


PURDUE UNIVERSITY

Computer Security

CS 426

Lecture 20

Database Security
Labeled Databases



Elisa Bertino
Purdue University
IN, USA
bertino@cs.purdue.edu

Center for Education and Research
in Information Assurance and Security

PURDUE UNIVERSITY

Bell-LaPadula Model at glance

The access of users to resources is controlled by 2 rules:

A. *Simple property rule* (also referred to as **NO READ UP**): if a user has a certain clearance level, then the user **CAN NOT** access (read) a resource having a classification higher than the user clearance.

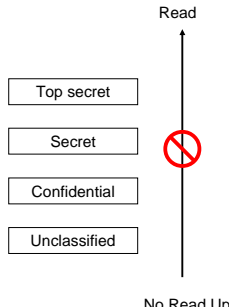
B. *Star property rule* (also referred to as **confinement property** or **NO WRITE DOWN**): if a user has a certain clearance level, then she cannot write (or create) a resource with a lower classification category

PURDUE UNIVERSITY

BLP Model : NO READ UP rule

A. **Simple property rule** (also referred to as **NO READ UP**) : if a user has a certain clearance level, then the user **CAN NOT** access (read) a resource having a classification higher than the user clearance.

- Examples:
 - a user having Clearance=Unclassified can not read a resource belonging to the Secret category
 - a user having Clearance=Confidential can read any resource belonging to the Confidential and Unclassified categories, but not a resource belonging to the Secret category

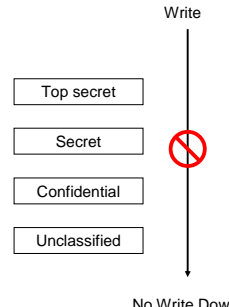


No Read Up

PURDUE UNIVERSITY

BLP Model : NO WRITE DOWN rule

A. **Star property rule** (also referred to as **confinement property** or **NO WRITE DOWN**): if a user has a certain clearance level, then she cannot write (or create) a resource with a lower classification category



No Write Down

Why Applying MAC to DB

- Data can be viewed as sensitive for many different reasons. Examples:
 - personal and private matters or communications, professional trade secrets,
 - company plans for marketing or finance,
 - military information, or government plans
- Such data is often mixed with other, less sensitive information that is legitimately needed by diverse users
- Restricting access to entire tables or segregating sensitive data into separate databases can create a working environment that is costly in hardware, software, user time, and administration.

Multilevel Relational Model

- The multilevel relational (MLR for short) model results from the application of the BLP model to relational databases
- Several issues
 - Granularity: to which element do we apply the classification?
 - Integrity constraints

MLR Model - notation

Standard relational model – each relation is characterized by two components

- A state-invariant *relation scheme*
 - $R(A_1, \dots, A_n)$ where A_i is an attribute over some domain D_i
- A state-dependent *relation* over R composed of distinct tuples of the form (a_1, \dots, a_n) , where each a_i is a value in domain D_i

MLR Model – keys and FD

- Functional dependencies
 - Let R be a relation and let X and Y be attribute sets, both subsets of the attribute set of R
we say that X **functionally determines** Y if and only if not two tuples may exist in the same relation over R with the same value for X but different values for Y
- Primary Keys (*entity integrity property*)
 - the primary key uniquely identifies each tuple in the relation
 - A primary key cannot contain attributes with null values
 - A relation cannot contain two tuples with the same value for the primary key

MLR Model

Given a relation, an access class can be associated with:

- The entire relation
- Each tuple in the relation
 - This is the common choice in commercial systems
- Each attribute value of each tuple in the relation
 - In the remainder we consider this case

Multilevel (ML) relations

A ML relation is characterized by two components

- A state-invariant *relation scheme*
 $R(A_1, C_1, \dots, A_n, C_n, TC)$ where:
 - A_i is an attribute over some domain D_i
 - C_i is a classification attribute for A_i ; its domain is the set of access classes that can be associated with values of A_i
 - TC is the classification attribute of the tuple
- A set of state-dependent *relation instances* R_c over R for each access class in the access class lattice. Each instance R_c is composed of distinct tuples of the form $(a_1, c_1, \dots, a_n, c_n, tc)$, where:
 - a_i is a value in domain D_i
 - c_i is the access class for a_i
 - tc is the access class of the tuple determined as the least upper bound of all c_i in the tuple
 - Classification attributes cannot assume null values

