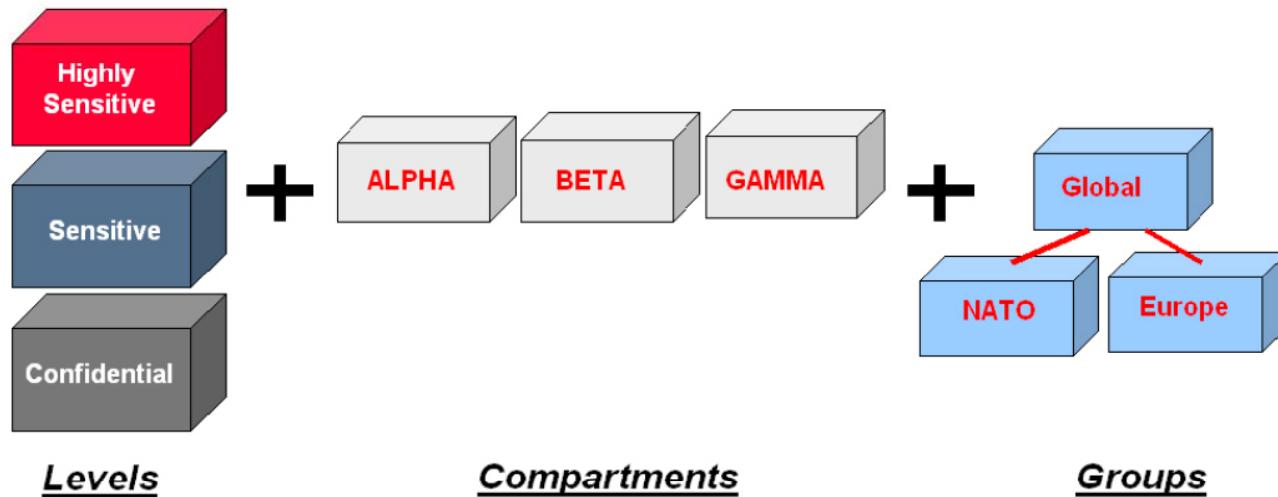
The more sensitive the information, the higher its level.

The less sensitive the information, the lower its level.

NOTE: Labels have a character form and a numeric form

# Data Labels: Compartments

- Compartments identify areas that describe the sensitivity of the labeled data, providing a finer level of granularity within a level
- The compartment component is not hierarchical



## Example:

Confidential (10)

Highly Confidential (20)

Sensitive (30)

## Compartments

## Departments:

Finance (it has sensitive and highly confidential data)

Chemical (it has sensitive data)

Operation (it has sensitive, highly confidential and confidential data)

# Data Labels: Compartments

## Levels:

Sensitive

HC

Confidential

## Compartments:

Financial

Financial

Chemical

Operation

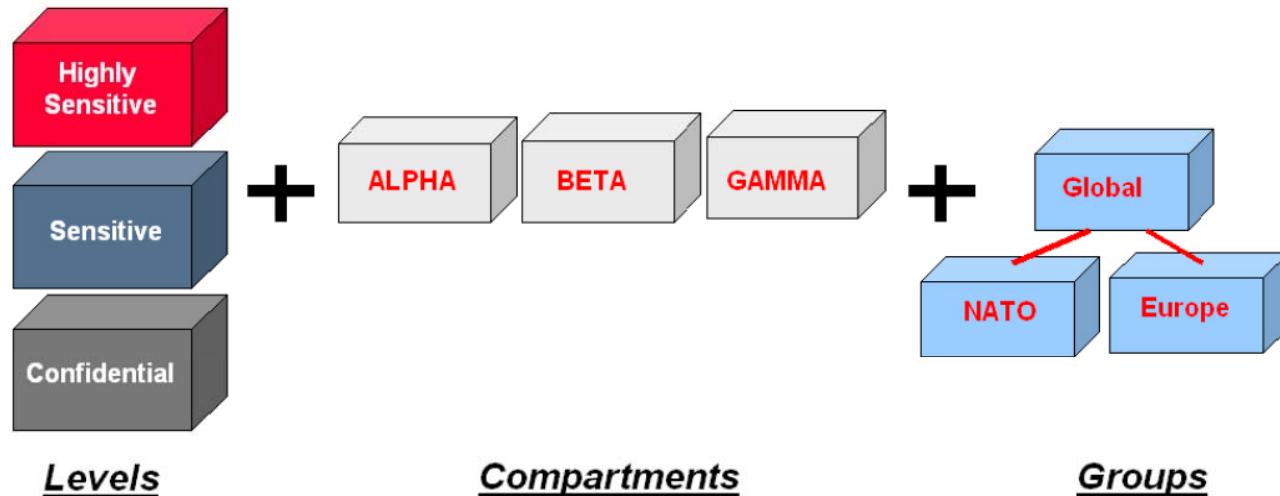
Operation

- Note that some data in the protected table may not belong to any compartment.
- If compartments are specified, then a user whose level would normally permit access to a row's data will nevertheless be prevented from such access unless the user's label also contains all the compartments appearing in that row's label.

# Data Labels: Groups

- The group component is hierarchical and is used to reflect ownership
- EXAMPLE: suppose one has two groups of users, Finance and Engineering. Users with the label Finance cannot access to data labeled Engineering (and vice versa), because they are “at the same level”
- Suppose that one has a group Board of Directors (BoD). Users in this group must be allowed to access the data of both Finance and Engineering group.
- To this end, one can establish a group hierarchy, where BoD is the group “parent” of Finance and Engineering groups

# Data Labels: Groups



## Example:

Confidential (10)

Highly Confidential (20)

Sensitive (30)

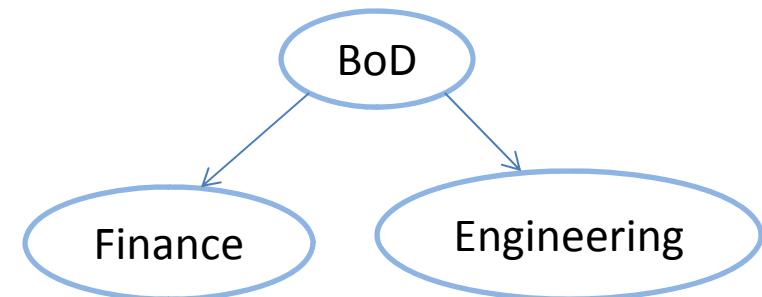
## Departments:

Finance

Chemical

Operation

## Groups:



# Data Labels

- A label can be any one of the following four combinations of components:
  - a single level component, with no groups or compartments, such as U::
  - a level and a set of compartments with no groups, such as U:Alpha, Beta:
  - a level and a set of groups with no compartments, such as U::FIN, ASIA
  - a level with both compartments and groups, such as U:Beta,Psi:ASIA,FIN

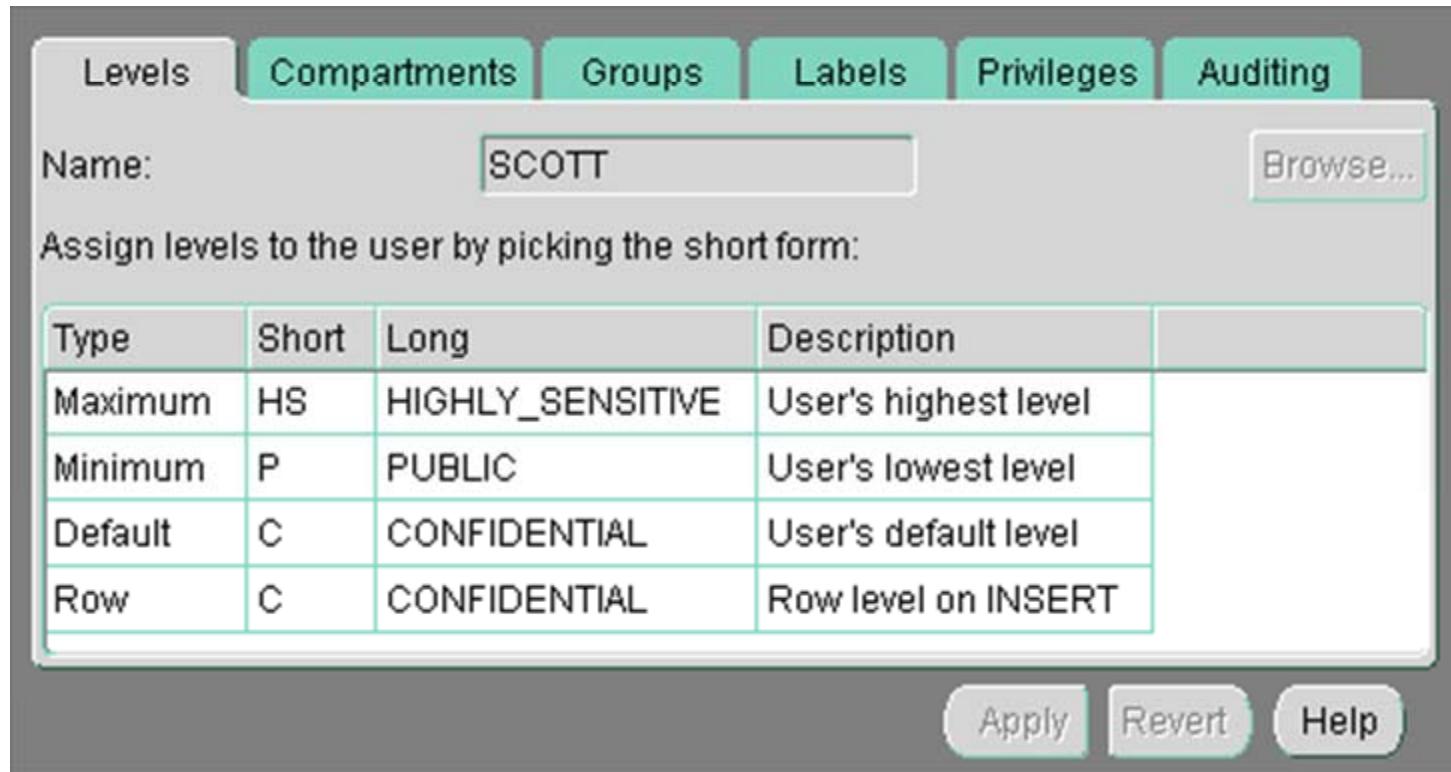
# Examples

Industry	Levels	Compartments	Groups
Defense	TOP_SECRET	ALPHA	UK
	SECRET	DELTA	NATO
	CONFIDENTIAL	SIGMA	SPAIN
	UNCLASSIFIED		
Financial Services	ACQUISITIONS	INSURANCE	CLIENT
	CORPORATE	EQUITIES	TRUSTEE
	CLIENT	TRUSTS	BENEFICIARY
	OPERATIONS	COMMERCIAL_LOANS	MANAGEMENT
		CONSUMER_LOANS	STAFF
Judicial	NATIONAL_SECURITY	CIVIL	ADMINISTRATION
	SENSITIVE	CRIMINAL	DEFENSE
	PUBLIC		PROSECUTION
			COURT
Health Care	PRIMARY_PHYSICIAN	PHARMACEUTICAL	CDC
	PATIENT_CONFIDENTIAL	INFECTIOUS_DISEASES	RESEARCH
	PATIENT_RELEASE		NURSING_STAFF
			HOSPITAL_STAFF

# User Labels

- A user label specifies that user's sensitivity level plus any compartments and groups that constrain the user's access to labeled data.
- Each user is assigned a range of levels, compartments, and groups, and each session can operate within that authorized range to access labeled data within that range.

# User Labels and level authorizations



User Default Level: The level that is assumed by default when connecting to Oracle

User Default Row Level: The level that is used by default when inserting data into Oracle

# User Labels and compartments

The screenshot shows a software interface for managing user compartments. At the top, there is a navigation bar with tabs: Levels, Compartments, Groups, Labels, Privileges, and Auditing. The 'Compartments' tab is currently selected, indicated by a dashed border. Below the tabs, a header reads "Assign compartments to the user and specify attributes:". A table is displayed, showing four compartments: OP (OPERATIONAL), FINCL (FINANCIAL), and CHEM (CHEMICAL). The fourth row is empty. Each compartment has columns for Short and Long names, WRITE access (checked for all), DEFAULT status (checked for OP and FINCL), and ROW designation (checked for OP and FINCL). A "Remove" button is located at the bottom left of the table area. At the bottom right, there are buttons for "Apply", "Revert", and "Help".

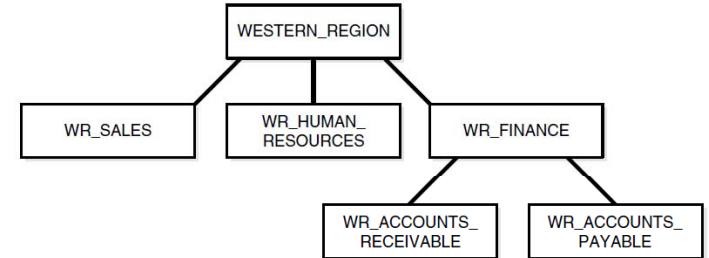
Short	Long	WRITE	DEFAULT	ROW
OP	OPERATIONAL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
FINCL	FINANCIAL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
CHEM	CHEMICAL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Remove

Apply Revert Help

- The administrator specifies the list of compartments that a user can place in her session label.
- Write access must be explicitly given for each compartment
- The Row designation indicates whether the compartment should be used as part of the default row label for newly inserted data.
- A user cannot directly insert, update, or delete a row that contains a compartment that she does not have authorization to write.

# User Labels and authorized groups



Screenshot of a user interface for managing session labels and authorized groups. The interface includes tabs for Levels, Compartments, Groups, Labels (selected), Privileges, and Auditing. A sub-section titled "Assign groups to the user and specify attributes:" contains a table and several buttons.

Short	Long	WRITE	DEFAULT	ROW	Parent
WR_HR	WR_HUMAN_RESOURCES	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	WR
WR_AP	WR_ACCOUNTS_PAYABLE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	WR_F...
WR_AR	WR_ACCOUNTS_RECEIVABLE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	WR_F...
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Buttons at the bottom include "Remove", "Apply", "Revert", and "Help".

The administrator specifies the list of groups that a user can place in her session label.

Write access must be explicitly given for each group listed.

Row designation indicates whether the group should be used as part of the default row label for newly inserted data.