ML relations - example

Name	C _{Name}	Dept#	C _{Dept#}	Salary	C _{Salary}	TC
Bob	Low	Dept1	Low	100K	Low	Low
Ann	High	Dept2	High	200K	High	High
Sam	Low	Dept1	Low	150K	High	High

ML relations - instances

- A given relation may thus have instances at different access classes
- The relation instance at class c contains all data that are visible to subjects at level c
- That is, it contains all data whose access classes are dominated by c
- All elements with access classes higher than c , or incomparable, are masked by null values
- Sometimes, to avoid signaling channels, fictitious values (called *cover story values*) can be used

ML relations - instances

Name	C _{Name}	Dept#	C _{Dept#}	Salary	C _{Salary}	TC
Bob	Low	Dept1	Low	100K	Low	Low
Sam	Low	Dept1	Low	null	Low	Low

Low instance

ML relations - instances

Name	C _{Name}	Dept#	C _{Dept#}	Salary	C _{Salary}	TC
Bob	Low	Dept1	Low	100K	Low	Low
Ann	High	Dept2	High	200K	High	High
Sam	Low	Dept1	Low	150K	High	High

High instance

ML relations correctness conditions

ML relations must satisfy the following conditions:

- for each tuple in a ML relation, the attributes of the primary key must have the same access class
- for each tuple in a ML relation, the access class associated with the non-key attributes must dominate the access class of the primary key

ML relations keys and polyinstantiation

- In the standard relational model, each tuple is uniquely identified, by the values of its key attributes
- When access class are introduced, there may be the need for the simultaneous presence of multiple tuples with the same value for the key attributes but with different classification, which is phenomenon known as *polyinstantiation*

ML relations – polyinstantiation

Polyinstantiation occurs in the following two situations:

- When a low user inserts data in a field which already contains data at higher or incomparable level – *invisible polyinstantiation*
- When a high user inserts data in a field which already contains data at a lower level – *visible polyinstantiation*

ML relations invisible polyinstantiation

Suppose a low user asks to insert a tuple with the same primary key as an existing tuple at a higher level; the DBMS has three choices:

- 1) Notify the user that a tuple with the same primary key exists at higher level and reject the insertion
- 2) Replace the existing tuple at higher level with the new tuple being inserted at low level
- 3) Insert the new tuple at low level without modifying the existing tuple at the higher level (i.e. polyinstantiate the entity)

Choice 1 introduces a signaling channel

Choice 2 allows the low user to overwrite data not visible to him and thus compromising integrity

Choice 3 is a reasonable choice; as consequence it introduces a polyinstantiated entity

ML relations invisible polyinstantiation example

Name	C _{Name}	Dept#	C _{Dept#}	Salary	C _{Salary}	TC
Bob	Low	Dept1	Low	100K	Low	Low
Ann	High	Dept2	High	200K	High	High
Sam	Low	Dept1	Low	150K	High	High

Assume a low user issue the following insert operation

```
INSERT INTO Employee
VALUES (Ann, Dept1, 100k)
```

ML relations invisible polyinstantiation example

Name	C _{Name}	Dept#	C _{Dept#}	Salary	C _{Salary}	TC
Bob	Low	Dept1	Low	100K	Low	Low
Ann	High	Dept2	High	200K	High	High
Sam	Low	Dept1	Low	150K	High	High
Ann	Low	Dept1	Low	100K	Low	Low

The tuples with primary key "Ann" are *polyinstantiated*

ML relations visible polyinstantiation

Suppose a high user asks to insert a tuple with the same primary key as an existing tuple at lower level; the DBMS has three choices:

- 1) Notify the user that a tuple with the same primary key exists and reject the insertion
- 2) Replace the existing tuple at lower level with the new tuple being inserted at the high level
- 3) Insert the new tuple at high level without modifying the existing tuple at the lower level (i.e. polystantiate the entity)

Choice 1 does not introduce a signaling channel; however, rejecting the insertion may result in a DoS problem

Choice 2 would result in removing a tuple at lower level and thus introduce a signaling channel

Choice 3 is a reasonable choice; as consequence it introduces a polyinstantiated entity