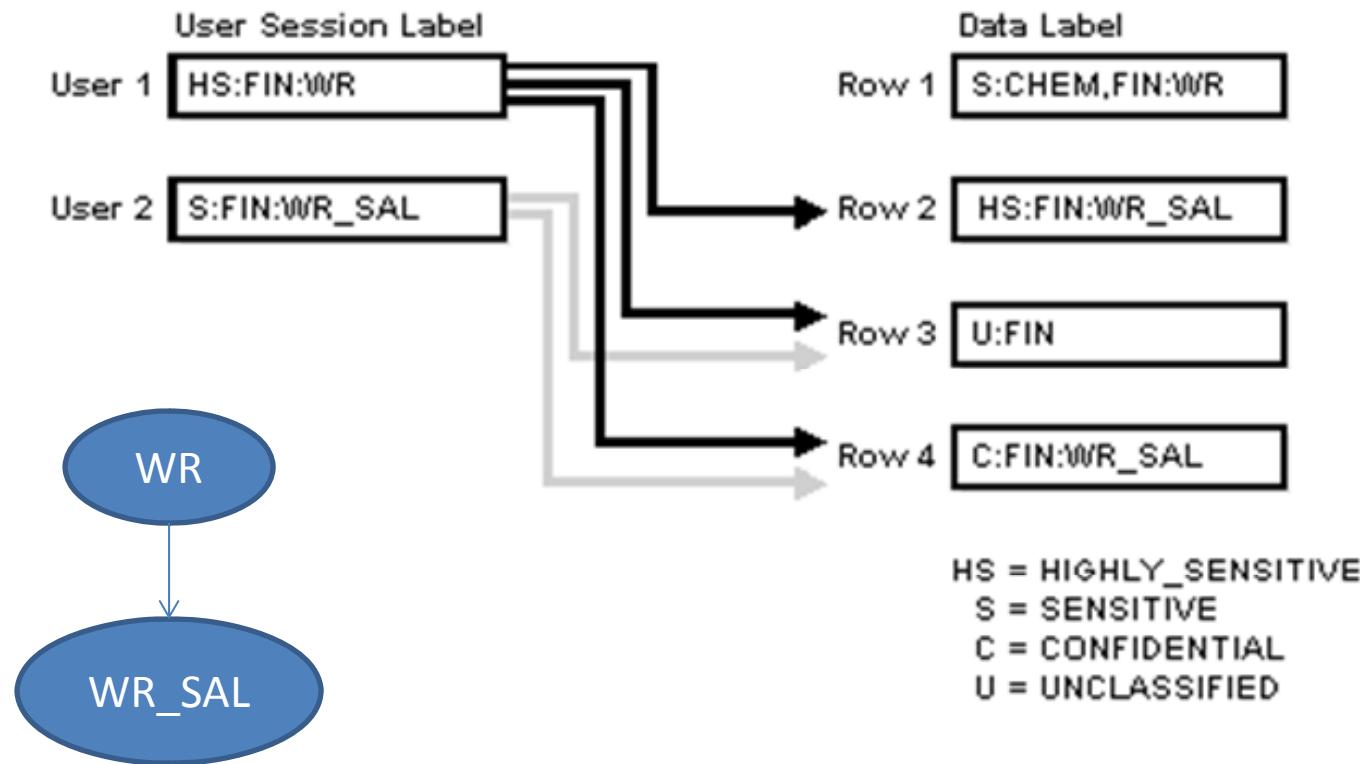
# Session Labels

- The *session label* is the particular combination of level, compartments, and groups at which a user works at any given time.
- The user can change the session label to any combination of components for which he is authorized.
- When a user writes data without specifying its label, a *row label* is assigned automatically, using the user's session label.

# How Data Labels and User Labels Work Together

- Each Oracle Label Security user can only access data within the range of his or her own label authorizations.
- Each user has:
  - Maximum and minimum levels
  - A set of authorized compartments
  - A set of authorized groups (and, implicitly, authorization for any subgroups)
  - For each compartment and group, a specification of read-only access, or read/write access
- Example:
  - if a user is assigned a maximum level of Highly Confidential, then the user potentially has access to Highly Confidential, and Confidential data. The user has no access to Sensitive data.

# How Data Labels and User Labels Work Together



# Policy Privileges

- The policy privileges enable a user or a stored program unit to **bypass** some aspects of the label-based access control policy
- The administrator can also authorize the user or program unit to perform specific actions, such as the ability of one user to assume the authorizations of a different user
- Privileges can be granted to program units, authorizing the procedure, rather than the user, to perform privileged operations

# Privileges in Oracle Label Security Policies

- Oracle Label Security supports special privileges that allow authorized users to *bypass certain parts of the policy*.

---

## Security Privilege Explanation

---

READ	Allows read access to all data protected by the policy
FULL	Allows full read and write access to all data protected by the policy
COMPACCESS	Allows a session access to data authorized by the row's compartments, independent of the row's groups
PROFILE_ACCESS	Allows a session to change its labels and privileges to those of a different user
WRITEUP	Allows users to set or raise only the level, within a row label, up to the maximum level authorized for the user. (Active only if LABEL_UPDATE is active.)
WRITEDOWN	Allows users to set or lower the level, within a row label, to any level equal to or greater than the minimum level authorized for the user. (Active only if LABEL_UPDATE is active.)
WRITEACROSS	Allows a user to set or change groups and compartments of a row label, but does not allow changes to the level. (Active only if LABEL_UPDATE is active.)

# Privileges in Oracle Label Security Policies

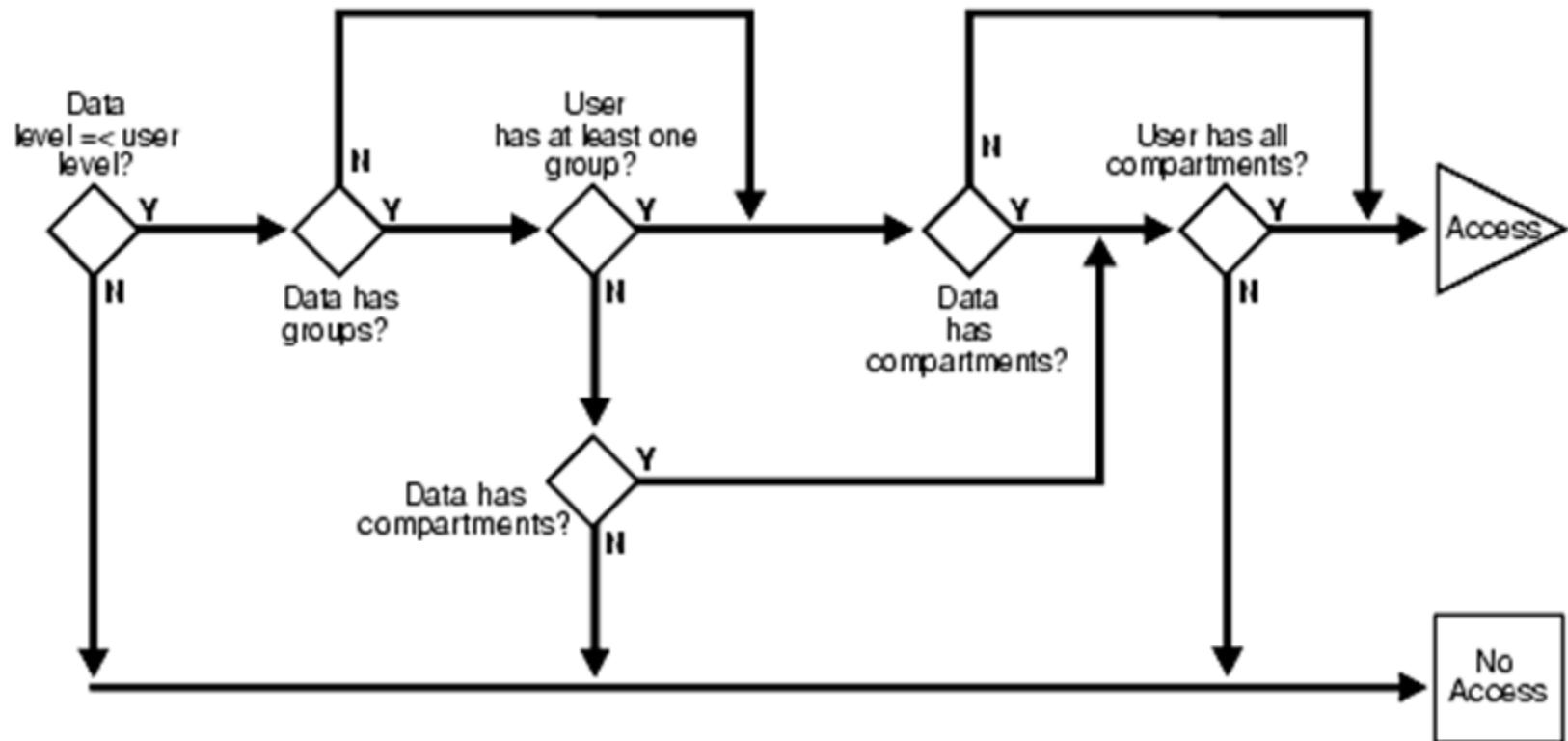
- READ
  - A user with READ privilege can read all data protected by the policy, regardless of his authorizations or session label. The user does not even need to have label authorizations.
  - A user with READ privilege can *write to any data rows for which* he or she has write access, based on any label authorizations.
  - The READ privilege enables optimal performance on SELECTs, since the system behaves as though the Oracle Label Security policy were not even present.
  - Useful
    - for system administrators who need to export data, but who should not be allowed to change data.
    - for people who must run reports and compile information, but not change data.
- FULL
  - The FULL privilege has the same effect and benefits as the READ privilege, with one difference: a user with FULL privilege can also *write to all the data*.

# Privileges in Oracle Label Security Policies

## COMPACCESS

- The COMPACCESS privilege allows a user to access data based on the row label's compartments, independent of the row label's groups.
- If a row label has no compartments, then access is determined by the group authorizations. However, when compartments do exist, and access to them is authorized, then the group authorization is bypassed.
- This allows a privileged user whose label matches all the compartments of the data to access any data in any particular compartment, independent of what groups may own or otherwise be allowed access to the data.

# *Label Evaluation Process for Read Access with COMPACCESS Privilege*



# Privileges in Oracle Label Security Policies

## PROFILE\_ACCESS

- The PROFILE\_ACCESS privilege allows a session to change its session labels and session privileges to those of a different user.
- This is a very powerful privilege, since the user can potentially become a user with FULL privileges.
- This privilege cannot be granted to a trusted stored program unit.

# Special Row Label Privileges

- Once the label on a row has been set, Oracle Label Security privileges are required to modify the label.
- These privileges include WRITEUP, WRITEDOWN, and WRITEACROSS.

# Special Row Label Privileges

- WRITEUP
  - The WRITEUP privilege enables the user to raise the level of data within a row, without compromising the compartments or groups.
  - The user can raise the level up to his or her maximum authorized level.
    - For example, an authorized user can raise the level of a data row that has a level lower than his own minimum level. If a row is UNCLASSIFIED and the user's maximum level is SENSITIVE, he can raise the row's level to SENSITIVE.
  - He can raise the level above his current session level, but cannot change the compartments.

# Special Row Label Privileges

- WRITEDOWN
  - The WRITEDOWN privilege enables the user to lower the level of data within a row, without changing the compartments or groups. The user can lower the level to any level equal to or greater than his or her minimum authorized level.
- WRITEACROSS
  - The WRITEACROSS privilege allows the user to change the compartments and groups of data, without altering its sensitivity level.

# Documentation

- **Oracle® Label Security Administrator's Guide**
- **11g Release 2 (11.2) E10745-02**
  - [http://docs.oracle.com/cd/E11882\\_01/network.12/e10745.pdf](http://docs.oracle.com/cd/E11882_01/network.12/e10745.pdf)

# Virtual Private Databases



1

## Oracle VPD

- Virtual Private Database (VPD)
  - Sometimes referred to as Oracle Row-Level Security (RLS) or Fine Grained Access Control (FGAC)
  - **Fine-grained access control:** associate security policies to database objects
  - **Application Context:** define and access application or session attributes, and use them in access control, e.g., for implementing temporal access control
- Combining these two features, VPD enables administrators to define and **enforce row-level access control policies** based on **session attributes**

2

# Why VPD?

- **Scalability**

- Table Customers contains 1,000 customer records. Suppose we want customers to access their own records only. Using views, we need to create 1,000 views. Using VPD, it can be done with a single policy function.

- **Simplicity**

- Say, we have a table T and many views are based on T. Suppose we want to restrict access to some information in T. Without VPD, all view definitions have to be changed. Using VPD, it can be done by attaching a policy function to T; as the policy is enforced in T, the policy is also enforced for all the views that are based on T.

- **Security**

- Server-enforced security (as opposed to application-enforced).
  - Cannot be bypassed.

3

# Oracle VPD

- **How does it work?**

When a user accesses a table (or view or synonym) which is protected by a VPD policy (function),

1. The Oracle server invokes the policy function.
2. The policy function returns a predicate, based on session attributes or database contents.
3. The server dynamically rewrites the submitted query by appending the returned predicate to the WHERE clause.
4. The modified SQL query is executed.

4

## Oracle VPD: Example

- Suppose Alice has/owns the following table.

```
my_table(owner varchar2(30), data varchar2(30));
```

- Suppose we want to implement the following policy:
  - Users can access only the data of their own. But Admin should be able to access any data without restrictions.

5

## Oracle VPD: Example

### 1. Create a policy function

```
Create function sec_function(p_schema varchar2, p_obj varchar2)
Return varchar2
As
    user VARCHAR2(100);
Begin
    if ( SYS_CONTEXT('userenv', 'ISDBA') ) then
        return '';
    else
        user := SYS_CONTEXT('userenv', 'SESSION_USER');
        return 'owner = ' || user;
    end if;
End;

// userenv = the pre-defined application context
// p_obj is the name of the table or view to which the policy will apply
// p_schema is the schema owning the table or view
```

6

## SYS\_CONTEXT

- In Oracle/PLSQL, the sys\_context function is used to retrieve information about the Oracle environment.
- The syntax for the sys\_context function is:  
`sys_context( namespace, parameter, [ length ] )`
- *namespace is an Oracle namespace that has already been created.*
- If the namespace is 'USERENV', attributes describing the current Oracle session can be returned.
- *parameter is a valid attribute that has been set using the DBMS\_SESSION.set\_context procedure.*
- *length is optional. It is the length of the return value in bytes. If this parameter is omitted or if an invalid entry is provided, the sys\_context function will default to 256 bytes*

7

## USERENV Namespace Valid Parameters

Parameter	Explanation	Return Length
AUDITED_CURSORID	Returns the cursor ID of the SQL that triggered the audit	N/A
AUTHENTICATION_DATA	Authentication data	256
AUTHENTICATION_TYPE	Describes how the user was authenticated. Can be one of the following values: Database, OS, Network, or Proxy	30
BG_JOB_ID	If the session was established by an Oracle background process, this parameter will return the Job ID. Otherwise, it will return NULL	30
CLIENT_IDENTIFIER	Returns the client identifier (global context)	64
CLIENT_INFO	User session information	64
CURRENT_SCHEMA	Returns the default schema used in the current schema	30
CURRENT_SCHEMAID	Returns the identifier of the default schema used in the current schema	30
CURRENT_SQL	Returns the SQL that triggered the audit event	64
CURRENT_USER	Name of the current user	30
CURRENT_USERID	Userid of the current user	30
DB_DOMAIN	Domain of the database from the DB_DOMAIN initialization parameter	256
DB_NAME	Name of the database from the DB_NAME initialization parameter	30
ENTRYID	Available auditing entry identifier	30
EXTERNAL_NAME	External of the database user	256
GLOBAL_CONTEXT_MEMORY	The number used in the System Global Area by the globally accessed context	N/A
HOST	Name of the host machine from which the client has connected	54
INSTANCE	The identifier number of the current instance	30