ML relations – polyinstantiation

- The introduction of data classification in relational DBMS introduces polyinstantiation
- Several approaches have been developed to handle this problem
 - Approaches that allow polyinstantiation
 - Sandhu&Jajodia, SeaView Model by Denning et al.
 - These approaches define the key of a multilevel relation to be a combination of the original key attributes and their classifications
 - Belief-based model by Smith and Winslett
 - Approaches that prevent polyinstantiation
 - Require that all keys be classified at the lowest possible access class
 - Partition the domain of the primary key among the various access classes so that each value has a unique possible classification

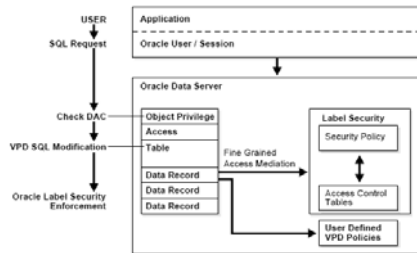
Case Study Oracle Label Security

Oracle label essential concepts

- Oracle Label Security enables row-level access control, based on the virtual private database technology of Oracle9i 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 Oracle9i applications

Oracle label architecture

Figure 1-2 Oracle Label Security Architecture



Label policy features

Oracle label controls the access to data by using 3 factors:

1. The label of the data row to which access is requested
2. The label of the user session requesting access
3. 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 group statements.

Level	Compartments (zero or more)	Groups (zero or more)
-------	--------------------------------	--------------------------

Example: The more sensitive the information, the higher its level.
 Confidential (10)
 Highly Confidential (20)
 Sensitive (30)
 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 of departments:
 - Financial (it has Sensitive and Highly Confidential data)
 - Chemical (it has Sensitive data)
 - Operation (it has Sensitive, Highly confidential and Confidential data).

Single-level ranking	Compartments (zero or more)	Groups (zero or more)
-------------------------	--------------------------------	--------------------------

Example:
 Confidential (10)
 Highly Confidential (20)
 Sensitive (30)

Example:
 Financial
 Chemical
 Operation

Data Labels - compartments

Levels:

Sensitive	Financial	Chemical	Operation
HC	Financial		Operation
Confidential			Operation

Data Labels - compartments

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 "father" of Finance and Engineering groups

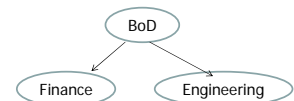
Data Labels – group example

Single-level ranking	Compartments (zero or more)	Groups (zero or more)
----------------------	-----------------------------	-----------------------

Example:
Confidential (10)
Highly Confidential (20)
Sensitive (30)

Example:
Financial
Chemical
Operation

Example:
BoD
Finance
Engineering



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

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

Type	Short	Long	Description
Maximum	HS	HIGHLY_SENSITIVE	User's highest level
Minimum	P	PUBLIC	User's lowest level
Default	C	CONFIDENTIAL	User's default level
Row	C	CONFIDENTIAL	Row level on INSERT

User Default Level: The level that is assumed by default when connecting to Oracle9i

User Default Row Level: The level that is used by default when inserting data into Oracle9i

User Labels and compartments

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"/>

- 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

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 checked="" type="checkbox"/>	WR_F...

- 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.

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*.

Table 3-3 Oracle Label Security Privileges

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.
 - useful for system administrators who need to export data, but who should not be allowed to 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*.

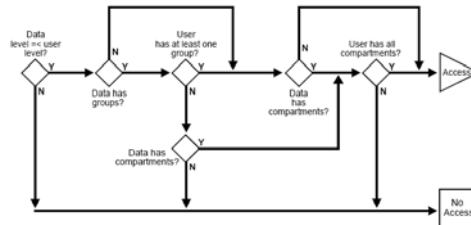
Privileges in Oracle Label Security Policies

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.

Privileges in Oracle Label Security Policies

Figure 3-9 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.

Privileges in Oracle Label Security Policies

- 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.
- 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. He can raise the level above his current session level, but cannot change the compartments.

Privileges in Oracle Label Security Policies

- 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.
- 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 10g
Release 10g Release 2 (10.2) B14267-02
http://www.oracle.com/pls/db102/to_pdf?pathname=network.102%2Fb14267.pdf&remark=portal+%28Administration%29