8

## USERENV Namespace Valid Parameters

ISDBA	Returns TRUE if the user has DBA privileges. Otherwise, it will return FALSE.	30
LANG	The ISO abbreviate for the language	62
LANGUAGE	The language, territory, and character of the session. In the following format: language_territory.charset	52
NETWORK_PROTOCOL	Network protocol used	256
NLS_CALENDAR	The calendar of the current session	62
NLS_CURRENCY	The currency of the current session	62
NLS_DATE_FORMAT	The date format for the current session	62
NLS_DATE_LANGUAGE	The language used for dates	62
NLS_SORT	BINARY or the linguistic sort basis	62
NLS_TERRITORY	The territory of the current session	62
OS_USER	The OS username for the user logged in	30
PROXY_USER	The name of the user who opened the current session on behalf of SESSION_USER	30
PROXY_USERID	The identifier of the user who opened the current session on behalf of SESSION_USER	30
SESSION_USER	The database user name of the user logged in	30
SESSION_USERID	The database identifier of the user logged in	30
SESSIONID	The identifier of the auditing session	30
TERMINAL	The OS identifier of the current session	10

9

## Oracle VPD: Example

### 2. Attach the policy function to my\_table

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                             object_name => 'my_table',
                             policy_name => 'my_policy',
                             function_schema => 'Alice',
                             policy_function => 'sec_function',
                             statement_types => 'select, update, insert',
                             update_check => TRUE );
```

- The VPD security model uses the Oracle *dbms\_rls package* (*RLS* stands for row-level security)
- *update\_check*: Optional argument for INSERT or UPDATE statement types. The default is FALSE. Setting *update\_check* to TRUE causes the server to also check the policy against the value after insert or update.

10

## DBMS\_RLS.ADD\_POLICY syntax

- DBMS\_RLS.ADD\_POLICY (  
object schema IN VARCHAR2 NULL,  
object\_name IN VARCHAR2,  
policy\_name IN VARCHAR2,  
function\_schema IN VARCHAR2 NULL,  
policy\_function IN VARCHAR2,  
statement\_types IN VARCHAR2 NULL,  
update\_check IN BOOLEAN FALSE,  
enable IN BOOLEAN TRUE,  
static\_policy IN BOOLEAN FALSE,  
policy\_type IN BINARY\_INTEGER NULL,  
long\_predicate IN BOOLEAN FALSE,  
sec\_relevant\_cols IN VARCHAR2,  
sec\_relevant\_cols\_opt IN BINARY\_INTEGER NULL);

11

## Oracle VPD-Example

### 3. Bob accesses my\_table

```
select * from my_table;  
=> select * from my_table where owner = 'bob';  
- only shows the rows whose owner is 'bob'
```

insert into my\_table values('bob', 'Some data'); **OK!**

insert into my\_table values('alice', 'Other data'); **NOT OK!**  
- because of the check option

12

## Policy Commands

- ADD\_POLICY – creates a new policy
- DROP\_POLICY – drops a policy

```
DBMS_RLS.DROP_POLICY (
    object schema IN VARCHAR2 NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2);
```
- ENABLE\_POLICY – enables or disables a fine-grained access control policy

```
DBMS_RLS.ENABLE_POLICY (
    object schema IN VARCHAR2 NULL,
    object_name IN VARCHAR2,
    policy_name IN VARCHAR2,
    enable IN BOOLEAN );
enable - TRUE to enable the policy, FALSE to disable the policy
```

13

## Column-level VPD

- Instead of attaching a policy to a whole table or a view, attach a policy only to security-relevant columns
  - Default behavior: restricts the number of rows returned by a query.
  - Masking behavior: returns all rows, but returns NULL values for the columns that contain sensitive information.
- Restrictions
  - Applies only to ‘select’ statements
  - The predicate must be a simple Boolean expression.

14

## Column-level VPD: Example

- Suppose Alice has (owns) the following table.

```
Employees(e_id number(2), name varchar2(10), salary number(3));
```

e_id	Name	Salary
1	Alice	80
2	Bob	60
3	Carl	99

- Policy: Users can access e\_id's and names without any restriction. But users can access only their own salary information.

15

## Column-level VPD: Example

### 1. Create a policy function

```
Create function sec_function(p_schema varchar2, p_obj
                           varchar2)
Return varchar2
As
  user VARCHAR2(100);
Begin
  user := SYS_CONTEXT('userenv', 'SESSION_USER');
  return 'name = ' || user;
End;
```

16

## Column-level VPD: Example

### 2. Attach the policy function to Employees (default behavior)

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                               object_name => 'employees',
                               policy_name => 'my_policy',
                               function_schema => 'Alice',
                               policy_function => 'sec_function',
                               sec_relevant_cols=>'salary');
```

17

## Column-level VPD: Example

### 3. Bob accesses table Employees (default behavior).

REMEMBER: default behavior restricts the number of rows returned by a query

```
select e_id, name from Employee;
```

e_id	Name
1	Alice
2	Bob
3	Carl

```
select e_id, name, salary from Employee;
```

e_id	Name	Salary
2	Bob	60

18

## Column-level VPD: Example

### 2'. Attach the policy function to Employees (**masking behavior**)

```
execute dbms_rls.add_policy (object_schema => 'Alice',
                               object_name => 'employees',
                               policy_name => 'my_policy',
                               function_schema => 'Alice',
                               policy_function => 'sec_function',
                               sec_relevant_cols=>'salary',
                               sec_relevant_cols_opt=>dbms_rls.ALL_ROWS);
```

19

## Column-level VPD: Example

### 3. Bob accesses table Employees (**masking behavior**).

REMEMBER: Masking behavior returns all rows, but returns NULL values for the columns that contain sensitive information.

```
select e_id, name from Employee;
```

e_id	Name
1	Alice
2	Bob
3	Carl

```
select e_id, name, salary from Employee;
```

e_id	Name	Salary
1	Alice	
2	Bob	60
3	Carl	

20

## Application Context

- Application contexts act as secure caches of data that may be used by a fine-grained access control policy.
  - Upon logging into the database, Oracle sets up an application context in the user's session.
  - You can define, set and access application attributes that you can use as a secure data cache.
- There is a pre-defined application context, “*userenv*”.
  - See Oracle Security Guide.

21

## Application Context

- One can create a customized application context and attributes.
  - Say, each employee can access a portion of the Customers table, based on the job-position.
  - For example, a clerk can access only the records of the customers who live in a region assigned to him. But a manager can access any record.
  - Suppose that the job-positions of employees are stored in a LDAP server (or in the Employee table).
  - Such information can be accessed and cached in an application context when an employee logs in.
- To set an attribute value in an application context,
  - `DBMS_SESSION.SET_CONTEXT('namespace', 'attributename', value);`
- To get an attribute value from an application context,
  - `SYS_CONTEXT('namespace', 'attributename');`

22

# Create Application Context

## 1. Create a PL/SQL package that sets the context

```
Create package Set_emp_env IS
    procedure Set_job_position IS
        jp varchar(100);
    begin
        select job_pos into jp from Employee
        where name = SYS_CONTEXT('USERENV', 'SESSION_USER');
        DBMS_SESSION.SET_CONTEXT('emp_env', 'job', jp);
    end;
End;
```

## 2. Create a context and associate it with the package

```
Create Context emp_env Using Emp_env_context;
```

- Any attribute in the “emp\_env” context can only be set by procedures in the “Emp\_env\_context” package.

23

# Using Application Context

## 3. Set the context before users retrieve data (at the login)

```
Create or Replace Trigger Emp_trig
    After Logon On Database
    Begin
        Emp_env_context. Set_job_position
    End
```

- Use an event trigger on login to pull session information into the context.

## 4. Use the context in a VPD function

```
if (SYS_CONTEXT('emp_env', 'job') = 'manager')
    return '';
else ...
```

24

## Multiple Policies

- It is possible to associate multiple policies to a database object.
  - The policies are enforced with AND syntax.
  - For example, suppose table T is associated with {P1, P2, P3}.
  - When T is accessed by query Q = select A from T where C.
  - Q' = select A from T where C  $\wedge$  (c1  $\wedge$  c2  $\wedge$  c3).

25

## Issue 1: Inconsistencies

- Suppose the policy authorize each employee to see his/her own salary
- Alice issues the following query:

```
SELECT AVG(*) FROM Employee
```
- The query will be rewritten to

```
SELECT AVG(*) FROM Employee where name = "Alice";
```
- What's the problem?

26

## Issue 2: Recursion

- Although one can define a policy against a table, one *cannot* select that table from within the policy that was defined against the table
  - That is, a policy function of an object should not access the object.
  - Suppose that a policy function PF that protects a table T accesses T.
  - When T is accessed, PF is invoked. PF tries to access T, and another PF is invoked. This results in endless function invocations.
- This cyclic invocation can occur in a longer chain.
  - For example, define a policy function for T, that accesses another table  $T_1$ . If  $T_1$  is protected by another policy function that refers to T, then we have a cycle.
  - It is hard to check. (A policy function can even invoke a C program.)

27

## Summary

- VPD provides a very powerful access control.
- It is difficult, if not impossible, to verify whether or not a particular user has access to a particular data item in a particular table in a particular state.
  - Such verification requires checking all policy functions.
  - As policy functions are too “flexible”, it is computationally impossible to analyze them.

28

# Database Vault

---

# Why Database Vault?

- Protecting Access to Application Data
  - “Legal says our DBA should not be able to read financial records, but the DBA needs to access the database to do her job. What do we do?”
  - “Our auditors require that we separate account creation from granting privileges to accounts.”
  - “No user should be able to by-pass our application to access information in the database directly.”
  - “New DBAs should not be able to make database changes without a senior DBA being present.”

# Why Database Vault?

- Regulations such as Sarbanes-Oxley (SOX) and Graham-Leach Bliley Act (GLBA), and Basel II require **Strong Internal Controls** and **Separation of Duty**
- Internal threats are a much bigger concern today require enforcement of operational security policies - **Who, When, Where** can data be accessed?
- Database consolidation strategy requires preventive measures against access to application data by **Powerful (DBA)** users

# Common Security Problems

- I have requirements around SOX and PCI, how can I prevent my DBA from looking at the application data, including Credit Cards and Personal Information?
- How can I prevent unauthorized modifications to my application and database?



Ad-Hoc Query

On Financial Data



Applications



Ad-Hoc Query

Tool

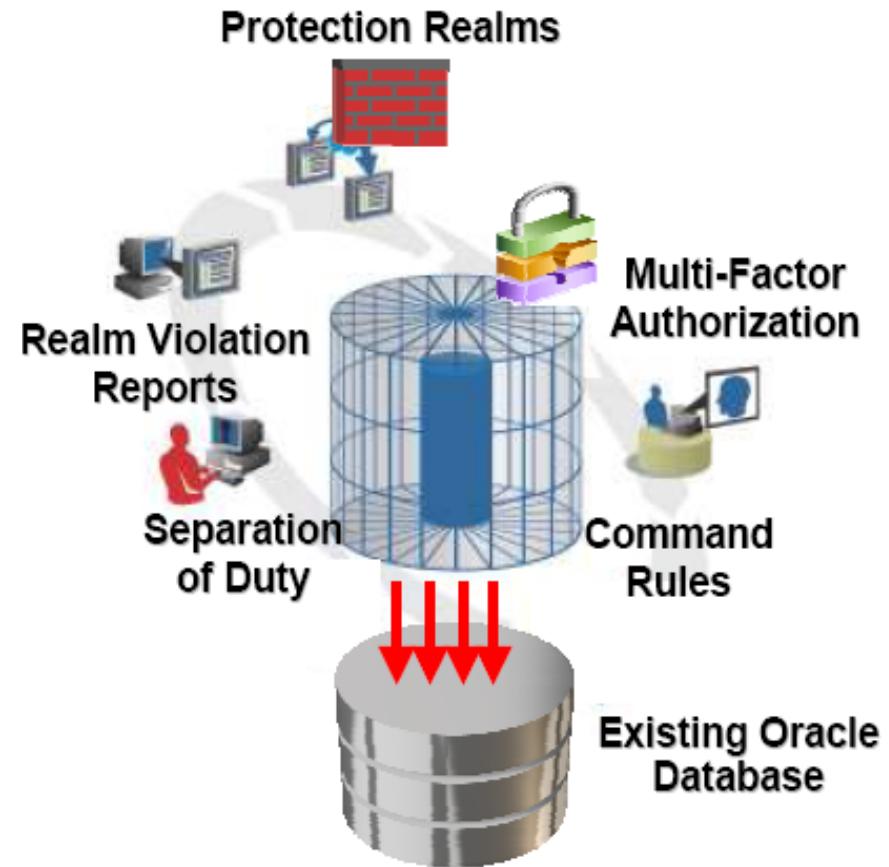


Remote DBA Services

# Oracle Database Vault

## Feature Overview

- Controls on privileged users
  - Restrict privileged users from accessing application data
  - Enforces separation of duty
- Real time access controls
  - Controls access based on IP address, authentication method, time of day,....
- Transparency
  - No changes to applications required



# Database Vault

## True “Separation of Duty”

- Protect any database object from any users (*realm*)
  - Function, job, package, synonym, trigger, view, table
  - Prevent users from viewing application data
- Prevent DBA users from creating powerful users
- Any user from executing a command (*command rule*)
  - Alter table, drop user, insert, create index, analyze
- Protect object from schema owner
  - HR user cannot modify HR objects
- Leverage sys\_context (*multi-factor authorization*)
  - Only modify database structure from local IP
  - Only accept DML statement based on date or time
  - Leverage built-in or user defined factors
    - Machine, User, Domain, Language, Protocol, etc.

# Command Rule Flexibility

Alter Database	Alter Database	Alter Table
Alter Function	Audit	Alter Tablespace
Alter Package Body	Alter Procedure	Alter Profile
Alter Session	Alter System	Alter Synonym
Alter Table	Alter Trigger	Alter User
Password	Alter Tablespace	Alter View
Change Password	Connect	Comment
Create Function	Create Index	Create Package
Create Database Link	Create Procedure	Create Role
Create Package Body	Create User	Create View
Create Table	Grant	Insert
Noaudit	Rename	Lock Table
Create Tablespace	Create Trigger	Truncate Table
Update	Insert	Delete
Execute	Select	

# Built-In Factors

<b>Authentication Method</b>	<b>Session User</b>	<b>Client IP</b>	<b>Database Name</b>
<b>Domain</b>	<b>Machine</b>	<b>Database Domain</b>	<b>Database Instance</b>
<b>Network Protocol</b>	<b>Database IP</b>	<b>Enterprise Identity</b>	<b>Proxy Enterprise Identity</b>
<b>Language</b>	<b>Database Hostname</b>	<b>Date</b>	<b>Time</b>

\* Additional factors can be defined

# Web Based Administrative Interface

ORACLE Database Vault

Database Instance: orcl

[Administration](#) [Database Vault Reports](#) [General Security Reports](#) [Monitor](#)

The links below allow you to protect applications and data using Oracle Database Vault features that include:  
Application Roles.

## Database Vault Feature Administration

[Realms](#) Realms  
[Command Rules](#)  
[Factors](#)  
[Rule Sets](#)  
[Secure Application Roles](#)  
[Label Security Integration](#)

[Administration](#) [Database Vault Reports](#) [General Security Reports](#) [Monitor](#)

## Web Based Management

- Realms
- Rules
- Factors
- Reports
- Dashboard

Copyright © 1996, 2006, Oracle. All rights reserved.

[About Oracle Database Vault Administrator](#)

[Database](#) | [Help](#) | [Logout](#)

ORACLE

# Oracle Database Vault Reports

Database Instance: orcl

Administration Database Vault Reports General Security Reports Monitor

Use this screen to run reports about potential Database Vault configuration issues and Database Vault audit events.

Run Report

[Expand All](#) | [Collapse All](#)

Reports

Select	Focus	Report Title
<input type="radio"/>		▼ Reports
<input type="radio"/>	<input type="radio"/>	▼ Database Vault Configuration Issues Reports
<input checked="" type="radio"/>		Command Rule Configuration Issues
<input type="radio"/>		Factor Configuration Issues
<input type="radio"/>		Factors Without Identities
<input type="radio"/>		Identity Configuration Issues
<input type="radio"/>		Realm Authorization Configuration Issues
<input type="radio"/>		Rule Set Configuration Issues
<input type="radio"/>		Secure Application Configuration Issues
<input type="radio"/>	<input type="radio"/>	▼ Database Vault Auditing Reports
<input type="radio"/>		Realm Audit
<input type="radio"/>		Command Rule Audit
<input type="radio"/>		Factor Audit

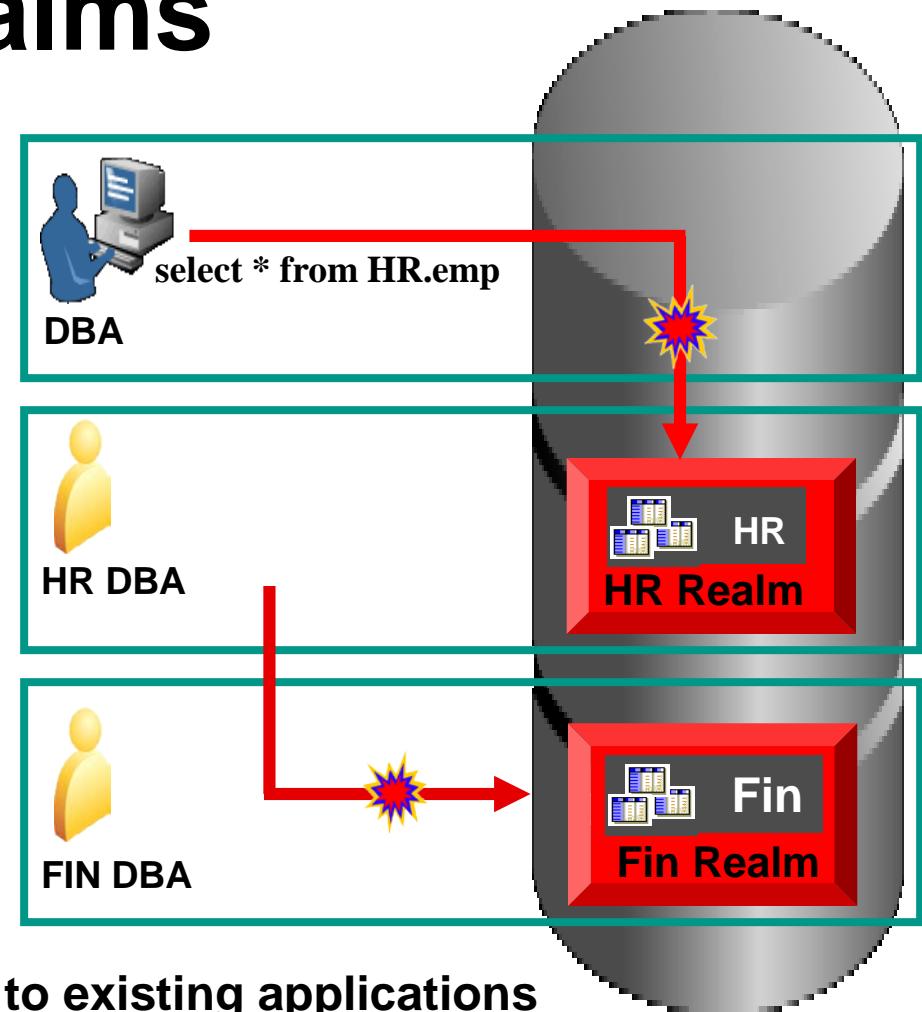
## Database Vault Reporting

- Over 3 dozen security reports for compliance
- Audit violation attempts
- Realm, Rule and Factor Reports
- System and Public Privileges

# Oracle Database Vault Realms

- Database DBA views HR data  
**Compliance and protection from insiders**

- HR DBA views Fin. data  
**Eliminates security risks from server consolidation**



Realms can be easily applied to existing applications  
with minimal performance impact

# Oracle Database Vault

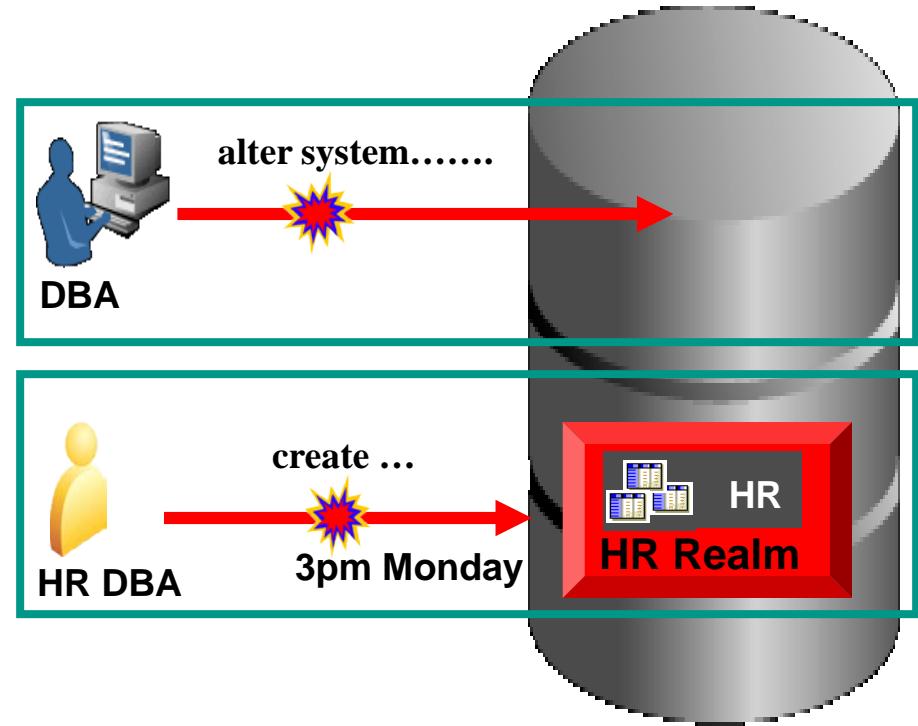
## Rules & Multi-factor Authorization

- Database DBA attempts remote “*alter system*”

**Rule based on IP Address blocks action**

- HR DBA performs unauthorized actions during production

**Rule based on Date and Time blocks action**



Factors and Command Rules provide flexible and adaptable security controls

# Oracle System User Blocked

```
Oracle SQL*Plus
File Edit Search Options Help

SQL*Plus: Release 10.1.0.2.0 - Production on Wed Apr 12 10:54:57 2006

Copyright (c) 1982, 2004, Oracle. All rights reserved.

Connected to:
Oracle Data Vault Release 10.2.0.1.0 - Development
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Data Vault options

SQL> show user
USER is "SYSTEM"
SQL>
SQL> @demo
SQL>
SQL> select user, employee_id, last_name, ssn, salary from hr.employees
  2  where employee_id < 117
  3 /
select user, employee_id, last_name, ssn, salary from hr.employees
                           *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

# Database Vault Rules and Factors Block(Remote Intranet Connection)



Oracle SQL\*Plus

File Edit Search Options Help

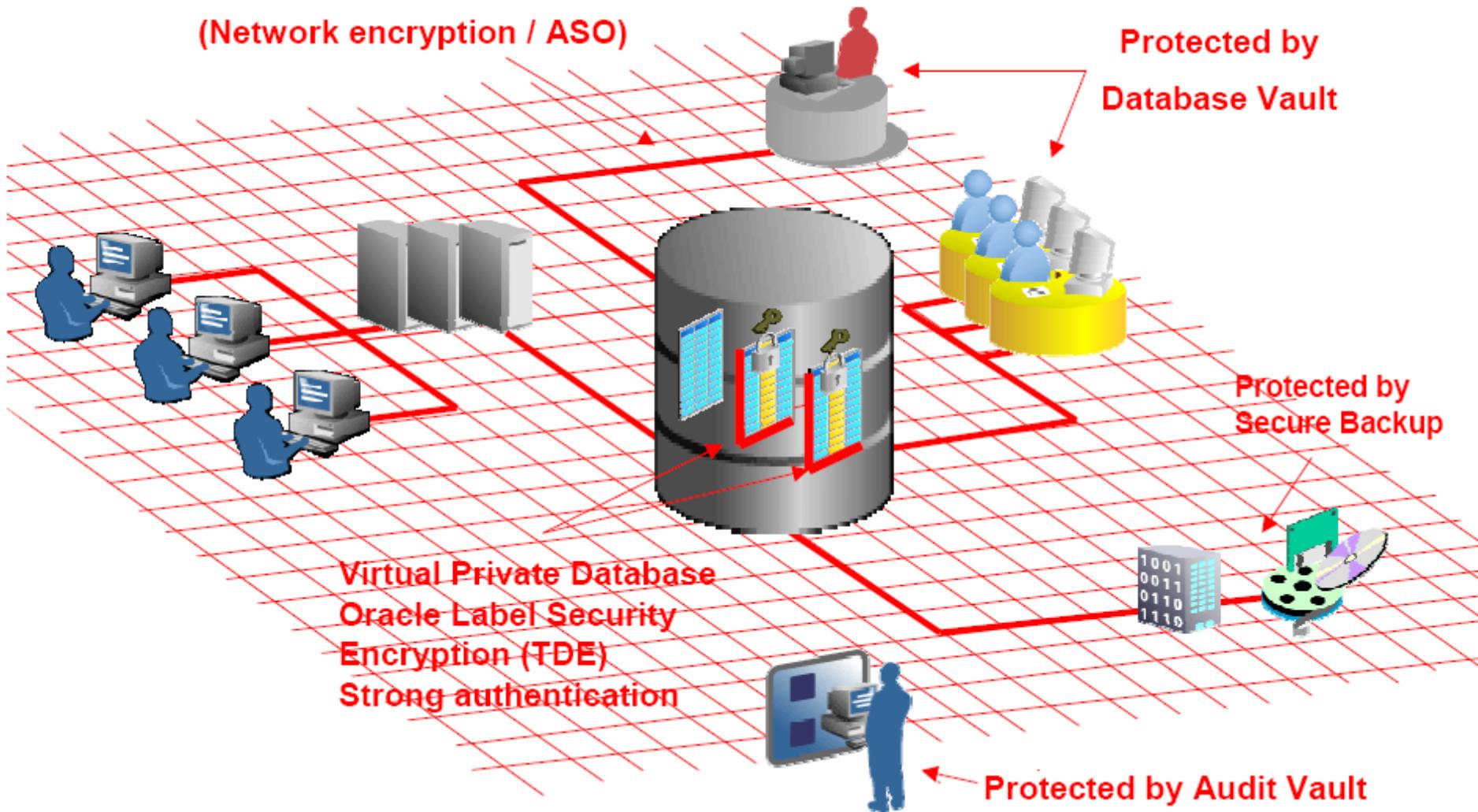
```
SQL*Plus: Release 10.1.0.2.0 - Production on Wed Apr 12 10:11:56 2006
Copyright (c) 1982, 2004, Oracle. All rights reserved.

Connected to:
Oracle Data Vault Release 10.2.0.1.0 - Development
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Oracle Data Vault options

SQL>
SQL> show user
USER is "SYSTEM"
SQL>
SQL> alter system switch logfile;
alter system switch logfile
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> |
```

# Oracle secured DB environment



# Hands-on Resources

- **Oracle Database Vault:**

<http://www.oracle.com/technetwork/database/options/database-vault/index.html>

- **Oracle Security Overview:**

<http://www.oracle.com/technology/deploy/security/database-security/index.html>

- Lab3-1: Protect Application Data from DBA and Privileged Users (no submission)

<http://st-curriculum.oracle.com/obe/db/11g/r1/prod/security/datavault/datavault.htm>

- Lab3-2: Restrict DBA commands based on IP address (no submission)

<http://st-curriculum.oracle.com/obe/db/11g/r1/prod/security/datavault/datavault2.htm>

# Oracle Database Vault Secured Installation

- Disallows connections with SYSDBA
  - Will affect
    - Oracle Data Guard and Data Guard Broker command line utilities
    - Oracle Recovery Manager command line utility
    - Oracle Real Application Clusters svrctl utility
    - Oracle ASM command line utilities
    - Custom DBA scripts
  - Can be re-enabled with the orapwd utility
- Enables password file and Turns off OS authentication
  - (e.g. sqlplus "/" as SYSDBA)

# Oracle Database Vault Secured Installation

- Requires Oracle Label Security version 10.2.0.2
- Requires one of the following:
  - Enterprise Manager 10.2.0.2
  - 10g Application Server Containers for J2EE (OC4J)
- Cannot be installed into an Oracle home that contains an ASM instance
- Best practice is to create a database vault owner and database vault manager
- Requires 270 MB of disk space for DB Vault software
- Requires 400 MB of /tmp disk space
- OS authentication is turned off for all databases in the Oracle home
- Database vault can be enabled for each database in the Oracle home (optional)